Internet of Things 35 (2026) 101841

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Decentralized proximity-aware clustering for collective self-federated learning

Davide Domini [a,*], Nicolas Farabegoli [a], Gianluca Aguzzi [a], Mirko Viroli [a], Lukas Esterle [b]

[a] Department of Computer Science and Engineering, University of Bologna - Cesena, via dell'Università 50, Cesena, 47521, (FC), Italy
[b] Department of Electrical and Computer Engineering & DIGIT, Aarhus University, Helsingforsgade 10, Building, 5123, room 414, 8200, Aarhus N, Denmark

ARTICLE INFO

2010 MSC:
00-01
99-00

Keywords:
Federated learning
Aggregate computing
Collective intelligence

ABSTRACT

In recent years, Federated Learning (FL) has emerged as a privacy-preserving paradigm for collaborative model training in IoT systems, enabling clients to learn a global model for tasks like classification, prediction, or anomaly detection in IoT environments without sharing raw data. However, traditional centralized FL architectures face bottlenecks, single points of failure, and struggle with non-IID data. These limitations hinder effective Collective Intelligence in large-scale IoT systems where numerous devices operate across diverse and dynamic environments. Existing clustered FL approaches often retain centralization or overlook how the spatial distribution inherent in IoT deployments directly influences data heterogeneity, challenging both the integration of spatially correlated devices and the establishment of intelligence distributed across the entire system. Creating such intelligence demands both decentralized architectures for scalability and effective integration of devices with similar data distributions. For these reasons, this article introduces *Proximity-Aware Self-Federated Learning (PSFL)*, a novel decentralized approach embodying collective intelligence principles. PSFL leverages field-based coordination to enable IoT devices to form *self-federations*, dynamically clustered groups that train specialized models based on both spatial proximity and local model characteristics. These self-federations reflect underlying data distributions, creating a distributed ecosystem of specialized models across the network. This approach overcomes global model limitations in non-IID settings through specialized federations based on local data distributions, enhancing performance while maintaining decentralization. We evaluate our approach using the Extended MNIST and CIFAR-100 datasets against state-of-the-art baselines, demonstrating its effectiveness in forming coherent, localized models under non-IID conditions.

1. Introduction

Research Context. The proliferation of Internet of Things (IoT) devices generates vast streams of data at the network edge, creating unprecedented opportunities for harnessing distributed information to build systems exhibiting collective intelligence (CI) [1]. Such systems, comprising numerous interconnected edge devices collaborating to achieve complex tasks, align with the vision of operating *without required reliance on a centralized cloud infrastructure*, a central theme for advancing intelligent IoT applications. However,

* Corresponding author.
E-mail addresses: davide.domini@unibo.it (D. Domini), nicolas.farabegoli@unibo.it (N. Farabegoli), gianluca.aguzzi@unibo.it (G. Aguzzi), mirko.viroli@unibo.it (M. Viroli), lukas.esterle@ece.au.dk (L. Esterle).

https://doi.org/10.1016/j.iot.2025.101841
Received 2 May 2025; Received in revised form 5 September 2025; Accepted 29 November 2025
Available online 5 December 2025
2542-6605/© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

conventional machine learning (ML) paradigms often demand data aggregation on central servers, posing significant privacy risks under regulations like the General Data Protection Regulation (GDPR) [2] and incurring substantial communication overhead. Federated Learning (FL) [3] has emerged as a key *advanced machine learning technique* for CI systems, allowing networks of clients (e.g., IoT devices) to collaboratively train shared models without exposing raw local data. These models can serve diverse purposes in IoT environments, from predicting sensor readings and detecting anomalies to classifying events, recognizing patterns, and enabling context-aware decision-making across distributed systems. This privacy-preserving approach is crucial for sensitive IoT domains like healthcare [4–8] and smart cities [9], where the ability to learn from collective data while preserving individual privacy enables more intelligent and responsive services.

Research Gap. Despite its potential, standard federated learning (FL) – especially when deployed in centralized architectures – encounters critical obstacles that hinder the development of effective CI within large-scale, dynamic IoT systems. Centralized control points introduce bottlenecks and single points of failure, compromising the *scalability and efficiency* required for vast device networks. More fundamentally, FL struggles with *non-Independent and Identically Distributed (non-IID)* data distributions [10] prevalent in real-world IoT deployments [11–13]. This data heterogeneity often arises directly from the *spatial distribution* of devices, sensors in close proximity naturally observe similar environmental conditions and thus collect statistically similar, locally relevant data [14]. Overlooking this inherent spatial correlation, which leads to non-IID data globally, severely impacts model accuracy and convergence [15], preventing the system from forming a cohesive and adaptive "distributed collective intelligence". While *clustered FL* methods [16,17] attempt to address non-IID data by grouping clients, they often retain centralized coordination or use synthetic partitioning strategies that fail to capture the complex interplay between spatial distribution and data statistics in dynamic IoT environments. This limits their ability to support truly *adaptive IoT systems through collective intelligence*.

Contribution. To address the limitations of existing approaches, this work introduces *Proximity-Aware Self-Federated Learning (PSFL)*. Our approach leverages advanced field-based coordination mechanisms from aggregate computing [18], a computational model for distributed systems that enables collective adaptive behaviors through a global-to-local programming paradigm. Using these principles, PSFL achieves two key objectives essential for enabling effective CI in decentralized IoT systems. First, it enables the *self-organizing formation* of learning federations (*self-federations*) directly among IoT devices. This organization is driven by the similarity of local data distributions (inferred through model characteristics), leveraging CI principles to ensure that data within a federation tends towards IID, while distributions between federations remain non-IID, thus better reflecting real-world spatial data heterogeneity.

Second, PSFL establishes a robust and adaptive *decentralized architecture* that eliminates reliance on a central server, directly addressing the need for *scalability and fault tolerance* in large-scale IoT deployments.

These goals are accomplished by employing Self-Organizing Coordinated Region (SCR) [19] and space-fluidity [20] principles from aggregate computing. These mechanisms facilitate the dynamic construction of distributed federations based on both *spatial proximity* and *data distribution similarity*. Crucially, this approach uses local model similarity metrics as a proxy for data similarity, guiding federation formation without requiring direct (and privacy-compromising) data sharing among nodes. Within each dynamically formed federation, a leader node is elected through the self-organizing process to act as the model aggregator, further enhancing system resilience against single points of failure.

To validate PSFL, we conducted extensive simulations using the Extended MNIST [21] and CIFAR-100 [22] datasets. We compare its performance against state-of-the-art FL algorithms, namely: FedAvg [3], FedProx [23], Scaffold [24], and IFCA [25], demonstrating its effectiveness in forming coherent, localized models in non-IID scenarios. Furthermore, we studied the effect of nodes movement during the training, showing that it does not influence the stability of formed federations. Finally, we performed resilience testing by simulating leaders failure, demonstrating the robustness inherent in our self-organizing architecture.

Paper structure. This manuscript is an extended version of the conference paper [26], providing:

(i) a more extensive and detailed coverage of related work;
(ii) an extended experimental evaluation adding more baselines (i.e., FedProx, Scaffold and IFCA);
(iii) a new experiment for testing the resilience of the self-organizing hierarchical architecture simulating aggregators failures.

The remainder of this paper is organized as follows: Section 2 illustrates a motivating example highlighting practical applications of our approach in smart city environments. Section 3 provides background on aggregate computing and federated learning while discussing related work. Section 4 details the proposed PSFL approach and its implementation. Section 5 presents an evaluation of our approach in a simulated setting. Section 6 discusses the limitations of our approach. Finally, Section 7 concludes the paper and outlines avenues for future research.

2. Motivating example: adaptive traffic management in smart cities 5.0

Imagine a futuristic smart city operating under a 5G (or beyond) infrastructure, populated by a vast number of autonomous vehicles (AVs) equipped with significant local intelligence. A primary goal for the city's collective system is to ensure smooth traffic flow, minimize congestion, and optimize travel times for all vehicles. To achieve this, each AV could leverage onboard machine learning (ML) models to continuously forecast near-term traffic conditions in its surroundings. Given the dense and dynamic nature of urban environments, and enabled by high-bandwidth, low-latency 5G communication, these AVs could potentially exchange information

directly with nearby peers (i.e., proximity-based communication), potentially reducing reliance on centralized cloud servers for coordination and leveraging the collective intelligence of the entire fleet. However, training these crucial traffic forecasting models presents significant challenges:

- **Privacy Concerns:** The raw data used for training (namely, detailed routes, speeds, destinations, times of travel) constitutes sensitive personal information. Sharing this data directly would violate user privacy regulations (like GDPR) and reveal individual habits. Federated Learning (FL) emerges as a natural candidate, allowing vehicles to collaboratively train models by sharing model updates (e.g., gradients or weights) instead of raw data.
- **Data Heterogeneity (Non-IID):** A standard FL approach typically aims to train a single global traffic model for the entire city. This is fundamentally flawed for traffic management. Traffic patterns are highly heterogeneous and localized. Downtown commercial districts experience vastly different congestion patterns and timings compared to residential suburbs, tourist areas, or highway corridors. Furthermore, these patterns change dynamically based on the time of day, day of the week, weather conditions, or special events. A single, monolithic model struggles to capture this rich local diversity, leading to poor prediction accuracy in specific zones. This spatial variation directly leads to non-Independent and Identically Distributed (non-IID) data across vehicles.
- **Scalability and Resilience:** Relying solely on a central server for orchestrating FL across potentially millions of vehicles in a large city introduces significant scalability bottlenecks and creates a single point of failure. Network latency or server downtime could cripple the entire learning process. A decentralized architecture which promotes collective intelligence among vehicles is far more suitable for such large-scale, dynamic systems.
- **Spatial Relevance:** Vehicles in close spatial proximity are more likely to experience similar immediate traffic conditions and thus possess locally relevant, albeit non-IID from a global perspective, data distributions. Collaboration among nearby peers seems inherently more valuable for learning localized patterns than averaging updates from vehicles across the entire city with vastly different experiences.

Simply applying standard FL is insufficient, due to the fundamental challenges in terms of scalability, adaptability, and contextual relevance [27,28]. We need a more nuanced approach where vehicles can intelligently collaborate. Instead of one global model, the system should foster the emergence of multiple, specialized models, each tailored to specific traffic patterns prevalent in different geographical areas or under particular conditions. Vehicles should dynamically group themselves into collaborative federations based not only on spatial proximity but also on the similarity of their local traffic experiences, as reflected in the effectiveness or characteristics of their local ML models. For instance, a car approaching an intersection might have a very different experience than a car on the perpendicular road approaching the same intersection. While both cars might be in relative spatial proximity, their experience of traffic might be very different. Creating federations based on both, the spatial proximity and the similarity of their local traffic experiences improves the overall performance across the entire system.

For these reasons, a novel learning paradigm like Proximity-Aware Self-Federated Learning (PSFL) is necessary. Notably, while this example focuses on CI in future traffic management, the illustrated principles can be generalized to all kinds on IoT devices. Our proposed system exhibits the following capabilities:

(i) **Decentralization:** Operate without reliance on a central server for model aggregation or federation management, therefore promoting collective intelligence among devices and enhancing scalability and resilience.
(ii) **Adaptive Proximity-Aware Clustering:** Enable devices to dynamically form and join learning federations based on both their geographical proximity and the similarity of their learned models (reflecting shared data distribution characteristics), leading to specialized and accurate local models.
(iii) **Dynamic Adaptation and Robustness:** Adapt the federation structures and models over time to reflect evolving patterns and remain robust to network dynamics, device mobility, and potential device failures or departures.

Research Questions. Building upon the challenges identified in our motivating example, this paper aims to develop a solution that addresses both the technical and practical aspects of enabling collective intelligence in spatial IoT deployments. To guide our investigation, we formulate the following research questions:

1. (RQ1) Can devices be clustered based on both spatial proximity and data distribution similarity in a fully decentralized manner?
2. (RQ2) Can a clustering strategy enhance global model accuracy when data distributions are non-IID?
3. (RQ3) Can a self-organizing architecture improve system robustness and fault tolerance?

By addressing these questions, we aim to create a federated learning approach that leverages spatial relationships and data similarity to form effective learning federations without central coordination, while maintaining resilience against network dynamics and node failures.

3. Background and related work

This section introduces the system model targeted by our approach, discusses aggregate computing paradigms for programming robust collective behaviors, and reviews federated learning techniques for distributed model training in heterogeneous environments.

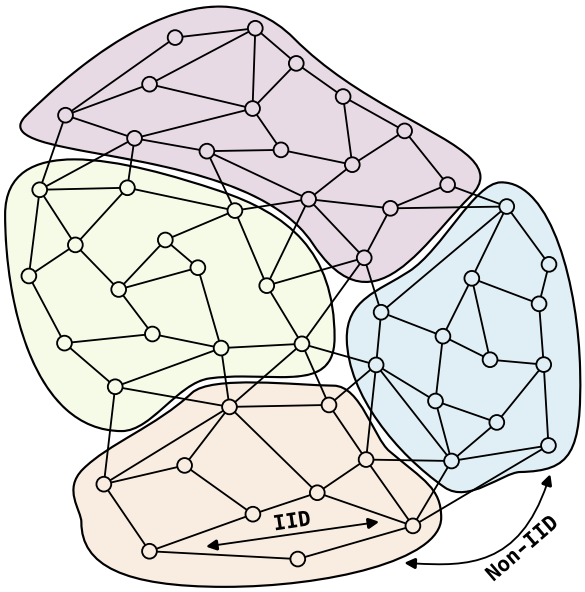


Fig. 1. This figure depicts the system model targeted by our approach, where a set of IoT devices (circles) are deployed in a spatial area A. Each area a_j (colored regions) has a unique data distribution P_j.

3.1. System model

The system we address in this paper can be classified as a *collective adaptive system* (CAS) [29], composed of numerous autonomous entities operating collaboratively. In this paper we consider a specific kind of CAS, where the entities are *IoT devices* equipped with sensors and actuators, capable of collecting data from the environment and performing actions based on the collected data (Fig. 1 and Table 2).

Particularly, as depicted in Fig. 6, we define a spatial area $A = \{a_1, a_2, \dots, a_k\}$ divided into k distinct continuous regions.

Each region a_j is associated with a probability distribution P_j (thereafter, also called data distribution) over some sample space $\mathcal{X}$: the data points observed in region a_j can be modeled as random variables $X_j^{(1)}, X_j^{(2)}, \dots$ that are independently and identically distributed according to P_j.

For distinct regions a_i and a_j ($i \neq j$) the corresponding distributions P_i and P_j differ significantly (i.e., they are *non-IID*). Formally, there exists a statistical divergence measure $D(\cdot\|\cdot)$ (such as Kullback-Leibler divergence or total variation distance) and a threshold $\epsilon > 0$ such that: $\forall i \neq j : D(P_i\|P_j) \geq \epsilon$. Within area A, we deploy a set of *IoT devices* $S = \{s_1, s_2, \dots, s_n\}$ ($n \gg |A|$), each capable of processing data and participating in FL. Each device s_i is positioned at coordinates $p_i = (x_i, y_i)$ in the environment and resides in a specific region a_j, though this region information is not explicitly available to the devices themselves.

For any device s_i located in region a_j, the local dataset D_i consists of samples drawn from distribution P_j. Consequently, devices in the same region have statistically similar data, while devices in different regions have statistically dissimilar data, leading to a non-IID data scenario across the entire system. Finally, each device s_i has a dynamic neighborhood N_i of devices it can directly communicate with. In the remainder of this paper, without loss of generality, we shall assume N_i is the neighborhood induced by a physical communication range r_c.

For the execution model, we consider a *distributed* and *asynchronous* approach, where each node independently performs computations and communicates with its neighbors. The evaluation cycle for each device consists of:

- **Sensing**: The device collects environmental data and retrieves recent messages from neighboring devices;
- **Computing**: The device processes information based on its local context (neighbor messages and sensor readings); and
- **Acting**: The device determines appropriate actions and prepares messages to be sent to neighbors.

This model abstracts away details that can be configured based on specific applications, such as neighbor relationships, round frequency, and message retention time, and can be applied to several domains, including smart cities [9], swarm robotics [30], and data centers [31]. The specific computation performed depends on the application requirements. In this paper, we utilize the *aggregate computing* paradigm [18], which provides powerful abstractions for expressing self-organizing computations through the functional composition of modular building blocks.

3.2. Aggregate computing in a nutshell

Aggregate Computing [18] has emerged as a powerful paradigm for programming and engineering CASs. It roots its formal foundation in the *field calculus* [32], and comes with a practical toolchain (like ScaFi [33], a Scala DSL implementing the field calculi) that allows to implement distributed systems with intrinsic self-adaptive and self-organising properties. It adheres to the concept of macroprogramming [34], where a single specification is used to program the collective behaviour of the system—all individuals executes the same program, resulting in an overall resilient coordination of activities. This paradigm provides three main primitives:

- **Neighbor interaction**: this primitive enables nodes to interact with neighbors, by sharing a value (e.g., a sensor reading);
- **State preservation**: this primitive allows nodes to preserve their state over time;
- **Network partitioning**: this primitive allows to partition the network based on a given condition.

For instance, a valid program leveraging the field calculus primitives is shown below:

```
def distanceFromSource(): Double = branch(!sense("obstacle")) {
    rep(Double.PositiveInfinity) { gradient =>
        mux(sense("source"))
            { 0 }
            { minHoodPlus(nbr(gradient) + nbrRange) }
    }
} { Double.PositiveInfinity }
```

This program computes the shortest distance from any node to a source node in a network, while avoiding obstacles. It is executed as a macroprogram on each node in the network, using the following field calculus primitives:

- |branch| partitions the network based on a condition (here, absence of obstacles);
- |rep| maintains state over time (storing the current distance estimate);
- |nbr| enables neighborhood interaction (sharing distance values between adjacent nodes).

The algorithm works by initializing all nodes with an infinite distance. Source nodes set their distance to 0, while other nodes calculate their distance by finding the minimum value among neighbors' distances plus the physical distance to those neighbors. This creates a potential field that can guide navigation toward the source. For a more comprehensive explanation of field calculus primitives and their formal semantics, the reader may refer to Viroli et al. [32].

Aggregate Computing is particularly well-suited for CASs not only due to its practical top-down programming approach, but also because it offers formal guarantees about the behaviors it can express. Indeed, based on these fundamental primitives, developers can compose higher-level abstractions (or building blocks) to express complex distributed behaviors more concisely [34]. The most used building blocks (see also Fig. 2 for a visual representation) are:

Sparse-choice — `S`. A scattered selection from the set of participating devices. This building block, named `S` for *sparse-choice*, is defined as follows:

```
def S(grain: Double, metric: () => Double): Boolean
```

The first parameter, `grain`, defines the desired diameter of the regions we want to create. The second parameter, `metric`, is a function that computes a distance metric between nodes; by varying this parameter, we can obtain variants, like the one based on *space-fluid sampling* [20], where the leader is selected based on a perception of a spatially distributed phenomenon. The returned value is a `Boolean` indicating whether the node is selected as a leader or not.

Gradient-cast — `G`. A multicast diffusing information along a gradient. This building block, named `G` for *gradient-cast*, is defined as follows:

```
def G[V](
    source: Boolean,
    value: V,
    accumulator: V => V,
    metric: () => Double,
): V
```

The `source` parameter indicates whether the node is the source of information. The `value` parameter specifies the information to be diffused. The `accumulator` parameter is a function that specifies how the information is propagated along the gradient, and the `metric` parameter defines the concept of distance between nodes. The returned value holds the information diffused so far along the gradient.

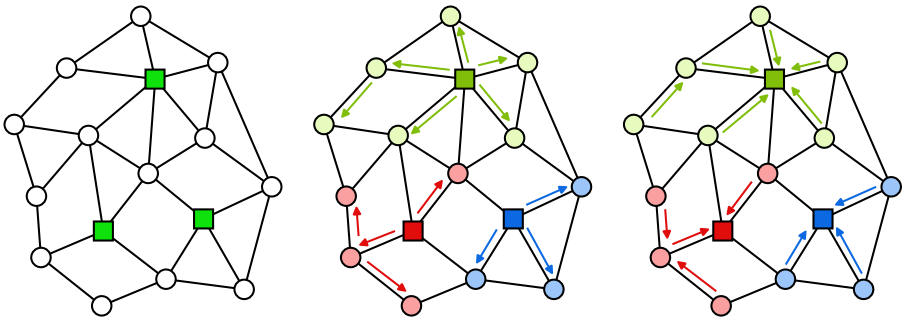


Fig. 2. Visual representation of the building blocks used in our approach. The leftmost block represents the *S* building block, which selects a subset of nodes based on a spatially distributed phenomenon. The middle block represents the *G* building block, which diffuses information along a gradient. The rightmost block represents the *C* building block, which aggregates information to a sink device.

Converge-cast — C. A multicast aggregating information to a sink device. This building block, named `C` for *converge-cast*, is defined as follows:

```
def C[P, V](
    potential: P,
    accumulator: (V, V) => V,
    local: V,
    neutral: V,
): V
```

The `potential` parameter indicates the potential of the node to be a sink. In other words, the potential is backtracked to reach the sink node. The `accumulator` parameter specifies how the information is aggregated along the path to the sink node. The `local` parameter specifies the local data to be aggregated, while the `neutral` parameter specifies the neutral element for the aggregation operation. The returned value of this function in the sink node is the aggregated value of the information collected from all nodes covered by the `potential`. In all the other nodes, the returned values it is a partial accumulation of the information collected along the path to the sink node.

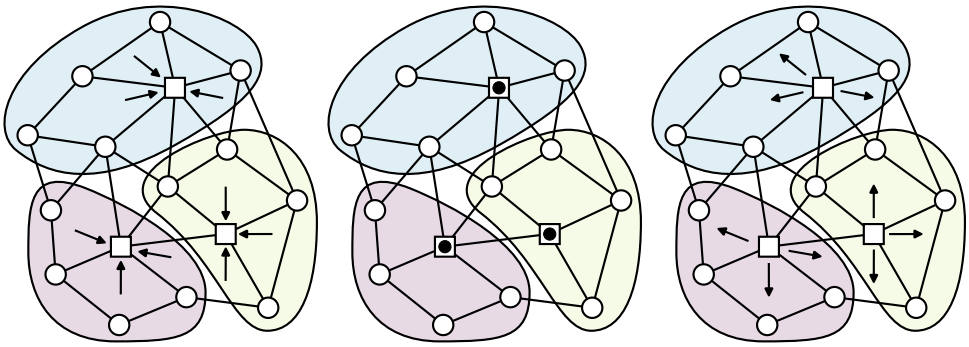
A description on how these building blocks can be used and combined to express more complex collective behaviors is provided in Section 3.3.

Building blocks properties. These building blocks exhibit robust *self-stabilizing* characteristics [35], demonstrating convergence to stable computational states within finite time across various initial conditions. They also show *eventual consistency* [36] properties, where computations tend toward equivalent results despite variations in network topology, density, or node population. The field calculus framework has been extensively studied for its space-time universality [37], showing its effectiveness in expressing computational functions within distributed environments. These well-established behaviors provide important foundations for designing robust and resilient distributed systems, which motivated our selection of this computational paradigm. Furthermore, this approach enables the expression of complex self-organizing behaviors through functional composition of modular computational elements, enhancing both development modularity and component reusability. A particularly relevant application of this compositional approach is the *Self-organizing Coordination Regions* (SCR) pattern [19], which provides structured mechanisms for spatial partitioning and coordination.

3.3. Self-organising coordination region

This paper addresses the challenges of distributed cooperative learning in modern highly distributed systems such as IoT systems (or pervasive computing, collective adaptive systems, cyber-physical systems, or edge computing). These systems exhibit *distribution*, *situatedness*, and *scale*, posing coordination challenges [19,38] including: the *locality principle* where operational efficiency depends on proximity; balancing centralized and decentralized *control and decision-making*; and adapting to *dynamic environments* with constant changes. A common solution involves *dynamically* partitioning networks into contiguous regions (i.e., forming a Voronoi-like partitioning of the network) for efficient node cooperation. This approach proves effective where data distribution changes rapidly over time, enabling decentralized services orchestration [39], organized communication in WSNs [40], distributed IoT data processing [9], and dynamic robot swarm control [30].

An example solution to achieve such partitioning without global supervisioning is the *self-organizing coordination region* (SCR) [19] pattern, which works as follows: a system-wide *multi-leader election* process determines a set of leaders among a set of leader candidates; the system (or its environment) forms a self-organising set of regions, each one regulated by a *single* leader; within each region, a



(A) Information collection. (B) Leader computation. (C) Information sharing.

Fig. 3. Graphical representation of the Self-organizing Coordination Regions pattern. First, information within each area is collected in the respective leader. Then, each leader processes the collected information and shares it back to the clients.

feedback loop is established, whereby the leader receives *upstream information flows* from the members of the region (possibly leveraging intermediaries) and emits a *control information flow downstream*—see Fig. 3.

Information within these regions is disseminated primarily through *gradient-based information casts* [41], where they propagate information across increasing distances, useful for defining regional boundaries and leader-user communication.

Accumulating information, especially in larger networks, poses greater challenges. While gossip methods can suffice in small regions, scalable solutions like *spanning tree* and *multi-path techniques* are necessary for larger areas.

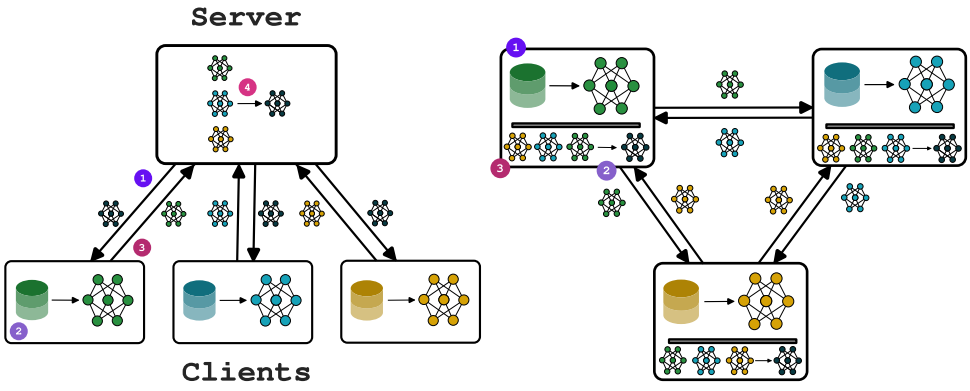
SCR can be implemented using various programming paradigms, but it is particularly effective when using *aggregate computing* [18], thanks to its ability to express self-organizing computations through the functional composition of modular building blocks presented in Section 3.2. A simple, yet effective, implementation of SCR is shown below:

```
1  val area = 100
2  // each area is (at least) 100 meters large
3  val leaders = S(area, nbrRange)
4  // each region is defined by a gradient from its leader
5  val region = G(leaders, 0, _ + nbrRange)
6  // each leader collects information from its region
7  val collected = C(region, _ ++ _, Set(data), Set.empty)
8  // each leader computes a local decision with the collected data
9  val decision = leaderLocalComputation(collected)
10 // each leader sends the decision to its region
11 val feedback = G(leaders, decision, _ + nbrRange)
```

In line 3 the `S` building block is used to identify the leaders of the regions. The election tries to select the best leader candidate so that regions of at least 100 m are formed. The adopted metric is the `nbrRange` meaning that the regions are determined by the physical distance between the nodes.

Then, in line 5, the `G` building block is used to propagate the information from the leaders to the nodes in their regions. In the example, each node in the region computes the distance from its leader. In line 7, the `C` building block is used to collect the information from the nodes in the region to the leader. This way, each region's leader collects the `data` from all the nodes in its region. The `_ ++ _` represents the union of two sets, and it is used to accumulate the information collected from the nodes in the region. The line 9 shows how the leader can compute a *local* decision based on the collected information, and at line 11 the leader sends the decision back to the nodes in its region using the `G` building block.

In this example, however, the region definition is based on physical space, where area boundaries correspond to spatial distances between nodes. More sophisticated variants of the SCR pattern can be employed to define regions based on different criteria, such as the *space-fluid* approach [20], which allows for defining regions based on domain-specific parameters rather than just spatial proximity. This approach generalizes the distance metric between nodes to account for multiple factors relevant to the application context, such as the similarity of data distributions.



(A) Centralized Federated learning schema. (B) Peer-to-peer federated learning schema.

Fig. 4. On the left, a visual representation of Centralized Federated Learning. In the first phase, the server shares the centralized model with the clients. In the second phase, the clients perform a local learning phase using data that is not accessible to the server. In the third phase, these models are communicated back to the central server, and finally, in the last phase, there is an aggregation algorithm. On the right, the P2P Federated Learning schema. Differently from the centralized version, there is no central server. Each client sends its local model to all the other clients, then the aggregation is performed locally.

3.4. Federated learning

Federated learning [3] is a machine learning framework that aims to collaboratively train a unique global model from distributed datasets. This technique has been introduced to enable training models without collecting and merging various datasets into a single central server, thus making it possible to work in contexts with strong privacy concerns (e.g., healthcare [4] or banking data [42]). Furthermore, in highly distributed systems, given the large amount of data that can be generated by various clients, it becomes infeasible to move all data to a single central server [12]. In this context, FL becomes crucial as it allows leveraging the computational capabilities of the various clients.

Even though various versions of FL exist [43,44], they all share a common learning flow. Traditionally, FL is based on a client-server architecture (Fig. 4A) and comprises the following steps:

1. *Model initialization*: the central server initializes a common base model that is shared with each client;
2. *Local learning*: each client performs one or more steps of local learning on its own dataset;
3. *Local models sharing*: each client sends back to the central server the new model trained on its own data (i.e., the local model);
4. *Local models aggregation*: the local models collected by the central server are aggregated to obtain the new global model.

The classical client-server architecture presents some limitations, namely:

(i) in federations with many clients, the server may be a bottleneck;
(ii) the server is a single point of failure, and if it fails to communicate with the clients, the entire the learning process is interrupted.

For these reasons, various distributed federated learning approaches have been introduced [45,46], based on peer-to-peer (Fig. 4B) or semi-centralized architectures. Hierarchical and multi-tiered approaches have also emerged to address network heterogeneity and improve scalability [47].

3.5. Data heterogeneity

Federated learning algorithms achieve remarkable performance when data are homogeneously distributed among clients [48]. However, in real-life datasets, data are often heterogeneous, namely *non-independently and identically distributed* (non-IID).

Data heterogeneity can be caused by various factors [49]. Beyond the smart traffic flow management example discussed in Section 2, numerous real-world applications exhibit similar heterogeneity. For instance, Fig. 5 presents CO_2 emissions per kilowatt hour of electricity load across various European countries [50]. The evident variability in emission profiles illustrates proximity-based non-IID data: within-country data tend to be homogeneous, while inter-country distributions diverge significantly. Comparable non-IID patterns emerge in domains such as PM10 air quality sampling and heating systems for buildings in smart cities, where regional characteristics–like district-specific infrastructure or climate–drive distinct data distributions.

Data heterogeneity can be categorized in various ways based on how the data are distributed among the clients [10,15]. The main categories include:

(i) *feature skew*: all clients have the same labels but different feature distributions (e.g., in a handwritten text classification task, we may have the same letters written in different calligraphic styles);

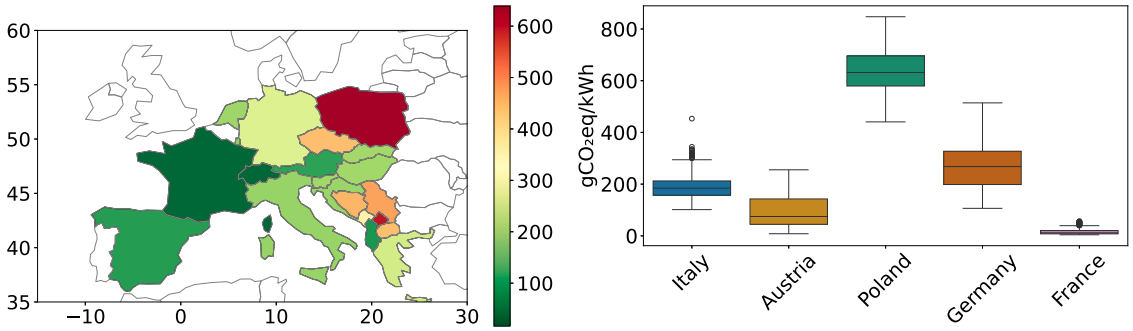


Fig. 5. CO_2 emissions per kilowatt hour of electricity load across various European countries, illustrating proximity-based non-IID data. Emission patterns vary significantly between countries, reflecting real-world heterogeneity relevant for federated learning.

(ii) *label skew*: each client has only a subset of the classes; and
(iii) *quantity skew*: each client has a significantly different amount of data compared to others.

Non-IID data is a critical aspect because in such scenarios, individual clients have local objectives that diverge from the global one [10]. As a result, after the local training, the clients produce vastly different updates (i.e., *local update drift*) that, once aggregated, lead to a reduction in the accuracy of the global model or convergence issues [15].

For these reasons, FL with non-IID data is a growing research field. In the literature, there are various families of approaches aiming to enhance learning in these scenarios. One approach involves attempting to train a single global model by improving base algorithms (e.g., FedAvg) through the addition of regularization terms to constrain the local updates or considering that each device may perform a different number of local training steps. This is implemented by various algorithms, including: FedProx [23], Scaffold [51], and FedNova [52]. Another possible approach–the one we focus on–involves dividing the devices into clusters and having multiple global models, one for each cluster, in order to create specialized models.

3.6. Clustered federated learning

Clustered federated learning is an approach to tackle data heterogeneity. It is a technique of personalized FL in which, instead of a single global model, *multiple* models are trained to target various local distributions. It is based on the assumption that clients can be divided into subgroups (i.e., *clusters*), and that within each cluster, clients have data that follow similar distributions (i.e., *IID* data). Moreover, studies [25,53,54] show that aggregating models among clients within the same cluster leads to improved performance and personalization; while aggregating models from different clusters results in a degradation of the accuracy of the global model. A key aspect of these approaches is how to measure the *similarity* among various clients. Generally, there are three methods [55], namely:

(i) *loss-based*: clients measure the losses of the models from various clusters on their own data, with the model having the lowest loss considered most similar [25];
(ii) *gradient-based*: clients measure similarity based on the distance between gradient updates [53,56]; and
(iii) *weight-based*: clients measure similarity based on the weights of their models [53,57].

In literature, several algorithms based on client clustering have been proposed, such as: PANM [55], FedSKA [58], IFCA [25]. Recent advances include clustered federated learning approaches that address heterogeneous environments [59] and greedy agglomerative frameworks for improved clustering [60]. While these algorithms perform well when the number of clusters matches the true number of underlying data distributions, their performance degrades significantly when this assumption is violated. This poses a major challenge in distributed and dynamic environments, where determining the correct number of clusters is inherently difficult. Moreover, most of these algorithms rely on a client-server architecture.

3.7. Baseline algorithms

This section describes the baseline algorithms used for comparison in this work. Table 1 summarizes the main characteristics we focus on, comparing the baselines with the proposed approach.

FedAvg. One of the most common algorithms for FL is *FedAvg*, where the server aggregates the models by averaging the weights of the models received from the client devices. Formally, given a population of K devices, each of them with a local dataset D_k. Each device k performs E epochs of *local training* on its dataset, computing updates to the weights of its local model, denoted as $\mathbf{w}_k^t$ at the end of the t-th global round. The server then computes the weights of the global model $\mathbf{w}^{t+1}$ as the average of the weight vectors from

Table 1
Comparison of federated learning approaches.

Method	Decentralized	Non-IID	Clustered	Spatial Correlation
FedAvg	✗	✗	✗	✗
FedProx	✗	✓	✗	✗
Scaffold	✗	✓	✗	✗
IFCA	✗	✓	✓	✗
PSFL	✓	✓	✓	✓

the local models shared by the devices:

$$\mathbf{w}^{t+1} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{w}_k^t. \tag{1}$$

This process is repeated for a fixed number of rounds, with the server distributing the global model to the devices at the beginning of each round.

FedProx. This algorithm extends FedAvg by introducing a proximal term to the objective function that penalizes large local model updates. This modification helps address statistical heterogeneity across devices while safely accommodating varying amounts of local computation due to system heterogeneity.

Formally, the local objective function for device k in FedProx is:

$$\arg\min_{\mathbf{w}} h_k(\mathbf{w};\mathbf{w}_t) = F_k(\mathbf{w}) + \frac{\mu}{2}\|\mathbf{w}-\mathbf{w}_t\|^2 \tag{2}$$

where $F_k(\mathbf{w})$ is the local loss function, $\mathbf{w}_t$ is the global model at round start, and $\mu \geq 0$ is the proximal term coefficient that controls how far local updates can deviate from the global model.

Scaffold. This algorithm introduces control variates to correct model drift caused by data heterogeneity. Unlike FedAvg, where each client performs model updates locally and communicates its version to the server, Scaffold tries to minimize this drift using control variates that track the direction of gradient descent for each client. In the local model update phase, each client i adjusts its model $\mathbf{y}_i$ using the formula $\mathbf{y}_i \leftarrow \mathbf{y}_i - \eta_l(g_i(\mathbf{y}_i) + \mathbf{c} - \mathbf{c}_i)$, where η_l is the local learning rate, $g_i(\mathbf{y}_i)$ represents the local gradient, and $\mathbf{c}$ and $\mathbf{c}_i$ are the server and client control variates respectively. The client's control variate $\mathbf{c}_i$ is then updated using either $\mathbf{c}_i^+ \leftarrow g_i(\mathbf{x})$ or $\mathbf{c}_i^+ \leftarrow \mathbf{c}_i - \frac{1}{K\eta_l}(\mathbf{x}-\mathbf{y}_i)$, where $\mathbf{x}$ is the server's model and K represents the number of local updates. Finally, the server performs global updates. The global model is updated as $\mathbf{x} \leftarrow \mathbf{x} + \eta_g \frac{1}{|S|}\sum_{i\in S}(\mathbf{y}_i - \mathbf{x})$, where η_g is the global learning rate and S is the set of selected clients. Simultaneously, the server control variate is adjusted using $\mathbf{c} \leftarrow \mathbf{c} + \frac{1}{N}\sum_{i\in S}(\mathbf{c}_i^+ - \mathbf{c}_i)$, with N being the total number of clients.

IFCA. IFCA algorithm is based on two main assumptions:

(i) a single global model may not perform optimally across diverse client data distributions; and
(ii) clients can be partitioned into clusters where the data within each cluster are independently and identically distributed.

Accordingly, the algorithm aims to train a separate model for each cluster and to determine the appropriate cluster membership for each client device. In each communication round, the central server broadcasts all cluster models to the clients. Each client evaluates these models on its local data and selects the one yielding the lowest loss, thereby determining its cluster affiliation. Clients then perform local updates on the selected model and send the updates back to the server. The server aggregates these updates separately for each cluster, typically using the FedAvg algorithm, to refine the corresponding cluster models. This iterative process continues, allowing both the models and the client-cluster assignments to converge over time.

4. Proximity-aware self-federated learning

4.1. Problem statement

Building on the system model described in Section 3.1, we now examine how to perform a federated learning process in a distributed and asynchronous manner in such a system. (see Fig. 6).

Each device s_i is positioned in some region a_j and thus observes data generated according to the corresponding probability distribution P_j. Locally, each device s_i creates a dataset D_i consisting of samples drawn from this distribution. Herein, we consider a classification task where each sample in the dataset D_i consists of a feature vector $\mathbf{x}$ and a label y. Therefore, the complete local dataset is represented as $D_i = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$.

For each global round t, the devices should devise a set of *federations* $F^t = \{f_1^t, \ldots, f_m^t\}$, where each federation f_j is represented as a tuple (l_j, S_j), with:

(i) l_j the leader device responsible for the federation; and

Table 2
Summary of symbols in PSFL.

Symbol	Description
A	Spatial area divided into k distinct contiguous areas
a_j	A specific region within A
P_j	Probability distribution associated with region a_j
$X_j^{(i)}$	Random variables distributed according to P_j
D_i	Local dataset of node s_i with samples from distribution P_j
S	Set of IoT devices deployed in the area A
s_i	An IoT device, part of the set S
r_c	Communication range of an IoT device
N_i	Neighborhood of IoT device i
$\mathcal{M}_i^T$	Local model of IoT device i at time step t
T	Total number of time steps in the learning process
F^t	Set of federations at time step t
f_j^t	A specific federation at time step t
l_j	Leader node of federation f_j
$\mathcal{M}^T$	Set of all federation-wise models at final time step T
$L(D_i, \mathcal{M}_i^T)$	Average error of model nn_i^T on data distribution Θ_i
$ds(i,j)$	Dissimilarity measure based on loss functions $L_{i,j}$ and $L_{j,i}$
G	Gradient field indicating accumulated error to a l_i
σ	Maximum tolerable path error for federation coherence

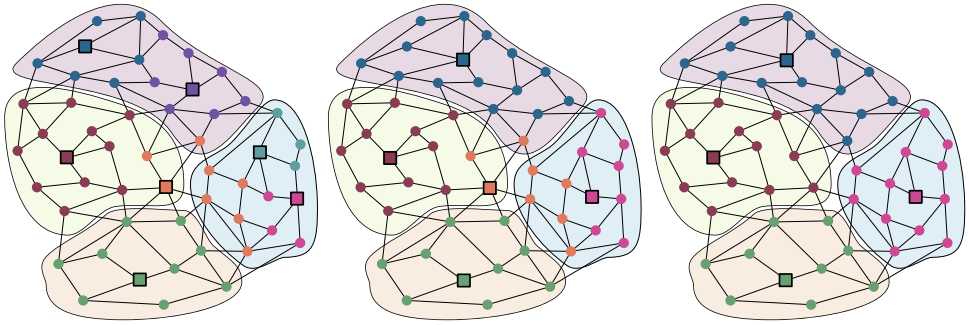


Fig. 6. Graphical representation of the problem. Nodes color represents the federation they belong to, while the background color represents the data distribution of the area. Squared nodes represent the leaders of the federations. During the learning process (time flows from left to right), the nodes begin to form various federations eventually reaching a stable division that matches the areas they belong to.

(ii) S_j the set of nodes that are part of the federation.

Given T the number of global rounds, the goal of our approach is to create a set of federations F^T which approximates the subregions A and find the set of federation-wise models $\{\mathcal{M}_1^T, \ldots, \mathcal{M}_m^T\}$ (namely, one model for each federation). Consider a correct partitioning of federations, we want to find a set of federation-wise models that minimize the expected error across all areas with respect to their underlying probability distributions, which can be formally described as:

$$\min_{\mathcal{M}^T} \sum_{j=1}^{k} \mathbb{E}_{(x,y)\sim P_j} \ell(f(x; \mathcal{M}_j^T), y) \tag{3}$$

Where $\mathbb{E}_{(x,y)\sim P_j}$ denotes the expectation over the probability distribution P_j of area a_j, ℓ is a suitable loss function, and $f(x; \mathcal{M}_j^T)$ represents the prediction made by model $\mathcal{M}_j^T$ for input x.

4.2. Proposed solution

PSFL addresses the challenge of organizing nodes into coherent federations that maximize learning efficacy through exploitation of data distribution similarities. We formulate this as a decentralized problem using the Self-organizing Coordination Regions (SCR) paradigm [19], where federations represent dynamic learning regions with homogeneous data characteristics. Our approach follows a structured distributed protocol comprising four principal phases, implemented in ScaFi [33] through field calculus primitives (Listing 7):

1. *(Self-)Federation formation phase:* nodes participate in a distributed multi-leader election process governed by a space-fluid metric that accounts for both topological proximity and model similarity. This process establishes federation boundaries and designates leaders responsible for aggregation operations. They are “self” federated since the nodes autonomously decide to join a federation based on their data distribution.

2. *Model collection phase:* federation leaders establish collection paths through which they gather trained models from constituent nodes, utilizing converge-cast patterns to efficiently aggregate distributed information.
3. *Federation-wise aggregation phase:* leaders perform weighted model aggregation operations on collected models to derive a unified federation-specific model that captures the collective knowledge within each federation's data distribution.
4. *Model dissemination phase:* the aggregated models are propagated from leaders to all federation members through gradient-cast operations, ensuring uniform model representation across each federation.

4.3. Space-fluid self-federation

While phases 2 through 4 represent standard operations in federated learning, the key innovation of PSFL lies in the federation formation process (phase 1). This critical phase adaptively creates federations that align with the underlying data distributions across nodes. To implement this data-aware federation formation, we leverage a decentralized multi-leader election mechanism based on the space-fluid sparse choice algorithm [20]. In our experiments, since all nodes have identical computational characteristics, the leader within each cluster is selected as the node with the smallest identifier. However, the mechanism is fully dynamic and can incorporate additional criteria, such as prioritizing nodes with higher computational capacity, to exclude resource-constrained nodes in more heterogeneous scenarios.

Model Dissimilarity Metric. A critical aspect of the space-fluid paradigm is its use of topological distance metrics that transcend mere physical proximity. Instead of conventional Euclidean measures, our approach employs a semantically richer distance function that captures model compatibility between adjacent nodes. This function is defined as a *dissimilarity* measure between two nodes' models, computed from the cross-evaluation of each model on the other node's dataset.

Formally, we define a loss-based dissimilarity measure $ds(i, j)$ between nodes i and j as:

$$ds(i, j) = L_{i,j} + L_{j,i} \tag{4}$$

where $L_{i,j}$ represents the average loss incurred when applying model $\mathcal{M}_j$ to node i's local dataset D_i:

$$L_{i,j} = L(D_i, \mathcal{M}_j) = \frac{1}{|D_i|} \cdot \sum_{(x,y)\in D_i} \ell(f(x; \mathcal{M}_j), y) \tag{5}$$

Here, $f(x; \mathcal{M}_j)$ denotes the prediction generated by model $\mathcal{M}_j$ for input x, and ℓ represents a suitable loss function.

Notably, the proposed approach is agnostic to the specific dissimilarity measure employed. In fact, other dissimilarity measures can be used, for instance measuring differences in model weights or gradients.

Gradient Field Propagation. Given this dissimilarity measure between neighboring nodes, we now extend it to define the federation formation process across the entire network. Rather than relying solely on local pairwise comparisons, we employ *gradient fields* [41] to propagate dissimilarity information throughout the network, implemented using the gradient-cast (G) building block described in Section 3.2.

Formally, we define a gradient field $G : N \to \mathbb{R}^+$ that propagates from federation leaders through the network according to the dissimilarity measure. For each node $n \in N$, the value $G(n)$ represents the cumulative dissimilarity along the minimum-dissimilarity path from n to its nearest federation leader. This can be expressed recursively as:

$$G(n) = \begin{cases} 0 & \text{if } n \text{ is a leader} \\ \min_{m\in \mathrm{Nbr}(n)}\{G(m) + ds(n, m)\} & \text{otherwise} \end{cases} \tag{6}$$

where $\mathrm{Nbr}(n)$ denotes the set of neighbors of node n. With this definition, spatial and model coherence emerge naturally through the gradient field:

- Nodes with similar models (i.e., low dissimilarity) will have lower $G(n)$ values, leading to a higher likelihood of forming federations together.
- Nodes with dissimilar models (i.e., high dissimilarity) will have higher $G(n)$ values, making them less likely to be part of the same federation.
- The accumulated dissimilarity increases with path length, naturally incorporating both topological proximity and model similarity in federation formation.

The G value thus encodes a dual constraint: it depends on both the model similarity between neighboring nodes and the topological distance (number of hops) to the federation leader, creating federations that are both spatially compact and distributionally coherent.

Establishing Federation Boundaries. Federation boundaries are established using a threshold parameter $\sigma \in \mathbb{R}^+$, which represents the maximum permissible cumulative dissimilarity within a federation. A node n joins a federation with leader l if and only if $G(n) \leq \sigma$, ensuring that federation membership incorporates both model similarity and network proximity constraints.

Federation Formation Algorithm. Summarizing, with this new dissimilarity metric, we can now define the federation formation process as follows:

1. Each node initially declares itself as a potential federation leader.
2. Leaders broadcast their candidacy through the network, propagating both their model parameters and the accumulated dissimilarity metric.
3. Nodes evaluate received candidacies, filtering out those where $G(n) > \sigma$.
4. When multiple valid candidacies remain, nodes apply a deterministic competition policy based on accumulated dissimilarity and node identifiers to ensure consistent leader selection.

5. Experimental evaluation

5.1. Learning setup

To evaluate the effectiveness of our approach, we performed learning on two well-known datasets in computer vision imported from the ProFed benchmark [62], namely: Extended MNIST [21] and CIFAR-100 [22] . Particularly, for the EMNIST dataset, we used the version with handwritten letters, which contains 124800 train samples and 20800 test samples from 26 classes (i.e., latin alphabet letters). Data have been synthetically partitioned to obtain multiple non-IID distributions and simulate labels skewness (i.e., a device has only a subset of the labels). More specifically, we conducted several simulations with a variable number of areas $|A| \in \{3, 5, 9\}$. We performed a synthetic partitioning of the data–as done in other works in the literature such as [10,63,64]–rather than using real-world non-IID datasets, to be able to control the degree of label skewness and to simulate various situations with different numbers of areas, enabling a more extensive evaluation.

The learning process was conducted using both the algorithm proposed by us and a set of baseline algorithms for comparison, namely: FedAvg, FedProx, Scaffold, and IFCA. The same hyperparameters were employed for all the approaches. The neural networks were trained for a total of 60 global rounds. For the EMNIST experiments, we employed a simple multilayer perceptron (MLP), whereas for the CIFAR-100 experiments we adopted the well-known MobileNet architecture [65]. At each global round, every device performed 2 epochs of local learning using a batch size of 64, the ADAM optimizer with a learning rate of 0.001, and a weight decay of 0.0001. Additionally, in the PSFL algorithm proposed, FedAVG was used to aggregate the models within the various federations.

PSFL was implemented using ScaFi [33]—a macroprogramming language for aggregate computing–combined with PyTorch [66] for the neural network training and the Alchemist simulator [67]. On the other hand, all the baselines were implemented in Python using only PyTorch. All the code is publicly available on GitHub [1].

For the baseline algorithms, we conducted 20 simulations with varying random seeds. Conversely, for the PSFL, we conducted 180 simulations varying:

(i) the random seed (from 0 to 19);
(ii) the number of areas (3, 5, and 9); and
(iii) the loss threshold σ (20, 40, and 80) used to determine the maximum expansion of a federation.

Moreover, we conducted 10 additional simulations with 9 areas in which we explored the effects of introducing a new node into an already stable system. The primary goal was to evaluate the system's ability to self-adapt without disrupting the existing federations. A special node, denoted as ★, was introduced to traverse all nine areas in sequence. The movement pattern of ★ includes systematically moving from one area to the next, covering the entire set of areas. Upon entering each new area, ★ remains there for a fixed period of 5 time steps to simulate interaction and impact on the area. The total duration of the simulation is 140 time steps.

Finally, we assessed the resilience of our self-organizing hierarchical architecture by simulating a failure scenario. In this experiment, we randomly killed two leaders after a certain period of time and evaluated whether the system could self-stabilize by re-electing new leader nodes to function as aggregator servers.

5.2. Metrics

Regarding the learning process, we have extrapolated the loss and the accuracy for each device. The loss used in this case is a negative log-likelihood loss which is defined as:

$$NLL(D, \mathcal{M}) = -\sum_{i=1}^{|D|} y_i \log(\mathcal{M}(x_i)), \quad \text{where } x_i, y_i \in D \tag{7}$$

where $\mathcal{M}(x_i)$ is the predicted probability of the true label y_i. The accuracy was calculated as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{8}$$

[1] anonymized for review

```
def proximityAwareSelfFederatedLearning(
    sigma: Double,
    initialModel: Model,
): Model = {
    // Maintain model state over time
    rep(initialModel) { prevModel =>
        // 0.1 Perform local training on the node's dataset
        val localModel = localTraining(localDataset, prevModel)
        // 0.2 Define dissimilarity metric based on models
        def metric(): Double =
            computeModelLossDissimilarity(
                localModel, nbr(localModel)
            )
        // Proximity-aware self-federated learning process
        // 1. Leader election based on model similarity
        val leaders = S(sigma, metric)
        // 1.1 Create the federation influence
        val region = G(leaders, metric)
        // 2 Collect models from the region
        // Leaders collect models from their region, ++ is union
        val collectedModels = C(
            region, _ ++ _, Set(localModel), Set.empty
        )
        // 3. Leaders aggregate the collected models
        val aggregatedModel = mux(leaders) {
            aggregateModelsFunction(collectedModels) // averaging
        } else {
            localModel // Non-leaders do not perform aggregation
        }
        // 4. Leaders broadcast the aggregated model back
        G(leaders, aggregatedModel, region)
    }
}
// Helper function to compute dissimilarity between two models
def computeModelDissimilarity(model1: Model, model2: Model): Double
```

Fig. 7. PSFL algorithm implementation (ScaFi-based) showing the main distributed learning functions, including decentralized leader election, model collection, aggregation, and dissemination. [61].

These values have been evaluated for both the training set and the validation set. During the testing phase, only the accuracy was verified since it is the metric of interest in our classification problem.

Regarding the federations, we have evaluated the number of federations created and their correctness. The federation count $|F|$, can be described as the unique number of leaders in the system at a given time step. Formally, given a set of leaders $L = \{l_1, l_2, \ldots, l_m\}$, the federation count is $|F| = |L|$. Federation correctness is evaluated by checking whether the leader and the devices belong to the

same area, meaning they share an IID data distribution. Mathematically, the correctness metric for a federation f_j is expressed as:

$$\diamondsuit(f_j) = \sum_{n \in f_j} \chi(\xi(n), \xi(l_j)) \tag{9}$$

where $\xi(n)$ denotes the actual area of node n, and $\xi(l_j)$ is the area of the federation leader l_j. The function χ outputs 0 when both areas are the same, indicating correct placement, and 1 when they differ. Therefore, $\diamondsuit(f_j)$ calculates the count of nodes that are mistakenly part of federation f_j. A value of 0 for $\diamondsuit(f_j)$ confirms that all nodes are correctly aligned, while any value greater than 0 signals misalignments within the federation. It is important to note that the actual area to which devices and leaders belong is not known to them; it is only used externally to the simulation to verify federations correctness.

Finally, in our experiment involving node movement, we evaluated changes in leadership that a moving node n_i perceives to understand the stability of the federations. At a given time t, this metric evaluates whether the leader of node n_i has changed compared to its leader at time $t-1$. Formally, this is defined as:

$$DL(n_i, t) = \chi(\zeta(n_i, t), \zeta(n_i, t-1)) \tag{10}$$

where $\zeta(n_i, t)$ identifies the leader of node n_i at time t. Therefore, this metric will result in 1 if the leader has changed and 0 otherwise.

5.3. Results

The results of our study were systematically collected during two distinct phases:

1. *training phase:* this phase involved monitoring the variations in model performance over time and assessing the accuracy of federations (see Fig. 8);
2. *testing phase:* stable federations were tested by introducing new test data to evaluate the overall effectiveness (Fig. 9).

5.3.1. Training phase

Fig. 8 reports the results of the experiments conducted on the EMNIST dataset. The training results on CIFAR-100 exhibit trends consistent with those observed on the other dataset. For the sake of conciseness, they are not included in the main text but are available in the accompanying code repository. The corresponding test results on CIFAR-100 are reported and discussed in this paper.

The charts in the first two rows illustrate a consistently decreasing trend in both training and validation loss. Periodic small increases are observable, which can be attributed to the aggregation of the new global model, occurring every t steps. A similar but opposite trend is evident in the validation accuracy, which shows a steady increase over time. Notably, this behavior persists regardless of the threshold selected for the formation of federations.

Addressing (RQ1), our results confirm that devices can be effectively clustered based on both spatial proximity and data distribution similarity through our decentralized approach. However, the federation formation process exhibits a clear threshold-dependent behavior For instance, in the case of a low threshold ($\sigma = 20$) and only three areas (each with a larger spatial extent), the system does not always converge to the correct number of federations, often stabilizing around five. However, the area correctness metric remains consistently at zero, indicating that all devices are grouped with others from the same area, although multiple federations may exist within a single area. Importantly, this does not lead to a degradation in final model accuracy. When using a medium threshold ($\sigma = 40$), we observe that after an initial transient phase, the system converges correctly in terms of both the number of federations and their area consistency. Finally, with a higher threshold ($\sigma = 80$) and a larger number of areas (thus each with smaller spatial extent), a reverse behavior emerges: for example, with nine areas, the system does not always converge to the correct number of federations and federations correctness is slightly higher than zero. This suggests that, in some cases, multiple distinct areas are incorrectly grouped into a single federation.

5.3.2. Testing phase ((RQ2))

Fig. 8 reports the results of the experiments conducted on the EMNIST dataset, while Table 3 presents the results on CIFAR-100, which exhibit similar trends. As expected, FedAvg, which lacks any mechanism to handle non-IID data, performs the worst across all scenarios. FedProx and Scaffold show slightly improved performance, as they include strategies to mitigate non-IID effects. However, since they are not clustered algorithms and still aim to produce a single global model, their effectiveness is limited in highly skewed settings such as those in our experiments. All three methods–FedAvg, FedProx, and Scaffold–exhibit a noticeable drop in accuracy as the number of areas (and thus data skewness) increases. In contrast, our proposed approach, PSFL, and IFCA demonstrate robust and comparable performance across different PSFL threshold settings, directly replying to (RQ2), therefore confirming the effectiveness of our approach in handling non-IID data distributions. It is important to highlight, however, that IFCA–as with most clustered algorithms in the literature–requires the number of clusters to be defined a priori. Fig. 9 shows the case where the number of clusters is correctly set equal to the number of areas (i.e., the number of underlying data distributions). In real-world scenarios, though, this hyperparameter is often difficult to estimate accurately, and an incorrect choice can significantly impact performance, as illustrated in Fig. 10, which reports IFCA performance under mismatched cluster counts.

5.3.3. Node movement (RQ3)

Chart analysis of node movement (Fig. 11) indicates that federations remain stable even as new nodes enter the system. After an initial period of instability, the system reorganizes, forming new federations as coherent as the previous ones. The behavior of

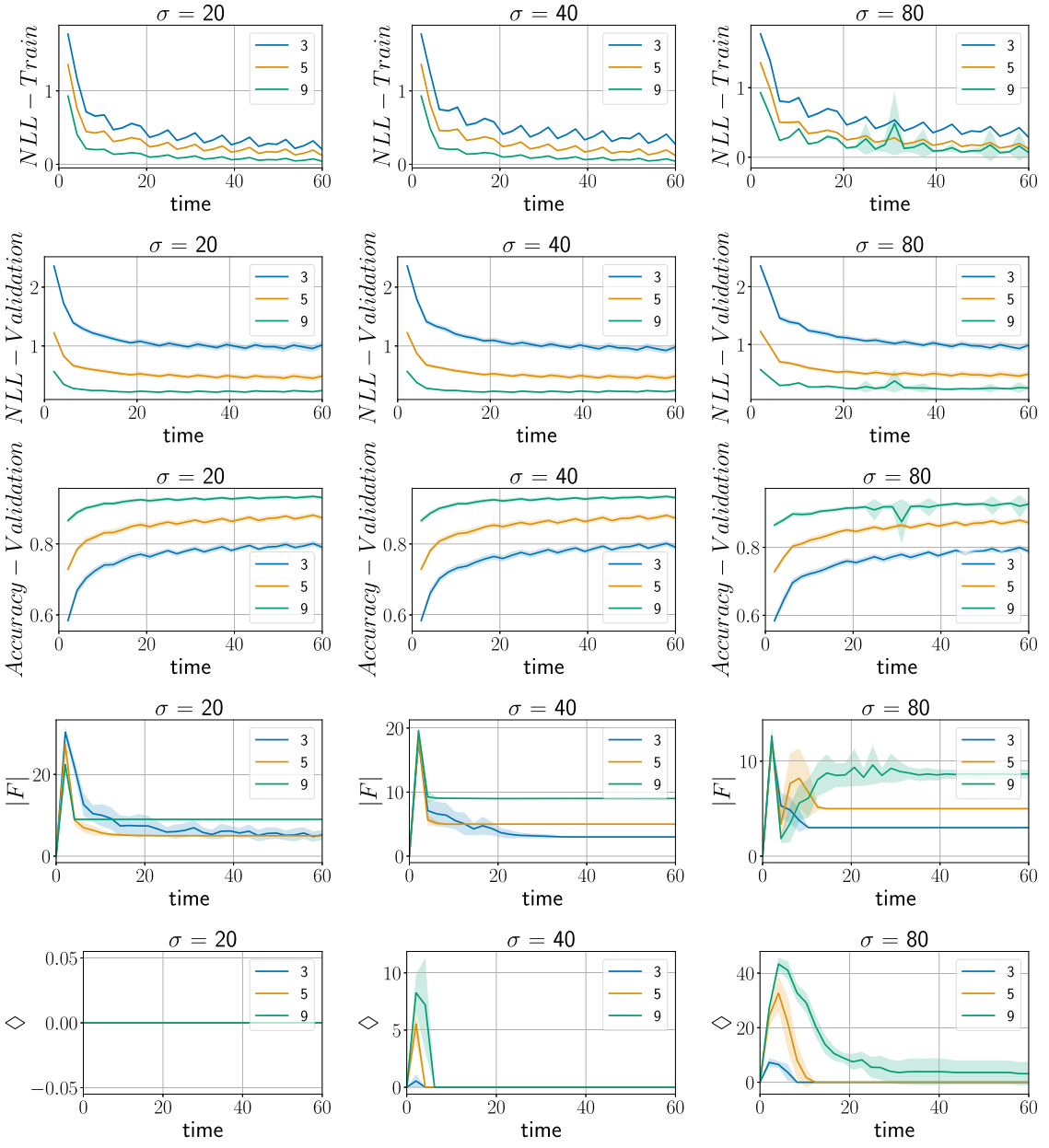


Fig. 8. Data collected during training and validation. Each column represents a different loss threshold, while rows represent metrics explained in Section 5.2. Lines color represent the number of areas.

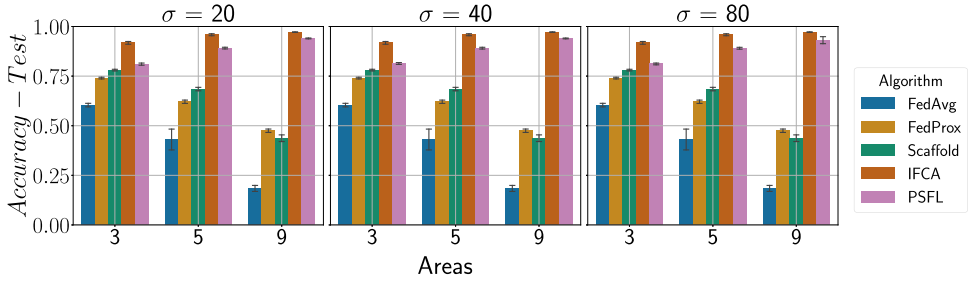


Fig. 9. Test accuracy of our PSFL approach compared against other approaches with different applied thresholds ($\phi = 20, 40$, and 80) and different number of areas. PSFL is closely trailing IFCA in precisely defined cluster areas across all thresholds.

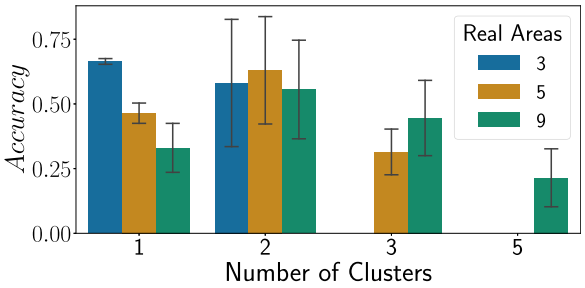


Fig. 10. Comparison of accuracy of IFCA in different settings with different number of areas against different predefined clusters showing the limited applicability if the clusters are not known *a priori*. Performance drops significantly while the variance increases when clusters are mismatched. In all cases of *Real Areas*, PSFL would outperform IFCA as it always achieves an accuracy between 0.8 and 0.95, depending on the selected σ as it does not require defined clusters *a priori*. IFCA results with correctly defined clusters as well as PSFL results are reported in Fig. 9.

Table 3
Test results on the CIFAR-100 dataset with mean threshold. The trend is similar to that discussed in Fig. 9 for the EMNIST dataset.

Algorithm	Subregions		
	3	5	9
FedAvg	0.45 ± 0.100	0.42 ± 0.010	0.35 ± 0.010
FedProx	0.51 ± 0.050	0.49 ± 0.030	0.41 ± 0.003
Scaffold	0.49 ± 0.010	0.47 ± 0.002	0.40 ± 0.002
IFCA	0.53 ± 0.003	0.54 ± 0.001	0.62 ± 0.020
PSFL	$\mathbf{0.56 \pm 0.010}$	$\mathbf{0.57 \pm 0.006}$	$\mathbf{0.69 \pm 0.002}$

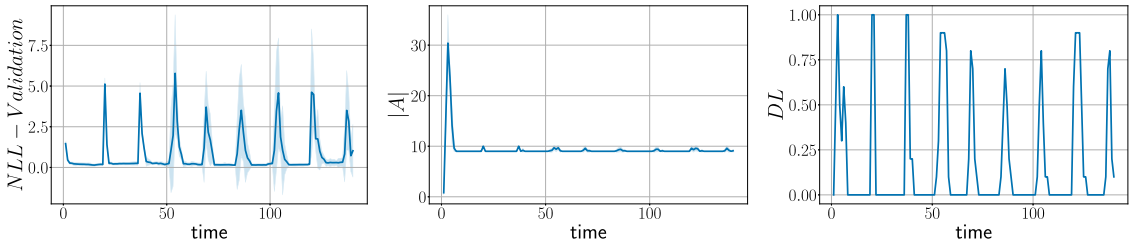


Fig. 11. Movement experiments with ★ node over time (average of 10 simulations, $\sigma = 20.0$). Despite transient phases, the system self-reorganizes maintaining stable federations.

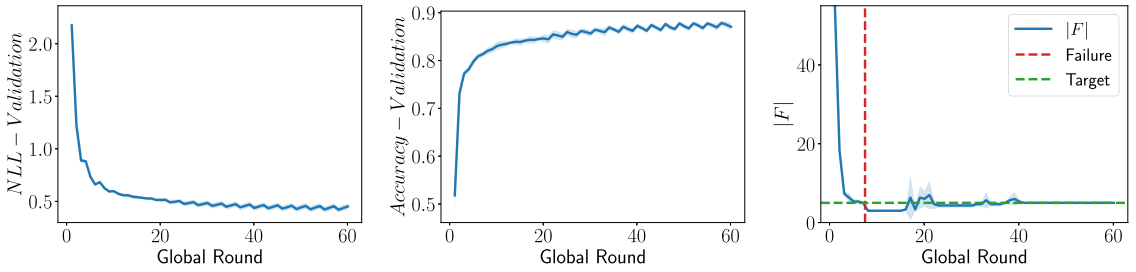
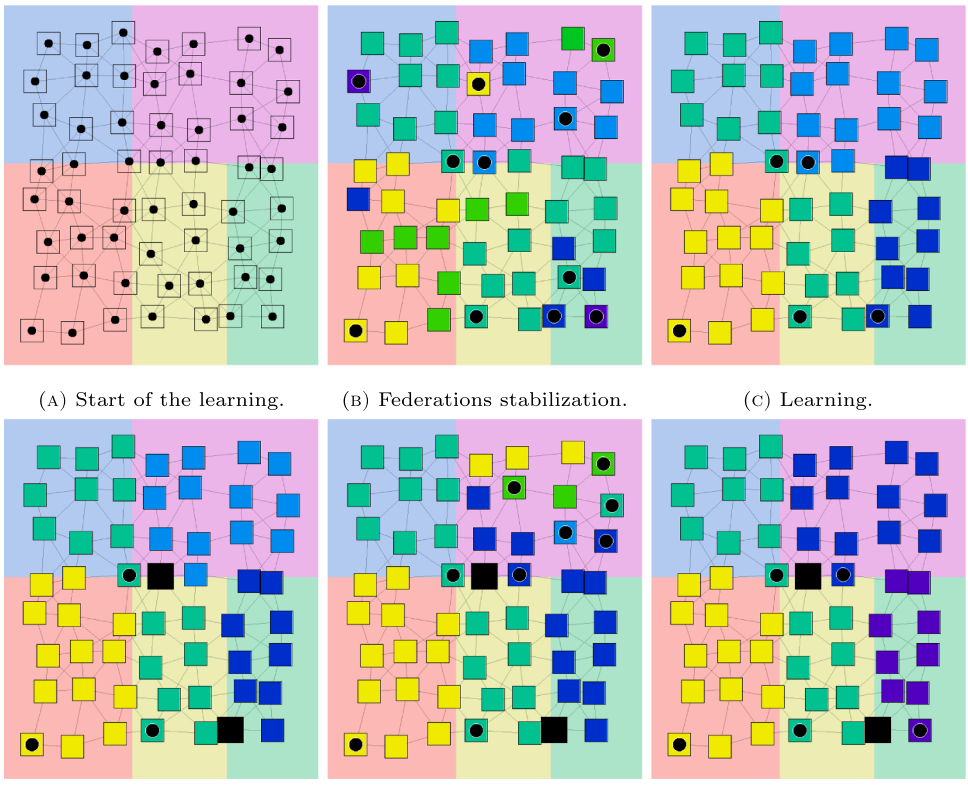


Fig. 12. Leader failure resilience evaluation. Despite two aggregator failures, learning continues with only a brief federation count perturbation at round 7.

the system is evident when observing two key metrics: DL and $NLL - Validation$. Immediately following the node's movement phase, the DL metric stabilizes near zero. Conversely, the $NLL - Validation$ initially peaks but subsequently stabilizes at a low value. Interestingly, when the node transitions to a new area, it attempts to establish a new federation but is prevented by a high error metric. Eventually, the node integrates into an existing federation, aiding in the stabilization of the system as the number of federations, denoted by $|F|$, returns to the optimal count of nine.



(A) Start of the learning. (B) Federations stabilization. (C) Learning.

(D) Aggregators failure. (E) Federations re-stabilization. (F) Learning.

Fig. 13. Alchemist simulation snapshots showing 56 nodes across 5 subregions. Background colors indicate data distributions, node colors show federations. Black dots mark active aggregators, black squares show failed ones. The system transitions from multiple aggregators per region to stable single-aggregator configurations.

5.3.4. Leaders failure (RQ3)

This last experiment has been designed to evaluate the resilience of the proposed self-organizing hierarchical architecture. We simulated a scenario involving 9 areas and a total of 56 devices. We ran 5 simulations with varying random seeds and a threshold of $\sigma = 40$. After allowing the system to stabilize under normal conditions (Fig. 13A to C), we introduced a disruption by randomly killing two aggregator devices within the network (Fig. 13D). The goal was to observe whether the system could autonomously recover, elect new aggregators, and continue the learning process without significant degradation in performance. The results, as shown in performance charts (Fig. 12) and visualized in the Alchemist simulator (Fig. 13), demonstrate that the PSFL architecture successfully re-stabilizes into a new configuration. Specifically, the performance charts show only a transitional perturbation in the number of federations during the failure phase (global round 7, indicated by the red dashed line), with no significant impact on the overall performance of the system. This trend can also be observed in the simulations (Fig. 13E). However, the system adapts by electing new aggregators, and the configuration stabilizes once more (Fig. 13E).

These findings highlight the robustness of the PSFL architecture, emphasizing its capability to recover from disruptions and maintain stable performance. This resilience is a critical feature for real-world applications where system stability under failure conditions is essential.

6. Limitations

While the proposed approach exhibits strong adaptability and resilience, some limitations highlight opportunities for future research and refinement.

6.1. Communication overhead and efficiency

PSFL inherently involves more frequent communication exchanges than traditional centralized approaches, as nodes must regularly share model updates with their neighbors. While the gradient computation and collect-cast operations enable self-organization, they also introduce additional message overhead that grows with network density. In bandwidth-limited or resource-constrained environments, this increased communication load can affect scalability and energy efficiency. Nonetheless, PSFL typically incurs

lower overhead than fully peer-to-peer schemes, where every client exchanges its model with all others. Future work will investigate communication-efficient variants, including techniques such as gradient quantization and sparse neural networks, as suggested in recent studies [68].

6.2. Heterogeneous device capabilities

The current approach assumes relatively homogeneous computational capabilities across devices. In real-world federated learning deployments, however, devices often differ significantly in processing power, memory availability, and battery life. The present leader election mechanism does not account for these variations, which may result in resource-constrained devices being selected as aggregators and, consequently, in reduced overall system performance. Future work could address this limitation by incorporating device capability information into the leader election process, ensuring that aggregators are chosen based on their ability to sustain the required computational workload.

7. Conclusions and future work

This article introduces Proximity-Aware Self-Federated Learning (PSFL), a novel self-organizing federated learning framework that leverages decentralized federation of models based on geographic proximity and data similarity. This approach, allowing a dynamic federation formation without a central coordinating device, addresses the challenges of applying federated learning in modern IoT systems, namely:

(i) bottlenecks and single point of failures caused by the centralized architecture used in common approaches; and
(ii) data heterogeneity (i.e., non-IID data) common in IoT scenarios.

The results show that PSFL successfully addresses our research questions. Regarding (RQ1), we demonstrated that devices can indeed be clustered based on both spatial proximity and data distribution similarity in a fully decentralized manner through our self-federating mechanism. For (RQ2), PSFL consistently outperforms the three baseline algorithms that do not utilize clustering strategies–namely, FedAvg, FedProx, and Scaffold–across a variety of scenarios, confirming that our clustering strategy enhances model accuracy with non-IID data. This performance gap becomes increasingly evident as the number of areas increases. In comparison to the IFCA baseline, PSFL achieves similar performance when the number of clusters used for IFCA training exactly matches the number of areas. However, it significantly outperforms IFCA when this condition is not met. Addressing RQ3, our experiments with leader failures and node movements demonstrate that the self-organizing architecture significantly improves system robustness and fault tolerance. These findings highlight the effectiveness of PSFL in enhancing federated learning, particularly in real-world settings characterized by inherently non-IID data distributions, high environmental variability, and the absence of prior knowledge about aggregator devices.

Future research may expand the evaluation of our approach to other well-known datasets, employing new partitioning methodologies to simulate diverse distributions and further increasing the number of areas and devices to encompass scenarios of varying complexities. Moreover, collecting data from a real-world use case would provide a valuable opportunity to evaluate our system in more realistic and challenging conditions.

Additionally, it would be interesting to evaluate the system's performance when multiple nodes move simultaneously. This investigation could determine whether such movement is beneficial, perhaps due to cross-fertilization among federations, or detrimental due to the instability it causes in federations. Finally, building on this point, and considering the dynamic nature of the systems we are studying, another area for future work could involve integrating continuous learning techniques to adapt models to data that evolves over time.

CRediT authorship contribution statement

Davide Domini: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Data curation, Conceptualization; **Nicolas Farabegoli:** Writing – review & editing, Writing – original draft, Validation, Software, Data curation; **Gianluca Aguzzi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Data curation, Conceptualization; **Mirko Viroli:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization; **Lukas Esterle:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Conceptualization.

Data availability

The code and the data are available at https://github.com/davidedomini/experiments-2025-iot-self-federated-learning

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Lukas Esterle was supported by the Independent Research Fund Denmark through the FLOCKD project under the grant number 1032-00179B. Gianluca Aguzzi and Nicolas Farabegoli were supported by the Italian PRIN project "CommonWears" (2020 HCWWLP). Finally, this work was also supported by FAIR-Future Artificial Intelligence Research" project (PNRR, M4C2, Investimento 1.3, Partenariato Esteso PE00000013), funded by the European Commission under the NextGenerationEU programme.

References

[1] P. Lévy, Collective Intelligence: Mankind's Emerging World in Cyberspace, Perseus books, 1997.
[2] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance), 2016,
[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A.y. Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 54 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a.html.
[4] J. Xu, B.S. Glicksberg, C. Su, P.B. Walker, J. Bian, F. Wang, Federated learning for healthcare informatics, J. Heal. Informat. Res. 5 (1) (2021) 1–19. https://doi.org/10.1007/S41666-020-00082-4
[5] D.C. Nguyen, Q. Pham, P.N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O.A. Dobre, W. Hwang, Federated learning for smart healthcare: a survey, ACM Comput. Surv. 55 (3) (2023) 60:1–60:37. https://doi.org/10.1145/3501296
[6] A. Rahman, M.S. Hossain, G. Muhammad, D. Kundu, T. Debnath, M.S. Rahman, M.S.I. Khan, P. Tiwari, S.S. Band, Federated learning-based AI approaches in smart healthcare: concepts, taxonomies, challenges and open issues, Clust. Comput. 26 (4) (2023) 2271–2311.
[7] D. Kundu, M.M. Rahman, A. Rahman, D. Das, U.R. Siddiqi, M.G.R. Alam, S.K. Dey, G. Muhammad, Z. Ali, Federated deep learning for Monkeypox disease detection on GAN-augmented dataset, IEEE Access 12 (2024) 32819–32829.
[8] M.S.I. Khan, A. Rahman, T. Debnath, M.R. Karim, M.K. Nasir, S.S. Band, A. Mosavi, I. Dehzangi, Accurate brain tumor detection using deep convolutional neural network, Comput. Struct. Biotechnol. J. 20 (2022) 4733-4745. https://doi.org/10.1016/j.csbj.2022.08.039
[9] G. Aguzzi, R. Casadei, D. Pianini, M. Viroli, Dynamic decentralization domains for the internet of things, IEEE Internet Comput. 26 (6) (2022) 16–23. https://doi.org/10.1109/MIC.2022.3216753
[10] Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-IID data silos: an experimental study, in: Proceedings of the International Conference on Data Engineering, IEEE, 2022, pp. 965–978. https://doi.org/10.1109/ICDE53745.2022.00077
[11] E. Dritsas, M. Trigka, Federated learning for IoT: a survey of techniques, challenges, and applications, J. Sens. Actuat. Netw. 14 (1) (2025) 9. https://doi.org/10.3390/JSAN14010009
[12] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for Internet of Things: a comprehensive survey, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1622–1658. https://doi.org/10.1109/COMST.2021.3075439
[13] A. Rahman, K. Hasan, D. Kundu, M.J. Islam, T. Debnath, S.S. Band, N. Kumar, On the ICN-IoT with federated learning integration of communication: concepts, security-privacy issues, applications, and future perspectives, Future Gener. Comput. Syst. 138 (2023) 61–88.
[14] L. Esterle, Deep learning in multiagent systems, in: Deep Learning for Robot Perception and Cognition, Elsevier, 2022, pp. 435–460.
[15] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, et al., Advances and open problems in federated learning, Found. Trend. Mach. Learn. 14 (1–2) (2021) 1–210. https://doi.org/10.1561/2200000083
[16] X. Ma, J. Zhu, Z. Lin, S. Chen, Y. Qin, A state-of-the-art survey on solving non-IID data in federated learning, Future Gener. Comput. Syst. 135 (2022) 244–258. https://doi.org/10.1016/J.FUTURE.2022.05.003
[17] A.Z. Tan, H. Yu, L. Cui, Q. Yang, Towards personalized federated learning, IEEE Trans. Neural Networks Learn. Syst. 34 (12) (2023) 9587–9603. https://doi.org/10.1109/TNNLS.2022.3160699
[18] J. Beal, D. Pianini, M. Viroli, Aggregate programming for the Internet of Things, Computer (Long Beach Calif) 48 (9) (2015) 22–30. https://doi.org/10.1109/MC.2015.261
[19] R. Casadei, D. Pianini, M. Viroli, A. Natali, Self-organising coordination regions: a pattern for edge computing, in: Proceedings of the International Conference on Coordination Models and Languages, 11533 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 182–199. https://doi.org/10.1007/978-3-030-22397-7_11
[20] R. Casadei, S. Mariani, D. Pianini, M. Viroli, F. Zambonelli, Space-fluid adaptive sampling by self-organisation, Log. Method. Comput. Sci. 19 (4) (2023). https://doi.org/10.46298/LMCS-19(4:29)2023
[21] G. Cohen, S. Afshar, J. Tapson, A.v. Schaik, EMNIST: Extending MNIST to handwritten letters, in: 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14–19, 2017, IEEE, 2017, pp. 2921–2926. https://doi.org/10.1109/IJCNN.2017.7966217
[22] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).
[23] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, in: Proceedings of Machine Learning and Systems, mlsys.org, 2020. https://proceedings.mlsys.org/book/316.pdf.
[24] S.P. Karimireddy, S. Kale, M. Mohri, S.J. Reddi, S.U. Stich, A.T. Suresh, SCAFFOLD: Stochastic controlled averaging for federated learning, in: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event, 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 5132–5143. http://proceedings.mlr.press/v119/karimireddy20a.html.
[25] A. Ghosh, J. Chung, D. Yin, K. Ramchandran, An efficient framework for clustered federated learning, IEEE Trans. Inf. Theory 68 (12) (2022) 8076–8091. https://doi.org/10.1109/TIT.2022.3192506
[26] D. Domini, G. Aguzzi, N. Farabegoli, M. Viroli, L. Esterle, Proximity-based self-federated learning, in: IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2024, Aarhus, Denmark, September 16–20, 2024, IEEE, 2024, pp. 139–144. https://doi.org/10.1109/ACSOS61780.2024.00033
[27] A. Taïk, S. Cherkaoui, Federated edge learning: design issues and challenges, IEEE Netw. 35 (2) (2021) 252–258. https://doi.org/10.1109/MNET.011.2000478
[28] C. Prigent, A. Costan, G. Antoniu, L. Cudennec, Enabling federated learning across the computing continuum: systems, challenges and future directions, Future Gener. Comput. Syst. 160 (2024) 767–783. https://doi.org/10.1016/J.FUTURE.2024.06.043
[29] A. Fersch, Collective adaptive systems, in: Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers, 2015, pp. 893–895.
[30] G. Aguzzi, R. Casadei, M. Viroli, Macroswarm: a field-based compositional framework for swarm programming, CoRR abs/2401.10969 (2024). 2401.10969https://doi.org/10.48550/ARXIV.2401.10969
[31] R. Casadei, M. Viroli, G. Audrito, D. Pianini, F. Damiani, Engineering collective intelligence at the edge with aggregate processes, Eng. Appl. Artif. Intell. 97 (2021) 104081. https://doi.org/10.1016/J.ENGAPPAI.2020.104081
[32] M. Viroli, J. Beal, F. Damiani, G. Audrito, R. Casadei, D. Pianini, From field-based coordination to aggregate computing, in: G.D.M. Serugendo, M. Loreti (Eds.), Coordination Models and Languages - 20th IFIP WG 6.1 International Conference, COORDINATION 2018, Held as Part of the 13th International Federated Conference on Distributed Computing Techniques, DisCoTec 2018, Madrid, Spain, June 18–21, 2018. Proceedings, 10852 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 252–279. https://doi.org/10.1007/978-3-319-92408-3_12
[33] R. Casadei, M. Viroli, G. Aguzzi, D. Pianini, ScaFi: a scala DSL and toolkit for aggregate programming, SoftwareX 20 (2022) 101248. https://doi.org/10.1016/J.SOFTX.2022.101248
[34] R. Casadei, Macroprogramming: concepts, state of the art, and opportunities of macroscopic behaviour modelling, ACM Comput. Surv. 55 (13s) (2023) 275:1–275:37. https://doi.org/10.1145/3579353

[35] M. Viroli, G. Audrito, J. Beal, F. Damiani, D. Pianini, Engineering resilient collective adaptive systems by self-stabilisation, ACM Trans. Model. Comput. Simul. 28 (2) (2018) 16:1–16:28. https://doi.org/10.1145/3177774

[36] J. Beal, M. Viroli, D. Pianini, F. Damiani, Self-adaptation to device distribution in the Internet of Things, ACM Trans. Auton. Adapt. Syst. 12 (3) (2017) 12:1–12:29. https://doi.org/10.1145/3105758

[37] G. Audrito, J. Beal, F. Damiani, M. Viroli, Space-time universality of field calculus, in: G.D.M. Serugendo, M. Loreti (Eds.), Coordination Models and Languages - 20th IFIP WG 6.1 International Conference, COORDINATION 2018, Held as Part of the 13th International Federated Conference on Distributed Computing Techniques, DisCoTec 2018, Madrid, Spain, June 18–21, 2018. Proceedings, 10852 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 1–20. https://doi.org/10.1007/978-3-319-92408-3_1

[38] G. Aguzzi, R. Casadei, M. Viroli, Machine learning for aggregate computing: a research roadmap, in: 42nd IEEE International Conference on Distributed Computing Systems, ICDCS Workshops, Bologna, Italy, July 10, 2022, IEEE, 2022, pp. 119–124. https://doi.org/10.1109/ICDCSW56584.2022.00032

[39] W. Jaradat, A. Dearle, A. Barker, Towards an autonomous decentralized orchestration system, Concurr. Comput. Pract. Exp. 28 (11) (2016) 3164–3179. https://doi.org/10.1002/CPE.3655

[40] M. Díaz, B. Rubio, J.M. Troya, A coordination middleware for wireless sensor networks, in: Systems Communications 2005 (ICW / ICHSN / ICMCS / SENET 2005), 14–17 August 2005, Montreal, Canada, IEEE Computer Society, 2005, pp. 377–382. https://doi.org/10.1109/ICW.2005.5

[41] G. Audrito, R. Casadei, F. Damiani, M. Viroli, Compositional blocks for optimal self-healing gradients, in: 11th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2017, Tucson, AZ, USA, September 18–22, 2017, IEEE Computer Society, 2017, pp. 91–100. https://doi.org/10.1109/SASO.2017.18

[42] G. Long, Y. Tan, J. Jiang, C. Zhang, Federated learning for open banking, in: Q. Yang, L. Fan, H. Yu (Eds.), Federated Learning - Privacy and Incentive, 12500 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 240–254. https://doi.org/10.1007/978-3-030-63076-8_17

[43] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl. Base. Syst. 216 (2021) 106775. https://doi.org/10.1016/J.KNOSYS.2021.106775

[44] M. Adam, A. Albaseer, U. Baroudi, M. Abdallah, Survey of multimodal federated learning: exploring data integration, challenges, and future directions, IEEE Open J. Commun. Soc. 6 (2025) 2510–2538.

[45] I. Hegedüs, G. Danner, M. Jelasity, Decentralized learning works: an empirical comparison of gossip learning and federated learning, J. Parallel Distribut. Comput. 148 (2021) 109–124. https://doi.org/10.1016/J.JPDC.2020.10.006

[46] T. Wink, Z. Nochta, An approach for peer-to-peer federated learning, in: Proceedings of the International Conference on Dependable Systems and Networks Workshops, IEEE, 2021, pp. 150–157. https://doi.org/10.1109/DSN-W52860.2021.00034

[47] S. Wang, S. Hosseinalipour, M. Gorlatova, C.G. Brinton, M. Chiang, UAV-assisted online machine learning over multi-tiered networks: a hierarchical nested personalized federated learning approach, IEEE Trans. Netw. Serv. Manag. 20 (2) (2023) 1847–1865. https://doi.org/10.1109/TNSM.2022.3216326

[48] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, M. Jirstrand, A performance evaluation of federated learning algorithms, in: Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning, DIDL@Middleware 2018, Rennes, France, December 10, 2018, ACM, 2018, pp. 1–8. https://doi.org/10.1145/3286490.3286559

[49] H. Zhu, J. Xu, S. Liu, Y. Jin, Federated learning on non-IID data: a survey, Neurocomputing 465 (2021) 371–390. https://doi.org/10.1016/J.NEUCOM.2021.07.098

[50] , (https://www.electricitymaps.com). Accessed: 2025-4-23.

[51] S.P. Karimireddy, S. Kale, M. Mohri, S.J. Reddi, S.U. Stich, A.T. Suresh, SCAFFOLD: Stochastic controlled averaging for on-device federated learning, CoRR abs/1910.06378 (2019). 1910.06378.

[52] J. Wang, Q. Liu, H. Liang, G. Joshi, H.V. Poor, Tackling the objective inconsistency problem in heterogeneous federated optimization, in: Proceedings of the Annual Conference on Neural Information Processing Systems, 2020. https://proceedings.neurips.cc/paper/2020/hash/564127c03caab942e503ee6f810f54fd-Abstract.html.

[53] F. Sattler, K. Müller, W. Samek, Clustered federated learning: model-agnostic distributed multi-task optimization under privacy constraints, CoRR abs/1910.01991 (2019). 1910.01991.

[54] D. Domini, G. Aguzzi, L. Esterle, M. Viroli, FBFL: A field-based coordination approach for data heterogeneity in federated learning, CoRR abs/2502.08577 (2025). https://doi.org/10.48550/ARXIV.2502.08577

[55] Z. Li, J. Lu, S. Luo, D. Zhu, Y. Shao, Y. Li, Z. Zhang, Y. Wang, C. Wu, Towards effective clustered federated learning: a peer-to-peer framework with adaptive neighbor matching, IEEE Trans. Big Data (2022) 1–16. https://doi.org/10.1109/TBDATA.2022.3222971

[56] M. Duan, D. Liu, X. Ji, Y. Wu, L. Liang, X. Chen, Y. Tan, A. Ren, Flexible clustered federated learning for client-level data distribution shift, IEEE Trans. Parall. Distribut. Syst. 33 (11) (2022) 2661–2674. https://doi.org/10.1109/TPDS.2021.3134263

[57] M.N.H. Nguyen, S.R. Pandey, N.D. Tri, E. Huh, N.H. Tran, W. Saad, C.S. Hong, Self-organizing democratized learning: toward large-scale distributed learning systems, IEEE Trans. Neural Netw. Learn. Syst. 34 (12) (2023) 10698–10710. https://doi.org/10.1109/TNNLS.2022.3170872

[58] X. Li, X. Chen, B. Tang, S. Wang, Y. Xuan, Z. Zhao, Unsupervised graph structure-assisted personalized federated learning, in: K. Gal, A. Nowé, G.J. Nalepa, R. Fairstein, R. Radulescu (Eds.), Proceedings of the European Conference on Artificial Intelligence, 372 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 1430–1438. https://doi.org/10.3233/FAIA230421

[59] Y. Yan, X. Tong, S. Wang, Clustered federated learning in heterogeneous environment, IEEE Trans. Neural Netw. Learn. Syst. 35 (9) (2024) 12796–12809. https://doi.org/10.1109/TNNLS.2023.3264740

[60] M. Mehta, C. Shao, A greedy agglomerative framework for clustered federated learning, IEEE Trans. Ind. Informat. 19 (12) (2023) 11856–11867. https://doi.org/10.1109/TII.2023.3252599

[61] D. Davide, A. Gianluca, F. Nicolas, V. Mirko, E. Lukas, IoT Experiments PSFL, 2025, https://github.com/davidedomini/experiments-2025-iot-self-federated-learning.

[62] D. Domini, G. Aguzzi, M. Viroli, ProFed: a benchmark for proximity-based non-IID federated learning, CoRR abs/2503.20618 (2025). https://doi.org/10.48550/ARXIV.2503.20618

[63] M.F. Elvebakken, A. Iosifidis, L. Esterle, Adaptive parameterization of deep learning models for federated learning, CoRR abs/2302.02949 (2023). https://doi.org/10.48550/ARXIV.2302.02949

[64] D. Domini, G. Aguzzi, L. Esterle, M. Viroli, Field-based coordination for federated learning, in: I. Castellani, F. Tiezzi (Eds.), Coordination Models and Languages - 26th IFIP WG 6.1 International Conference, COORDINATION 2024, Held as Part of the 19th International Federated Conference on Distributed Computing Techniques, DisCoTec 2024, Groningen, the Netherlands, June 17–21, 2024, Proceedings, 14676 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 56–74. https://doi.org/10.1007/978-3-031-62697-5_4

[65] M. Sandler, A.G. Howard, M. Zhu, A. Zhmoginov, L. Chen, MobileNetV2: inverted residuals and linear bottlenecks, in: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018, Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4510–4520. http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html. https://doi.org/10.1109/CVPR.2018.00474

[66] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).

[67] D. Pianini, S. Montagna, M. Viroli, Chemical-oriented simulation of computational systems with ALCHEMIST, J. Simulat. 7 (3) (2013) 202-215. https://doi.org/10.1057/jos.2012.27

[68] D. Domini, L. Erhan, G. Aguzzi, L. Cavallaro, A.D. Zenoozi, A. Liotta, M. Viroli, Sparse self-federated learning for energy efficient cooperative intelligence in society 5.0, CoRR abs/2507.07613 (2025). https://doi.org/10.48550/ARXIV.2507.07613