Logical Methods in Computer Science
Volume 22, Issue 1, 2026, pp. 19:1–19:30
https://lmcs.episciences.org/

Submitted Feb. 14, 2025
Published Mar. 06, 2026

FBFL: A FIELD-BASED COORDINATION APPROACH FOR DATA HETEROGENEITY IN FEDERATED LEARNING

DAVIDE DOMINI [a], GIANLUCA AGUZZI [a], LUKAS ESTERLE [b], AND MIRKO VIROLI [a]

[a] Alma Mater Studiorum—Università di Bologna, Via dell'Università, 50, Cesena (FC), Italy
e-mail address: davide.domini@unibo.it, gianluca.aguzzi@unibo.it, mirko.viroli@unibo.it

[b] Aarhus University, Aarhus, Denmark
e-mail address: lukas.esterle@ece.au.dk

Abstract. In the last years, Federated learning (FL) has become a popular solution to train machine learning models in domains with high privacy concerns. However, FL scalability and performance face significant challenges in real-world deployments where data across devices are non-independently and identically distributed (non-IID). The heterogeneity in data distribution frequently arises from spatial distribution of devices, leading to degraded model performance in the absence of proper handling. Additionally, FL typical reliance on centralized architectures introduces bottlenecks and single-point-of-failure risks, particularly problematic at scale or in dynamic environments.

To close this gap, we propose Field-Based Federated Learning (FBFL), a novel approach leveraging macroprogramming and field coordination to address these limitations through: (i) distributed spatial-based leader election for personalization to mitigate non-IID data challenges; and (ii) construction of a self-organizing, hierarchical architecture using advanced macroprogramming patterns. Moreover, FBFL not only overcomes the aforementioned limitations, but also enables the development of more specialized models tailored to the specific data distribution in each subregion.

This paper formalizes FBFL and evaluates it extensively using MNIST, FashionMNIST, and Extended MNIST datasets. We demonstrate that, when operating under IID data conditions, FBFL performs comparably to the widely-used FedAvg algorithm. Furthermore, in challenging non-IID scenarios, FBFL not only outperforms FedAvg but also surpasses other state-of-the-art methods, namely FedProx and Scaffold, which have been specifically designed to address non-IID data distributions. Additionally, we showcase the resilience of FBFL's self-organizing hierarchical architecture against server failures.

1. Introduction

Research Context. Machine learning requires large datasets to train effective and accurate models. Typically, data are gathered from various sources into one central server where the training process occurs. However, centralizing data on a single device poses significant privacy challenges. These concerns arise not only from the need to share sensitive information but also from the heightened risk of storing all data in a single location, which, if compromised, could result in large-scale data exposure. Federated learning (FL) has emerged as a promising

Key words and phrases: Personalization, Federated Learning, Aggregate Computing, Field-based Coordination.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.46298/LMCS-22(1:19)2026

solution for training machine learning models in scenarios where *data privacy* is a primary concern, enabling devices to collaboratively learn a shared model while keeping their data local [MMR+17]. This paradigm not only alleviates the necessity for central data storage but also addresses privacy and efficiency concerns inherent in traditional systems with centralized learning.

Research Gap. Despite its advantages, in the current landscape of FL, training is distributed, but model construction is still predominantly centralized, posing challenges in scenarios characterized by spatial dispersion of devices, heightened risk of single points of failure, and naturally distributed datasets. Existing peer-to-peer solutions attempt to tackle these concerns; however, they often overlook the spatial distribution of devices and do not exploit the benefits of semantically similar knowledge among nearby devices [MDAV25]. This builds on the assumption that devices in spatial proximity have similar experiences and make similar observations, as the phenomena to capture are intrinsically context dependent [Est22]. Moreover, existing approaches often lack the flexibility to seamlessly transition between fully centralized and fully decentralized aggregation methods. This limitation highlights the potential role of advanced coordination models and programming languages in bridging this gap.

Contribution. For these reasons, in this paper, we introduce Field-Based Federated Learning (FBFL), a novel approach that leverages computational fields (*i.e.,* global maps from devices to values) as key abstraction to facilitate device coordination [VBD+18, LLM17] in FL. By field-based coordination, global-level system behavior can be captured declaratively, with automatic translation into single-device local behavior.

We find that this approach offers a versatile and scalable solution to FL, supporting the formation of *personalized model zones*—spatially-based regions where devices with similar data distributions collaboratively train specialized models tailored to their local context, rather than a single global model shared across all participants. This represents a form of *personalized federated learning*, where instead of learning one universal model that may perform poorly on diverse local data distributions, the system creates multiple specialized models, each optimized for the specific characteristics of data within its respective zone. Most specifically, our approach actually relies on known field-based algorithms of information diffusion and aggregation developed in the context of aggregate computing [VAB+18], ultimately defining what we can define "fields of Machine Learning (sub)models". This method enables dynamic, efficient model aggregation without a centralized authority, thereby addressing the limitations of current FL frameworks. Therefore, our contributions are twofold:

- We demonstrate that field coordination enables performance comparable to centralized approaches under independent and identically distributed (IID) data settings;
- We exploit advanced aggregate computing patterns to efficiently create a self-organizing hierarchical architecture that group devices based on spatial proximity, improving performance under non-IID data settings.

By "self-organizing hierarchical" we mean a hybrid architecture based on peer-to-peer interactions, but which elects leaders in a distributed manner. These leaders will act as aggregators of local models pertaining to sub-portions of the system. We evaluate our approach in a simulated environment, where well-known computer vision datasets–MNIST [LCB+10], FashionMNIST [XRV17], and Extended MNIST [CATvS17]–are synthetically split to create

non-IID partitions. As baselines, we employ three state-of-the-art algorithms, namely: FedAvg [MMR+17], FedProx [LSZ+20], and Scaffold [KKM+20]. Our findings indicate that this field-based strategy not only matches the performance of existing methods but also provides enhanced scalability and flexibility in FL implementations.

Paper Structure. This manuscript is an extended version of the conference paper [DAEV24], providing: (i) a more extensive and detailed coverage of related work; (ii) an expanded formalization which adds the description of the self-organizing hierarchical architecture proposed by this paper; (iii) a largely extended experimental evaluation adding more datasets (*i.e.,* MNIST, Fashion MNIST and Extended MNIST) and more baselines (*i.e.,* FedAvg, FedProx and Scaffold); and (iv) 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 states the research questions. Section 3 reviews the necessary background on federated learning and field-based coordination. Section 4 presents related works, the baseline algorithms, and our motivation. Section 5 introduces our Field-Based Federated Learning approach: we formalize the problem (Section 5.1), describe the distributed algorithm (Section 5.2), and provide implementation details (Section 5.3). Section 6 reports the experimental evaluation and discussion. Section 7 discusses limitations. Finally, Section 8 concludes and outlines future work.

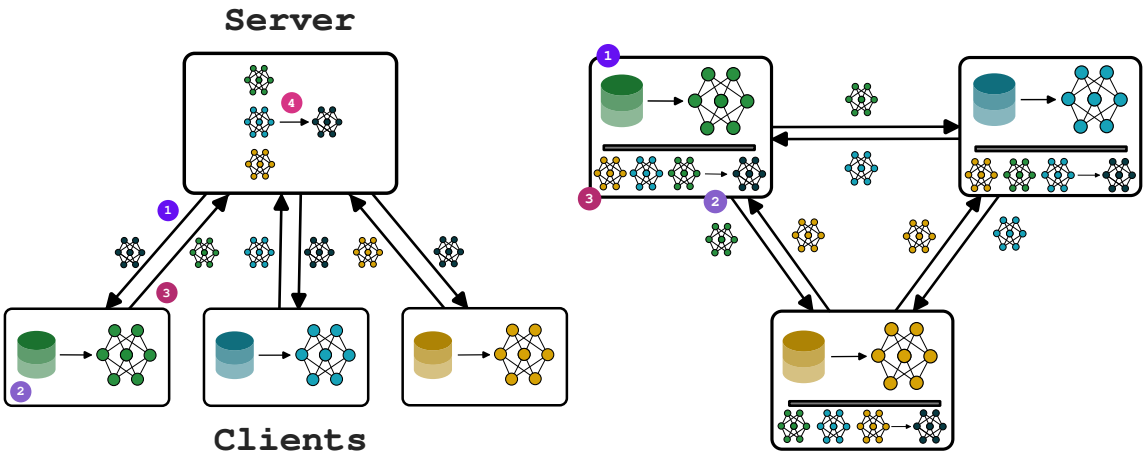
2. Research Questions

The complexity of modern systems where federated learning (FL) can be applied, such as the edge-cloud compute continuum, is rapidly increasing. This poses significant challenges in terms of scalability, adaptability, and contextual relevance [PCAC24]. Field-based approaches have demonstrated notable advantages in addressing such complexity in various domains, offering robust solutions in both machine learning [ACV22, DCAV24], software engineering [CPCC24] contexts and real world use cases [ABB+25]. However, research on field-based methodologies within FL, particularly in the area of personalized FL [DAF+24, Dom24, DFA+26], is still at an early stage. In particular, current approaches still suffer from three key limitations: reliance on centralized aggregation (limiting scalability and resilience), difficulty handling highly non-IID data, and lack of mechanisms to exploit spatial correlation in data distributions. Our proposed FBFL framework directly targets these limitations. To assess its effectiveness, we focus on the following research questions:

(RQ1) How does the Field-Based Federated Learning approach perform compared to centralized FL under IID data?

(RQ2) Can we increase accuracy by introducing personalized learning zones where *learning devices* are grouped based on similar experiences as often observed in spatially near locations? More precisely, what impact does this have on heterogeneous and non-IID data distributions?

(RQ3) What is the effect of creating a self-organizing hierarchical architecture through field coordination on federated learning in terms of resilience, adaptability and fault-tolerance?



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

FIGURE 1. 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. Background

3.1. **Federated learning.** Federated learning (FL) [MMR+17] is a machine learning technique introduced to collaboratively train a joint model using multiple, potentially *spatially distributed*, devices. In general, in the context of FL, a "model" refers to any machine learning model—such as decision trees, support vector machines, or neural networks—that can be trained on data to perform increasingly accurate predictions. These are statistical models whose parameters are iteratively updated based on local data to minimize prediction error (*i.e.,* the loss function). In this work, we focus specifically on neural networks as the model class of interest, given their widespread adoption and strong empirical performance in a variety of learning tasks.

The federation typically consists of multiple *client* devices and one *aggregation* server, which may be located in the cloud. The key idea behind FL is that the data always remains on the device to which it belongs: devices only share the weights of their models (*i.e.,* the parameters of the neural network), thus making FL an excellent solution in contexts where *privacy* is a crucial aspect (*e.g.,* in health-care applications [XGS+21, NPP+23]) or where data are *naturally distributed*, and their volume makes it infeasible to transfer them from each client device to a centralized server. Federated learning can be performed in different ways [YLCT19]. In horizontal FL (a.k.a., sample-based), participants share the same input features but hold distinct sample sets. They generally follow four steps (see Figure 1A):

(1) *model distribution*: the server distributes the initial global model to each device. This step ensures that all devices start with the same model parameters before local training begins;
(2) *local training*: each device trains the received model on its local dataset for a specified number of epochs. This step allows the model to learn from the local data while keeping the data on the device, thus preserving privacy;
(3) *model sharing*: after local training, each device sends its updated model parameters back to the server. This step involves communication overhead but is crucial for aggregating the learned knowledge from all devices;
(4) *model aggregation*: the server collects all the updated model parameters from the devices and combines them using an aggregation algorithm (*e.g.,* averaging the weights). This step produces a new global model that incorporates the knowledge learned by all devices. The process then repeats for a predefined number of rounds or until convergence.

Despite its advantages, FL faces several challenges, such as *non-IID* data distribution and *resilience* to server failures, discussed in the following section.

3.1.1. *Challenges.*

Resilience. Exploring the network topology is crucial, as the configuration of device communication can significantly influence FL performance. Various structures, such as *peer-to-peer* (P2P), *hierarchical*, and *centralized* networks, offer diverse benefits and challenges. Traditionally, FL systems have relied on a centralized server for model aggregation. However, this setup poses scalability challenges and risks introducing a single point of failure. In response, recent advancements (*e.g.,* [HDJ21, WN21, LZSL20]) have embraced P2P networks to address these limitations, going towards what is called decentralized federated learning—see Figure 1B. P2P architectures, devoid of a central authority, enhance scalability and robustness through the utilization of gossip [VBT17] or consensus algorithms [SNR20] for model aggregation.

Data heterogeneity. Federated learning is particularly effective when the data distribution across devices is *independent and identically distributed* (IID) [NSU+18], namely the users experience the same data distribution. For instance, in a text prediction application, all users may have similar writing styles. However, in real-world scenarios, data distribution is often non-IID, where devices have different data distributions. The skewness of data can be categorized in various ways based on how they are distributed among the clients [LDCH22, KMA+21]. 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); (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. In this work, we focus on label skew. More in detail, given the features x, a label y, and the local distribution of a device i, denoted as:

$$\mathbb{P}_i(x, y) = \mathbb{P}_i(x \mid y) \cdot \mathbb{P}_i(y) \tag{3.1}$$

the concept of label skew, between two distinct devices k and j, is defined as follows [HYS+24]:

$$\mathbb{P}_k(y) \neq \mathbb{P}_j(y) \text{ and } \mathbb{P}_k(x \mid y) = \mathbb{P}_j(x \mid y) \tag{3.2}$$

In other words, different clients have distinct label distributions (*i.e.,* $\mathbb{P}_i(y)$) while maintaining the same underlying feature distribution for each label (*i.e.,* $\mathbb{P}_i(x \mid y)$).

This data heterogeneity can lead to *model drift* [WLL+20] during training, where the model's performance degrades as training progresses, potentially causing slow or unstable convergence. In particular, this phenomenon occurs when clients with significantly different data distributions produce highly divergent model updates, each optimized for their own local objectives. As a result, the aggregated global model may exhibit large parameter shifts across rounds, which can harm overall performance and hinder convergence. To address these challenges, several approaches have been proposed, such as *FedProx* [LSZ+20] and *Scaffold* [KKM+20].

3.2. **Field-Based Coordination.** *Coordination based on fields* [AVD+19] (or *field-based coordination*) employs a methodology where computations are facilitated through the concept of *computational fields* (*fields* in brief), defined as distributed data structures that evolve over time and map locations with specific values. This method draws inspiration from foundational works such as Warren's *artificial potential fields* (namely, a technique where robots navigate by following gradient-based force fields that attract them to goals while repelling them from obstacles) [War89] and the concept of *co-fields* by Zambonelli et al. [MZL04]. Specifically, in the context of co-fields, these computational fields encapsulate contextual data, which is sensed locally by agents and disseminated by either the agents themselves or the infrastructure following a specific distribution rule.

In our discussion, *coordination based on fields* refers to a distinct macroprogramming and computation framework, often referred to as *aggregate computing* (AC) [VBD+18]. This framework enables the programming of collective, self-organizing behaviors through the integration of functions that operate on fields, assigning computational values to a collection of individual agents (as opposed to locations within an environment). Thus, fields facilitate the association of a particular domain of agents with their sensory data, processed information, and instructions for action within their environment. Computed locally by agents yet under a unified perspective, fields enable the representation of collective instructions (*e.g.,* a velocity vector field or a field of machine learning model parameters). To comprehend field-based computing fully, we highlight the system model and the programming model in the following sections.

3.2.1. *System Model.* An aggregate system may be defined as a collection of *agents* (or *nodes*) that interact within a shared environment to achieve a common goal.

To better understand the system's behavior, we must first define the system's structure (*i.e.,* the agents and their relationships), the agents' interaction (*i.e.,* how they communicate), and their behavior within the environment (*i.e.,* how they process information and act).

Structure. An *agent* represents an autonomous unit furnished with *sensors* and *actuators* to interface with either a logical or physical *environment*. From a conceptual standpoint, an agent possesses *state*, capabilities for *communication* with fellow agents, and the ability to *execute* simple programs. An agent's *neighborhood* is composed of other *neighbor* agents, forming a connected network that can be also represented as a graph—see Section 5.1 for more details. The composition of this network is determined by a *neighboring relationship*, designed based on the specific application needs and constrained by the physical network's limitations. Commonly, neighborhoods are defined by physical connectivity or spatial proximity, allowing for communication directly or via infrastructural support, based on certain proximity thresholds.

Interaction. Agents interact by asynchronously sending messages to neighbors, which can also occur stigmergically—through indirect coordination via environmental modifications—through sensors and actuators. The nature and timing of these messages depend on the agent's programmed behavior. Notably, our focus on modelling continuous, self-organizing systems suggest frequent interactions relative to the dynamics of the problem and environment.

Behavior. Based on the interaction of agents, an agent's behavior unfolds through iterative *execution rounds*, with each round encompassing the steps below, albeit with some flexibility in the actuation phase:

(1) *Context acquisition:* agents accumulate context by considering both their prior state and the latest inputs from sensors and neighboring messages.
(2) *Computation:* agents process the gathered context, resulting in (i) an *output* for potential actions, and (ii) a *coordination message* for neighborly collective coordination.
(3) *Actuation and communication:* agents execute the actions as specified by the output and distribute the coordination message across the neighborhood.

By cyclically executing these sense-compute-act rounds, the system exhibits a self-organizing mechanism that integrates and processes fresh data from both the environment and the agents, typically achieving self-stabilization

3.2.2. *Programming model.* This system model establishes a foundation for collective adaptive behavior, necessitating an in-depth elucidation of the "local computation step", facilitated by a *field-based programming model*. This model orchestrates the collective behavior through a *field-based program*, executed by each agent in adherence to the prescribed model. Field calculus defines the computation as a composition of *function operations* on fields, and any variants of it allow the developers to express at least i) the temporal evolution of fields, ii) the data interchange among agents, and iii) the spatial partitioning of computation. In the rest, we will present aggregate computing through the scala framework ScaFi [CVAP22]. In this variant, the three main operator constructs are `rep`, `nbr`, and `foldhood`. For instance, to model the temporal evolution of a field, one can employ the `rep` construct as follows:

```
// signature: def rep[A](init: A)(f: A => A): A
rep(0)(x => x + 1)
```

Where 0 is the initial value of the field, and `x =>x + 1` is a lambda that increments the field value by one at each cycle. Hence, `rep` is the incremental evolution of a field with each cycle, representing a non-uniform field. Particularly, the above-described code express a field of local counters, where each agent increments its own counter at each cycle.

To facilitate data exchange between agents, ScaFi leverages the `nbr` construct in conjunction with a folding operator:

```
// signature: def nbr[A](value: A): A
// signature: def foldhood[A](init: A)(op: (A, A) => A)(query: A): A
foldhood(0)(_ + _)(nbr(1))
```

This snippet computes each agent's neighbor count: if agent A has 3 neighbors, the result will be 3, as it sums the values of its neighbors (1 in this case). Particularly, here, `nbr(1)` captures neighboring values through a bidirectional communication primitive (namely, the agent sends the value 1 to its neighbors and receives their values in return), `_ + _` serves

as the aggregation function (summation operator, equivalent to `(a, b) =>a + b`), and 0 provides the initial value for the reduction operation.

Lastly, to express spatio-temporal evaluation, a combination of the aforementioned constructs is utilized:

```
rep(mid) { minId => foldhood(0)(math.min)(nbr(minId)) }
```

This code snippet demonstrates a sophisticated interplay between temporal evolution and spatial aggregation to compute the global minimum identifier across the entire network. The computation operates as follows: initially, each agent starts with its local identifier (`mid`). At each execution round, the `rep` construct maintains the current minimum value discovered so far, while `nbr(minId)` enables each agent to share its current minimum with its neighbors. The `foldhood` operation then aggregates these neighbor values using the `math.min` function, effectively propagating the smallest identifier throughout the network. Through this iterative process, the minimum value gradually diffuses across the network topology: agents with smaller identifiers influence their neighbors, and this influence cascades through the network until all agents converge to the global minimum. The temporal aspect ensures convergence stability, while the spatial aspect enables information propagation across network boundaries.

3.2.3. *Coordination Building Blocks.* On top of this minimal set of constructs, ScaFi provides a set of building blocks for developing complex coordination algorithms. These blocks are *self-stabilizing*: they converge to a stable state despite transient changes in network topology or agent behaviors. A cornerstone among these constructs is the *self-healing gradient* computation, a distributed algorithm for maintaining the minimum distance from a designated source node (*e.g.,* the leader) to every other node in the network. Building upon standard gradient-based approaches, this mechanism automatically recomputes and updates distance estimates whenever changes occur in the network (*e.g.,* node arrivals/removals or communication failures), highlighting the algorithm's *self-healing* property and making it highly suitable for dynamic, large-scale environments. Within ScaFi, this is described as follows:

```
def gradient(source: Boolean): Double =
  rep(Double.MaxValue) { dist =>
    mux(source) { 0.0 }
                {foldhood(dist) { math.min }(nbr(dist) + nbrRange())
  }
```

Where the `rep` construct maintains a distance estimate that evolves over time, while `mux` conditionally sets the distance to zero for source nodes. For non-source nodes, `foldhood` aggregates neighbor distances using the minimum function, where each neighbor's distance is incremented by the communication range (`nbrRange()`). This creates a distributed shortest-path computation that automatically adapts to network topology changes. Building upon this foundation, more sophisticated coordination algorithms can be developed, such as:

- **Gradient cast**: a mechanism to disseminate information from a source node across the system using a gradient-based approach. In ScaFi, this is expressed as:

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

Here, `source` identifies which nodes serve as the gradient's origin points, `value` represents the information to be disseminated from source nodes throughout the network, and `accumulator` defines how the propagated value is transformed as it spreads from node to node. For example, `G[Int](sense("source"), 0, _ + 1)` creates a hop-distance field that increments by one at each node as it propagates away from the source, where `sense("source")` identifies the source nodes.

- **Collect cast**: conceptually the inverse of gradient cast, it aggregates information from the system back to a specific zone. It is represented as:

```
C[V](potential: Double, accumulator: V, localValue: V, null: V)
```

Here, `potential` defines the collection gradient that establishes a spanning tree for routing values to collection points (typically following shortest paths). The `accumulator` parameter specifies how values are aggregated as they flow toward the collection point, `localValue` represents each node's contribution to the collection, and `null` serves as the default value when no data is available. For example, `C(gradient(source), _ + _, 1, 0)` collects the sum of values from all nodes in the network.

- **Sparse choice**: a distributed mechanism for node subset selection and spatial-based leader election, expressed as:

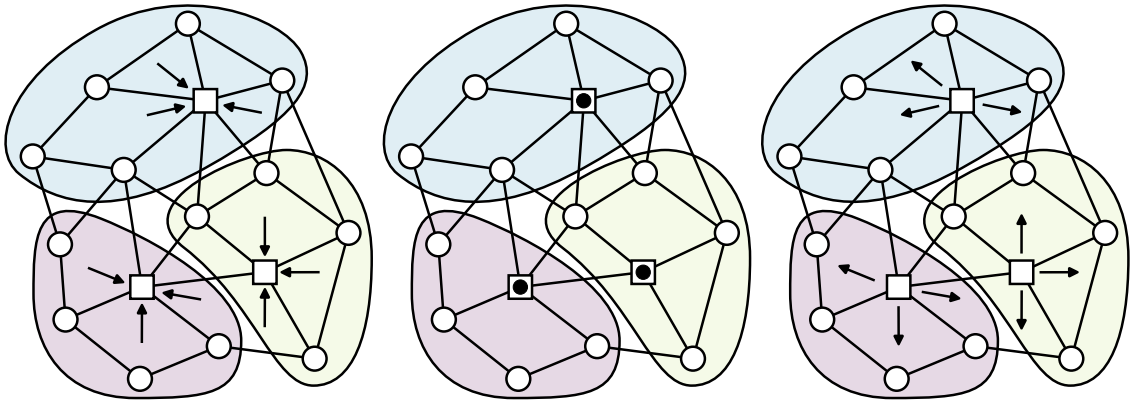
```
S(radius: Double, metric: => Double): Boolean
```

where `radius` specifies the maximum radius within which a leader can influence other nodes and `metric` defines distance used to evaluate node proximity (*e.g.,* Euclidean distance). The algorithm is "spatial-based" as it leverages physical distances between nodes to elect leaders: each leader node creates a sphere of influence with radius `radius`, and nodes within this radius become part of that leader's region.

By integrating these constructs, it becomes possible to execute complex collective behaviors, such as crowd management and swarm behaviors [ACV25a]. Furthermore, it is feasible to integrate the aforementioned programming model with deep learning techniques, advancing towards a paradigm known as *hybrid aggregate computing* [ACV22]. AC can orchestrate or enhance the learning mechanisms, and conversely, learning methodologies can refine AC. This paper adopts the initial perspective, delineating a field-based coordination strategy for FL. The objective is to create a distributed learning framework that is inherently self-stabilizing and self-organizing.

3.2.4. *Self-organizing Coordination Regions.* Recent advances in field-based coordination introduced a pattern called Self-organizing Coordination Regions (SCR) [CPVN19]. This strategy enables distributed collective sensing and acting without relying on a dedicated authority, all while ensuring both self-stabilization and self-organization of the system. The SCR pattern operates as follows:

(1) A distributed multi-leader election process selects a set of regional leaders across the network (*e.g.,* using the `S` operator);
(2) The system self-organizes into distinct regions, each governed by a single leader (*e.g.,* leveraging the `G` operator); and
(3) Within each region, a feedback loop is established where the leader collects upstream information from the agents under its influence and, after processing, disseminates downstream directives (using both the `C` and `G` operators, see Figure 2 for an overview).



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

FIGURE 2. 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.

This pattern effectively combines the previously discussed building blocks and is straightforward to implement in the aggregate computing context. The following ScaFi code demonstrates a compact implementation of the SCR pattern:

```
def SCR(radius: Double): Boolean = {
  val leader = S(radius) // Step 1: Elect leaders
  val potential = gradient(leader) // Step 2: Create regions
  val collectValue =
    C(potential, accumulationLogic, localValue, nullValue) // Step 3a: Collect data
  val decision = mux(leader) {
    decide(collectValue) // Step 3b: Leaders process data
  } {
    nullValue
  }
  G(leader, decision, identity) // Step 3c: Broadcast decisions
}
```

In this implementation, line 2 performs distributed leader election within the specified `radius`, line 3 establishes regions around each leader using gradient computation, lines 4-5 collect information from all agents within each region toward their respective leaders, lines 6-9 enable leaders to process the collected information and make decisions, and line 10 broadcasts these decisions back to all agents in each region.

4. Related Works and Motivation

In this section, we present several state-of-the-art methods that serve as baseline for comparison with the proposed approach. Additionally, we discuss the motivation for our work and describe a reference scenario that contextualizes our contribution.

4.1. **Baseline algorithms.**

FedAvg. One of the most common algorithms for horizontal FL is *FedAvg*, where the server aggregates the models by averaging the weights of the models received from the client devices. A *model* in machine learning refers to a mathematical representation (typically a neural network in our context) that learns patterns from data to make predictions or classifications. The *weights* (also called parameters) are the numerical values within the model that are learned during training and determine how input data is transformed to produce outputs. Formally, given a population of K devices, each of them with a local dataset D_k and size $n_k = |D_k|$. Each device k performs E epochs of *local training* on its dataset, producing an updated local model $\tilde{\mathbf{w}}_k^{(t)} \in \mathbb{R}^p$ in round t. The server then computes the weights of the global model $\mathbf{w}^{(t+1)}$ as a data-size weighted average of the vectors from the local models shared by the devices:

$$\mathbf{w}^{(t+1)} = \frac{1}{\sum_{k=1}^{K} n_k} \sum_{k=1}^{K} n_k \, \tilde{\mathbf{w}}_k^{(t)}. \tag{4.1}$$

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, define the proximalized local objective for device k at round t as

$$h_k(\mathbf{w}; \mathbf{w}^{(t)}) = F_k(\mathbf{w}) + \frac{\mu}{2} \|\mathbf{w} - \mathbf{w}^{(t)}\|_2^2, \tag{4.2}$$

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 controlling how far local updates can deviate from the global model. The local update approximately solves

$$\mathbf{w}_k^{(t+1)} \in \arg\min_{\mathbf{w}} \; h_k(\mathbf{w}; \mathbf{w}^{(t)}). \tag{4.3}$$

Scaffold. This algorithm introduces control variates—variance reduction techniques that track the local gradient drift—to correct the divergence of local models from the global optimum caused by heterogeneous data distributions across clients. 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 y_i using the formula $y_i \leftarrow y_i - \eta_l(g_i(y_i) + c - c_i)$, where η_l is the local learning rate, $g_i(y_i)$ represents the local gradient, and c and c_i are the server and client control variates respectively. The client's control variate c_i is then updated using either $c_i^+ \leftarrow g_i(x)$ or $c_i^+ \leftarrow c_i - \frac{1}{K\eta_l}(x - y_i)$, where 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 $x \leftarrow x + \eta_g \frac{1}{|S|} \sum_{i \in S}(y_i - x)$, where η_g is the global learning rate and S is the set of selected clients. Simultaneously, the server control variate is adjusted using $c \leftarrow c + \frac{1}{N} \sum_{i \in S}(c_i^+ - c_i)$, with N being the total number of clients.

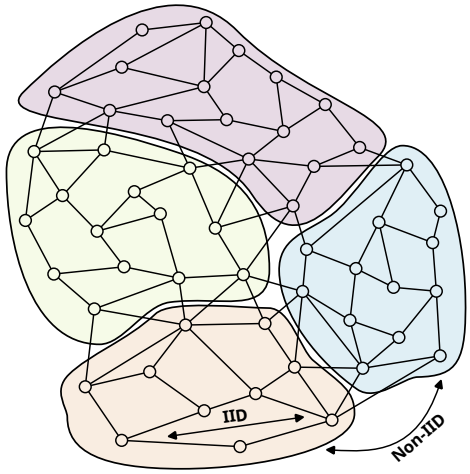


Figure 3. Spatial data distribution: homogeneous within subregions, non-IID across subregions.

4.2. **Motivation.** While these approaches to federate networks from multiple learning devices partially address non-IID data distribution challenges, they overlook a crucial aspect: the spatial distribution of devices and its relationship with data distribution patterns. Research has shown that devices in spatial proximity often exhibit similar data distributions, as they typically capture related phenomena or user behaviors [DAF+24]—see Figure 3. Consider, for instance, a motivating scenario involving autonomous vehicles (AVs) distributed across a metropolitan area collaboratively forecasting traffic conditions while preserving privacy. AVs operating in different environments (*e.g.,* highways, urban centers, or suburbs) collect data with distinct characteristics, highlighting strong spatial autocorrelation in traffic patterns. Collaboration among AVs in the same spatial region yields more accurate models, while aggregation across spatially dissimilar regions leads to degraded performance. This spatial correlation suggests that clustering devices based on spatial proximity could enhance model performance and personalization. However, current approaches in federated learning are predominantly centralized and rely on traditional clustering algorithms that do not consider the spatial aspects of data distribution. This gap highlights the need for an approach that simultaneously addresses: (i) decentralization, (ii) non-IID data handling, and (iii) spatial correlation in data distributions via distributed spatial-based leader election. Our work aims to bridge this gap through field-based coordination—see Table 1 as a comparison between current approaches and our proposal.

Table 1. Comparison of federated learning approaches.

Method	**Decentralized**	**Non-IID**	**Spatial Correlation**
FedAvg	✗	✗	✗
FedProx	✗	✓	✗
Scaffold	✗	✓	✗
P2P FL	✓	✗	✗
FBFL	✓	✓	✓

4.3. **Comparison with Other FL Variants.** FBFL operates within the horizontal FL paradigm, and more specifically aligns with the family of personalized and clustered FL approaches designed to mitigate the impact of non-IID data distributions (*e.g.,* [FMO20, GCYR22, RDF25]). In our work, the formation of self-organizing learning regions constitutes a fully distributed realization of clustered FL [LYG+25], where spatial proximity is used as a proxy for data similarity. This design eliminates the need for centralized clustering procedures and allows continuous reconfiguration in response to changes in network topology or data distribution—a property rarely achieved in centralized clustered FL.

By contrast, vertical FL addresses scenarios where participants share the same sample space but hold disjoint feature spaces [YLCT19, LKZ+24], and therefore does not directly address challenges such as label skew or heterogeneous sample distributions across devices—the primary focus of FBFL.

By integrating the adaptability and personalization strengths of clustered FL with the resilience and scalability of a self-organizing architecture, FBFL advances the state of the art in handling spatially correlated non-IID data in fully decentralized settings.

5. Field-Based Federated Learning

This section presents Field-Based Federated Learning (FBFL), our core contribution that integrates aggregate computing principles with federated learning to address the limitations identified in Sections 3 and 4. The following subsections formalize the problem (Section 5.1), detail our distributed algorithm (Section 5.2), and provide implementation specifics (Section 5.3). Finally, Table 2 summarizes the notation used throughout this section.

5.1. **Problem Formulation.** Consider a distributed network comprising a set of devices, denoted as $\mathcal{V}$. In our field-based approach, devices interact through direct peer-to-peer communication rather than relying on a central server. This network can be modelled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with nodes representing devices and edges symbolizing direct communication links. The neighbors of a device d, with which d can exchange messages, are denoted by $\mathcal{N}_d = \{d' \in \mathcal{V} \mid (d, d') \in \mathcal{E}\}$.

Assumption 1 (Network Connectivity)**.** *The communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected, ensuring field-based coordination can eventually propagate information between any two devices.*

Assumption 2 (Reliable Message Delivery)**.** *Communication between neighboring devices $d, d' \in \mathcal{V}$ where $(d, d') \in \mathcal{E}$ is reliable, though messages may experience bounded delays.*

Each device $d \in \mathcal{V}$ possesses a local dataset $\mathcal{D}_d = \{(x_i, y_i)\}_{i=1}^{n_d}$ where $x_i \in \mathbb{R}^m$ represents the input features, $y_i \in \mathcal{Y}$ denotes the corresponding labels, and $n_d = |\mathcal{D}_d|$ is the number of samples on device d. The local data distribution $\mathbb{P}_d(x, y)$ may vary across devices, particularly in non-IID scenarios where $\mathbb{P}_d \neq \mathbb{P}_{d'}$ for $d \neq d'$.

The dataset on device d is used to train a local model, characterized by a weights vector $\mathbf{w}_d \in \mathbb{R}^p$, where p represents the total number of parameters in the neural network architecture.

Assumption 3 (Shared Model Initialization)**.** *All devices $d \in \mathcal{V}$ start with the same initial model $\mathbf{w}_d^{(0)} = \mathbf{w}^{(0)}$, ensuring a common starting point for the learning process. This may be a pre-trained model or a randomly initialized model shared via a global elected leader.*

Table 2. Symbols used in Section 5 (Field-Based Federated Learning).

Symbol	Meaning
$\mathcal{V}$	Set of devices (nodes).
$\mathcal{E}$	Set of communication links (edges).
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Communication graph.
$\mathcal{N}_d$	Neighborhood of device d: $\{d' \in \mathcal{V} \mid (d, d') \in \mathcal{E}\}$.
$\mathcal{D}_d$	Local dataset on device d; $n_d = \lvert\mathcal{D}_d\rvert$ is its size.
$x_i \in \mathbb{R}^m,\ y_i \in \mathcal{Y}$	Feature vector and label of a sample; m is feature dimension.
$\mathbb{P}_d(x, y)$	Local data distribution at device d.
$\mathbf{w}_d \in \mathbb{R}^p$	Parameters of the model at device d; p is number of parameters.
$\mathbf{w}^{(0)}$	Initial shared model.
$\tilde{\mathbf{w}}_d^{(t)}$	Locally updated model of device d at round t.
$\mathbf{w}_l^{(t)}$	Regional model associated with leader l at round t.
$t \in \{0, \dots, \mathcal{T}\}$	Federated learning round index; $\mathcal{T}$ is the maximum number of rounds.
$f_d(\mathbf{w})$	Local empirical risk on device d.
$\ell(\cdot, \cdot)$	Loss function (*e.g.,* cross-entropy).
$h(\mathbf{w}; x)$	Model prediction for input x under parameters $\mathbf{w}$.
$\alpha_d,\ \alpha_{d,l}^{(t)}$	Data-size weights (global and regional, respectively).
$\mathcal{L}^{(t)}$	Set of elected leaders at round t.
$\mathcal{R}_l^{(t)}$	Region (influence set) of leader l at round t; $\{\mathcal{R}_l^{(t)}\}_{l \in \mathcal{L}^{(t)}}$ partitions $\mathcal{V}$.
$d_{\mathcal{L}}^{(t)}$	Leader assigned to device d at round t.
$\mathrm{dist}(d, l)$	Neighborhood metric between devices d and l (*e.g.,* Euclidean in space or hop distance on $\mathcal{G}$).
$D(\mathbb{P}_d, \mathbb{P}_{d'})$	Statistical distance between local distributions.
ϵ	Threshold for spatial-data similarity (Assumption 5).
E	Number of local training epochs per round.
$\mathcal{DL}(\mathrm{dist}, R)$	Distributed leader election procedure parameterized by metric and radius.
$d' \Longrightarrow_t d''$	Forward chain (multi-hop path) from node d' to d'' at round t.
R	Influence radius (in metric units or hops): maximum distance a leader influences.
$\mathcal{A}$	Aggregation operator (*e.g.,* data-size weighted average).

Assumption 4 (Synchronized Federated Learning Rounds)**.** *There exists a global synchronization mechanism that ensures all devices progress through federated learning rounds in discrete time steps $t \in \{0, 1, 2, \dots, \mathcal{T}\}$, where devices complete local training, model sharing, and aggregation phases within bounded time intervals before proceeding to round $t + 1$.*

At federated learning round t, the local objective function $f_d : \mathbb{R}^p \to \mathbb{R}$ measures the empirical risk on device d's local dataset:

$$f_d(\mathbf{w}_d^{(t)}) = \frac{1}{n_d} \sum_{i=1}^{n_d} \ell(h(\mathbf{w}_d^{(t)}; x_i), y_i), \tag{5.1}$$

where $h(\mathbf{w}_d^{(t)}; x_i)$ denotes the model's prediction for input x_i given parameters $\mathbf{w}_d^{(t)}$, and $\ell(\cdot, \cdot)$ represents the loss function (*e.g.,* cross-entropy for classification tasks).

In traditional federated learning, the global objective seeks to minimize a weighted combination of all local objectives:

$$f(\mathbf{w}) = \sum_{d \in \mathcal{V}} \alpha_d f_d(\mathbf{w}), \tag{5.2}$$

where $\alpha_d = \frac{n_d}{|D|}$ represents the relative weight of device d's data contribution, with $|D| = \sum_{d \in \mathcal{V}} n_d$ being the total number of samples across all devices.

In contrast, our field-based federated learning approach creates multiple specialized regional models through distributed spatial-based leader election. Let $\mathcal{L}^{(t)} \subseteq \mathcal{V}$ denote the set of elected leaders at round t, where each leader $l \in \mathcal{L}^{(t)}$ governs a region $\mathcal{R}_l^{(t)} \subset \mathcal{V}$ of nearby devices, and $\{\mathcal{R}_l^{(t)}\}_{l \in \mathcal{L}^{(t)}}$ forms a partition of $\mathcal{V}$. Rather than optimizing a single global model, we minimize regional objectives at each round:

$$f_l^{(t)}(\mathbf{w}_l) = \sum_{d \in \mathcal{R}_l^{(t)}} \alpha_{d,l}^{(t)} f_d(\mathbf{w}_l), \tag{5.3}$$

where $\alpha_{d,l}^{(t)} = \frac{n_d}{\sum_{d' \in \mathcal{R}_l^{(t)}} n_{d'}}$ represents the weight of device d's data within region l at round t (note that $\sum_{d \in \mathcal{R}_l^{(t)}} \alpha_{d,l}^{(t)} = 1$), and $\mathbf{w}_l$ is the regional model optimized for devices in $\mathcal{R}_l^{(t)}$.

Assumption 5 (Spatial-Data Correlation)**.** *Devices in spatial proximity exhibit similar data distributions,* i.e., *for devices $d, d' \in \mathcal{R}_l^{(t)}$ within the same region l, the statistical distance between their local distributions satisfies $D(\mathbb{P}_d, \mathbb{P}_{d'}) < \epsilon$ for some threshold $\epsilon > 0$ (for some statistical distance D,* e.g., *total variation or Wasserstein), while devices across different regions may have significantly different distributions.*

Discussion of Modeling Assumptions. The theoretical framework of FBFL rests on several key assumptions that warrant further discussion regarding their practical implications. Regarding Assumption 1, strict, permanent full connectivity is not a prerequisite for the system's operation. As long as a communication path exists between any pair of devices, field-based coordination mechanisms ensure that information eventually propagates across the network, with convergence occurring in $O(D)$ time, where D denotes the network diameter. Concerning Assumption 2, although our experimental evaluation does not explicitly consider message losses, prior work [ACV25b] on aggregate computing and its foundational building blocks has extensively studied unreliable communication settings, demonstrating robust collective behavior even under severe message drop rates (up to 70%). Assumption 3 can be satisfied in practice through a lightweight initialization phase: a preliminary gossip-based consensus step is sufficient to align devices on a shared initial model before the learning process starts. Finally, Assumption 5 reflects empirical evidence observed in many real-world scenarios, where spatial proximity induces correlations in local data distributions, yielding approximately i.i.d. data within regions and heterogeneous distributions across regions. While purely spatial partitioning may sometimes be overly restrictive, recent extensions of this line of work [DFA+26] propose adaptive notions of *space fluidity*, in which initially spatial regions are dynamically refined–expanded or contracted–based on additional metrics, such as proxies for data similarity.

5.2. **Algorithm.** Our implementation follows a semi-centralized approach where a subset of nodes are dynamically elected as *aggregators*—specialized devices responsible for model aggregation within their regions. This hierarchical structure is self-organizing: elected leaders adapt to network topology changes (*e.g.,* node failures or mobility), ensuring resilient coordination without manual intervention. To understand the algorithm's operation, it is essential to formalize the concept of *forward chain*:

Each node belongs to exactly one leader's region and communicates its model exclusively with that designated leader. We employ a distance-based leader selection rule that establishes a Voronoi-like partitioning of the network. Formally, given a node d and the leader set $\mathcal{L}^{(t)} \subseteq \mathcal{V}$ computed by the distributed leader election algorithm $\mathcal{DL}(\text{dist}, R)$ at round t, with an application-chosen influence radius $R > 0$ and a time-independent neighborhood metric $\text{dist}(\cdot,\cdot)$ (*e.g.,* Euclidean in physical space or hop-count over $\mathcal{G}$), node d falls under the influence of leader $l \in \mathcal{L}^{(t)}$ if and only if:

$$\text{dist}(d,l) \leq R \quad \text{and} \quad \forall\, l' \in \mathcal{L}^{(t)} \setminus \{l\},\ \text{dist}(d,l) < \text{dist}(d,l'), \tag{5.4}$$

Note that, for every node d there exists a leader l with $\text{dist}(d,l) \leq R$, hence each node is assigned. In case of no leader being within radius R, the node becomes leader $d_{\mathcal{L}}^{(t)} = d$. In cases of equidistance, a predefined tie-breaking rule is applied. The set of nodes under leader l's influence is denoted $\mathcal{R}_l^{(t)}$, while $d_{\mathcal{L}}^{(t)}$ represents the leader to which node d is assigned at round t. A forward chain from node d' to node d'' at round t (denoted $d' \Longrightarrow_t d''$) is defined as the sequence, constrained to remain within the leader's influence radius when d'' is a leader:

$$d_1, d_2, \ldots, d_k \quad \text{such that} \quad d_1 = d', \quad d_k = d'', \quad \text{and} \quad (d_i, d_{i+1}) \in \mathcal{E} \ \text{ for } i = 1, \ldots, k-1, \tag{5.5}$$

with $\mathcal{E}$ possibly depending on t in time-varying topologies. This formalization enables communication between nodes lacking direct connectivity through a chain of intermediate connections; in practice, routing follows the gradient-induced minimal-potential paths computed by the coordination layer.

Given these definitions, the round-t evolution is as follows (see Algorithm 1 for a comprehensive overview):

(1) *Leader Election:* A distributed leader election algorithm $\mathcal{DL}(\text{dist}, R)$ is executed to select a set of leaders $\mathcal{L}^{(t)} \subseteq \mathcal{V}$ and induce regions $\{\mathcal{R}_l^{(t)}\}_{l \in \mathcal{L}^{(t)}}$.

Assumption 6 (Gradual Topology Changes)**.** *Network topology and leader set changes occur at a rate slower than field computation convergence,* i.e., $T_{topology} \gg T_{field}$*, where* T_{field} *and* $T_{topology}$ *denote field stabilization time and average time between topology changes, respectively.*

(2) *Local Training:* Each device $d \in \mathcal{V}$ starts from $\mathbf{w}_d^{(t)}$ and performs local training on $\mathcal{D}_d$ for E epochs, producing an updated local model $\tilde{\mathbf{w}}_d^{(t)}$ by (approximately) minimizing $f_d(\cdot)$.
(3) *Leader Selection:* Each device d determines its leader $d_{\mathcal{L}}^{(t)} \in \mathcal{L}^{(t)}$ based on its distance to the leaders, constrained by radius R; following the partitioning rule described above.
(4) *Model Sharing:* Each device d shares its local model $\tilde{\mathbf{w}}_d^{(t)}$ with its leader $d_{\mathcal{L}}^{(t)}$ along the forward chain $d \Longrightarrow_t d_{\mathcal{L}}^{(t)}$, which by construction lies within radius R at steady state.
(5) *Model Aggregation:* Each leader $l \in \mathcal{L}^{(t)}$ aggregates the models from all devices in its influence region $\mathcal{R}_l^{(t)}$ using an aggregation function $\mathcal{A}$, computing the new regional model

Algorithm 1 Field-Based Federated Learning Algorithm

Require: number of devices K, Network graph $\mathcal{G}$, Epochs per round E, Maximum rounds $\mathcal{T}$, Initial model $\mathbf{w}^{(0)}$

procedure FBFL
  for round $= 0$ **to** $\mathcal{T}-1$ **do**
    Leaders $\leftarrow \mathcal{DL}(\text{dist}, R)$ ▷ election at round t
    for all $d \in \mathcal{V}$ **do**
      local training: from $\mathbf{w}_d^{(t)}$ produce $\tilde{\mathbf{w}}_d^{(t)}$ on d's dataset for E epochs
      leader selection: Compute $d_{\mathcal{L}}^{(t)}$ using Leaders
      model sharing: Send $\tilde{\mathbf{w}}_d^{(t)}$ to $d_{\mathcal{L}}^{(t)}$ following $d \Longrightarrow_t d_{\mathcal{L}}^{(t)}$
    end for
    for all $d \in \mathcal{V}$ **do**
      if $d \in$ Leaders **then**
        model aggregation: $\mathbf{w}_d^{(t+1)} \leftarrow \mathcal{A}\big(\mathbf{w}_d^{(t)}, \{\tilde{\mathbf{w}}_{d'}^{(t)} \mid d' \in \mathcal{R}_d^{(t)}\}\big)$
        for all $d' \in \mathcal{R}_d^{(t)}$ **do**
          dissemination: deliver $\mathbf{w}_d^{(t+1)}$ to d' following $d \Longrightarrow_t d'$
        end for
      end if
    end for
    Each $d' \in \mathcal{V}$ sets $\mathbf{w}_{d'}^{(t+1)}$ to the received regional model
  end for
end procedure

weights for the next round:

$$\mathbf{w}_l^{(t+1)} = \mathcal{A}\Big(\mathbf{w}_l^{(t)}, \Big\{\big(\tilde{\mathbf{w}}_d^{(t)}, n_d\big) \mid d \in \mathcal{R}_l^{(t)}\Big\}\Big) \quad \textit{e.g.,} \quad \mathbf{w}_l^{(t+1)} = \sum_{d \in \mathcal{R}_l^{(t)}} \alpha_{d,l}^{(t)} \tilde{\mathbf{w}}_d^{(t)}. \tag{5.6}$$

Assumption 7 (Aggregation Capacity Constraints)**.** *Each elected leader $l \in \mathcal{L}^{(t)}$ possesses sufficient computational and communication resources to aggregate models from all devices in its influence region $\mathcal{R}_l^{(t)}$ within the available time window of round t.*

(6) *Model Dissemination:* Each leader l disseminates $\mathbf{w}_l^{(t+1)}$ to all devices in its influence region $\mathcal{R}_l^{(t)}$ (bounded by R) following $l \Longrightarrow_t d'$ for all $d' \in \mathcal{R}_l^{(t)}$, and devices update

$$\forall d \in \mathcal{R}_l^{(t)} : \quad \mathbf{w}_d^{(t+1)} \coloneqq \mathbf{w}_l^{(t+1)}. \tag{5.7}$$

Steps 1–6 are repeated for $t = 0, 1, \ldots, \mathcal{T}-1$, allowing regional models to evolve iteratively based on local training and field-coordinated aggregation.

5.3. **Implementation Details.** Our hierarchical FL framework represents a direct application of the Self-organizing Coordination Regions (SCR) pattern [CPVN19]. The following ScaFi implementation demonstrates this field-based coordination:

Listing 1. FBFL implementation using aggregate computing primitives.

```
val leaders = S(radius, nbrRange) // Spatial-based leader election
def syncCondition: Boolean = ??? // Define synchronization condition through shared clock
rep(initModel())(localModel => {
  if(syncCondition) { localModel.trainLocally() } // Local SGD epoch training step
  val potential = gradient(leaders) // Compute routing potential
  val collectedModels = C(potential, _ ++ _, Set(localModel), Set.empty)
  val aggregatedModel = if (leaders) fedAvg(collectedModels) else emptyModel
  val globalModel = G(leaders, aggregatedModel, identity)
  mux(syncCondition) { globalModel } { localModel }
})
```

Building Blocks Semantics. This implementation leverages the aggregate computing primitives introduced in Section 3.2 to provide distributed model coordination. The formal algorithm in Algorithm 1 maps directly to these field-based operations:

- *Leader Election* (Step 1): Implemented using `S(radius, nbrRange)` for spatial-based leader selection, computing $\mathcal{L}^{(t)} = \mathcal{DL}(\text{nbrRange}, \text{radius})$.
- *Model Sharing* (Step 4): The forward chain $d \Longrightarrow_t d_{\mathcal{L}}^{(t)}$ is based on collect-cast `C(potential, _ ++ _, Set(localModel), Set.empty)`, where the potential field routes model updates $\tilde{\mathbf{w}}_d^{(t)}$ toward appropriate aggregators.
- *Model Dissemination* (Step 6): Broadcast from leaders to regions $l \Longrightarrow_t d'$ via gradient-cast `G(leaders, aggregatedModel, identity)`, delivering $\mathbf{w}_l^{(t+1)}$ to all $d' \in \mathcal{R}_l^{(t)}$.

Execution cadence. Devices execute the aggregate program asynchronously at frequency f_{field} (Sec. 3.2). FL rounds advance when a shared-clock condition [PBV16] holds (`syncCondition`), typically at a lower frequency f_{FL}. When `syncCondition` becomes true (entering round t), devices rely on the stabilized coordination fields (leaders and routing) to advance one federated-learning round: (i) they perform local training, and (ii) leaders aggregate a new regional model. Model propagation (upstream to leaders and downstream to devices) is continuous and may span multiple field-update steps; it is decoupled from, and may not complete within, the same round.

Convergence and cost. Leader election and both upstream/downstream model exchange stabilize in $\mathcal{O}(D)$ field-update steps, where D is the communication-graph diameter, since gradients converge along shortest paths [MADB22].

Decoupling of rates. Running the field layer at $f_{\text{field}} \gg f_{\text{FL}}$ keeps leaders and potential fields correct, so when `syncCondition` triggers, devices use a stabilized coordination state. This satisfies the round-level synchronization requirements of Assumption 4.

Inconsistent states. It is also feasible to set $f_{\text{field}} = f_{\text{FL}}$, *i.e.,* run coordination and FL rounds at the same cadence. This reduces the number of models exchanged during each round, but immediately after aggregation some nodes may temporarily rely on stale gradients/routing, causing short-lived regional inconsistencies (e.g., adjacent nodes bound to different leaders or holding pre- vs. post-aggregation models) until the fields restabilize. We used this configuration in our experiments; despite these transients, the system quickly re-stabilized and showed robust performance.

6. Experimental Evaluation

6.1. **Experimental setup.** To evaluate the proposed approach, we conducted experiments on three well-known used computer vision datasets: MNIST [LCB+10], FashionMNIST [XRV17], and Extended MNIST [CATvS17]—their characteristics are summarized in Table 3. In particular, data import and partitioning were efficiently managed by leveraging the ProFed benchmark [DAV25]. We employed three state-of-the-art federated learning algorithms for comparison: FedAvg [MMR+17], FedProx [LSZ+20], and Scaffold [KKM+20].

First, we evaluated FBFL against FedAvg under a homogeneous data distribution to assess its stability in the absence of data skewness. We then created multiple heterogeneous data distributions through synthetic partitioning and compared FBFL with all three baseline algorithms. Specifically, we employed two partitioning strategies: (i) a Dirichlet-based split, where each party received samples from most classes, but with a highly imbalanced distribution, leading to certain labels being significantly underrepresented or overrepresented (as previously done in the literature, e.g., [LKSJ20, YAG+19]); and (ii) a hard data partitioning strategy [XYDH25, GSH+24], where each region contained only a subset of labels, resulting in a more challenging learning scenario. A graphical representation of these distributions is given in Figure 4. In particular, the dataset partitioning process used to assign data to each device followed these steps: (i) The dataset was first partitioned according to a predefined distribution (*e.g.,* Dirichlet or Hard), generating a separate dataset for each subregion; (ii) Each device was then assigned to one (and only one) subregion; and (iii) For each device, a subset of data was randomly sampled from the dataset of its corresponding subregion. Sampled data points were removed from the subregion's dataset to ensure that no two devices shared the same data samples. As a result, the data assigned to devices within the same subregion are IID, while those assigned to devices across different subregion are non-IID (Figure 3). Experiments were conducted using different numbers of regions, specifically $A \in \{3, 6, 9\}$, and 50 devices.

Finally, we assessed the resilience of our self-organizing hierarchical architecture by simulating a failure scenario. In this experiment, we considered a setup with four subareas and initially allowed our proposed learning process to run as intended. After a predefined amount of time—specifically, after 10 global communication rounds—we simulated the failure of two out of the four leader nodes. The failure model was designed by randomly selecting two leader nodes and severing their communication links with all other nodes in the system, effectively isolating them. This experiment was intended to evaluate whether the system could self-stabilize by re-electing new leader nodes to function as aggregator servers, thereby effectively restoring a stable federation configuration aligned with the subarea structure.

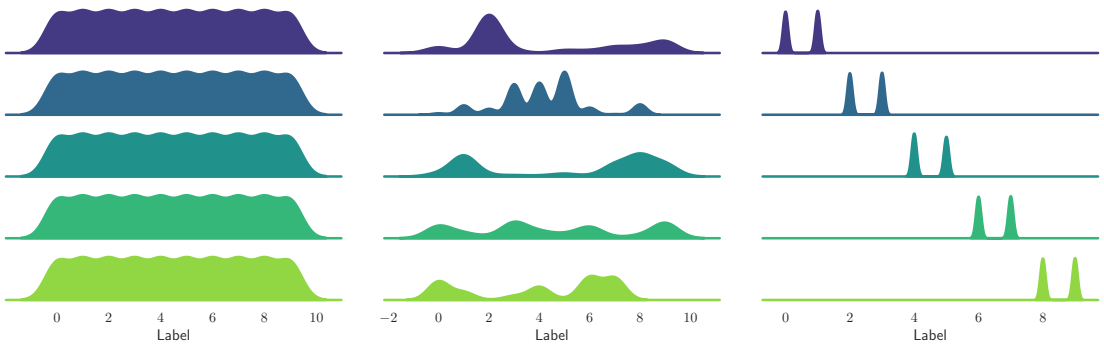
All experiments were implemented using PyTorch [PGC+17] for training. Additionally, the proposed approach leverages ScaFi for aggregate computing and ScalaPy [LS20] to enable interoperability between ScaFi and Python. Moreover, we used a well-known simulator for pervasive systems, namely: Alchemist [PMV13]. To ensure robust results, each experiment was repeated five times with different seeds.

To promote the reproducibility of the results, all the source code, the data, and instructions for running have been made available on GitHub; both for the baselines[1] and for the proposed approach[2].

[1] `https://github.com/davidedomini/experiments-2025-lmcs-field-based-FL-baselines`
[2] `https://github.com/davidedomini/experiments-2025-lmcs-field-based-FL`

Table 3. Overview of the datasets used in the experiments.

Dataset	Training Instances	Test Instances	Features	Classes
MNIST	60 000	10 000	784	10
Fashion MNIST	60 000	10 000	784	10
Extended MNIST	124 800	20 800	784	27



(a) IID data. (b) Dirichlet distribution. (c) Hard partitioning.

Figure 4. A graphical representation of three different data distribution in 5 subregions. Each color represents a different subregion. The second and the third images are two examples of non-IID data.

6.2. **Discussion.** In the following, we present the results of our experiments, comparing the proposed approach with the baseline algorithms under different conditions and replying to the research questions posed in Section 2.

RQ1

How does the Field-Based Federated Learning approach perform compared to centralized FL under IID data?

To answer RQ1, we evaluated FBFL against FedAvg under homogeneous data distribution. The goal of this evaluation was to show how the proposed approach, based on field coordination, achieves comparable performance to that of centralized FL while introducing all the advantages discussed in Section 5, such as: the absence of a single point of failure and the adaptability. Figure 5 shows results on the MNIST (first row) and Fashion MNIST (second row) datasets. It is worth noting that both the training loss and the validation accuracy exhibit similar trends. As expected, our decentralized approach shows slightly more pronounced oscillations in both metrics compared to FedAvg, due to the lack of a central coordinating authority. However, despite these inherent fluctuations in the decentralized setting, FBFL still achieves comparable final performance after the same number of global communication rounds, matching the effectiveness of traditional centralized learning methods as observed in the literature.

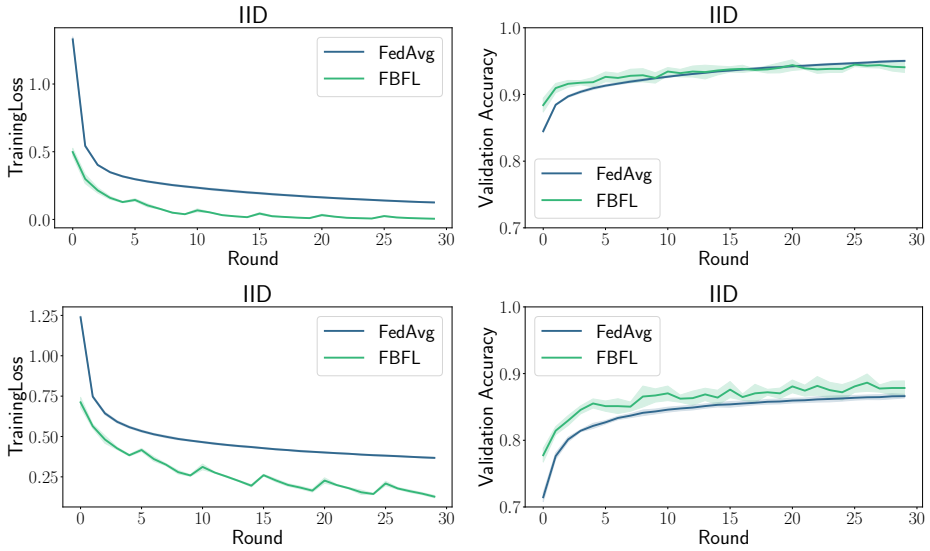


FIGURE 5. Comparison of the proposed method (FBFL) with FedAvg under IID data. The first row represents results on the MNIST dataset, while the second row on the Fashion MNIST dataset.

RQ2

Can we increase accuracy by introducing personalized learning zones where learning devices *are grouped based on similar experiences as often observed in spatially near locations? More precisely, what impact does this have on heterogeneous and non-IID data distributions?*

To rigorously evaluate our approach under non-IID conditions, we conducted extensive experiments comparing FBFL against baseline methods. We systematically explored two distinct types of data skewness: Dirichlet-based distribution (creating imbalanced class representations) and hard partitioning (strictly segregating classes across regions). The results reveal that baseline methods suffer from substantial performance degradation under these challenging conditions. These approaches consistently fail to capture the global objective and exhibit significant instability during the learning process. This limitation becomes particularly severe in scenarios with increased skewness, where baseline models demonstrate poor generalization across heterogeneous data distributions. Figure 6 presents key results from our comprehensive evaluation—which encompassed over 400 distinct experimental configurations. The first row shows results from the Extended MNIST dataset using hard partitioning across 6 and 9 distinct areas. The performance gap is striking: baseline algorithms plateau at approximately 0.5 accuracy, while FBFL maintains robust performance above 0.95. Notably, increasing the number of areas adversely affects baseline models' performance, leading to further accuracy deterioration. In contrast, FBFL demonstrates remarkable stability, maintaining consistent performance regardless of area count. The second and third rows present results from Fashion MNIST and MNIST experiments, respectively. While baseline methods show marginally better performance on these datasets (attributable to their reduced complexity compared to EMNIST), they still significantly underperform

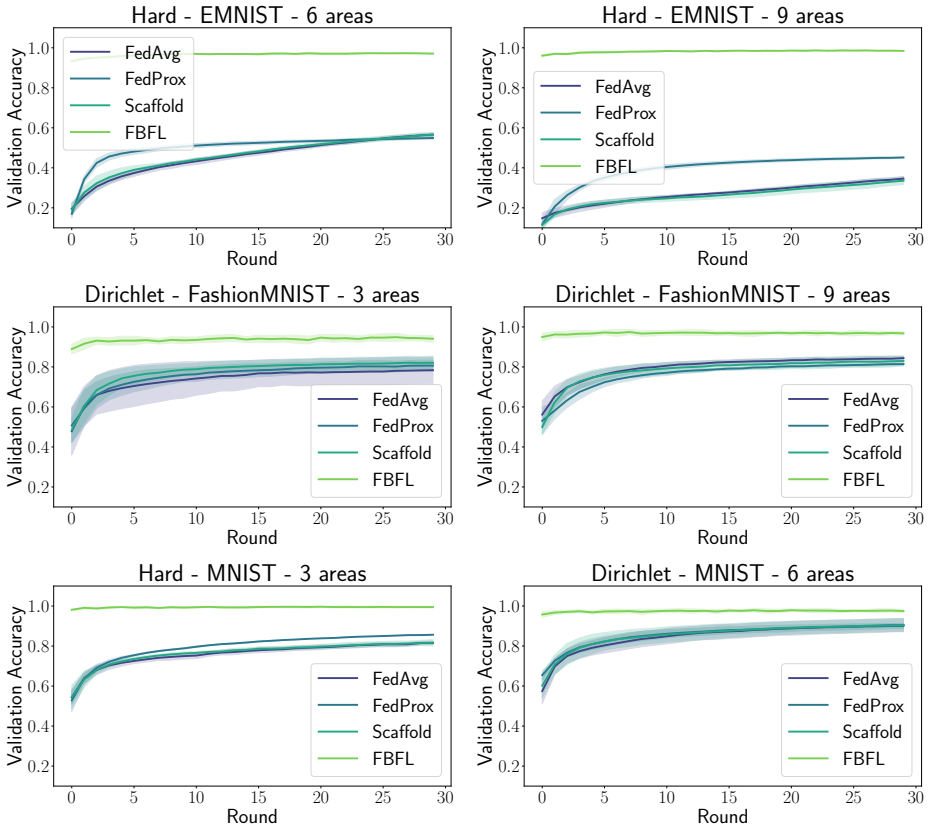


Figure 6. Comparison of the proposed method (FBFL) with all the baselines under non-IID data. The first row represents results on the Extended MNIST dataset, while the second row on the Fashion MNIST dataset and the third on the MNIST dataset.

compared to FBFL. These comprehensive findings underscore the fundamental limitations of traditional approaches in handling highly skewed non-IID scenarios. The results provide compelling evidence for RQ2, demonstrating FBFL's superior capability in maintaining high performance and stability across diverse data distributions through its innovative field-based coordination mechanism.

RQ3

What is the effect of creating a self-organizing hierarchical architecture through Field Coordination on Federated Learning in terms of resilience, adaptability and fault-tolerance?

The last experiment has been designed to evaluate the resilience of the self-organizing hierarchical architecture proposed by FBFL (RQ3). We simulated a scenario involving 4 distinct areas and a total of 50 devices. As for the other experiments, we ran 5 simulations with varying seeds. After allowing the system to stabilize under normal conditions (Figures 8a to 8c), we introduced a disruption by randomly killing two aggregator devices within the

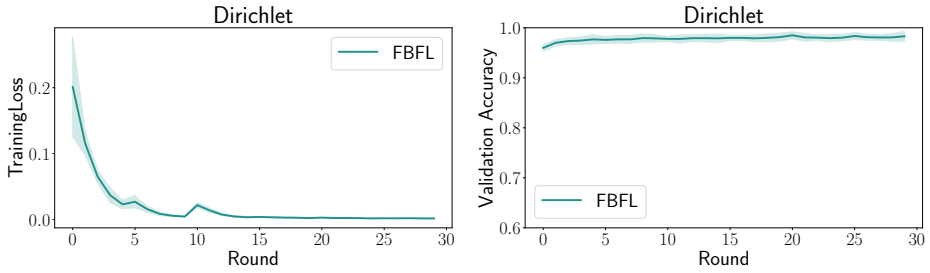


FIGURE 7. Evaluation metrics in the scenario of aggregators failure. It can be observed that, despite the failure of two aggregators, the evaluation metrics do not undergo significant variations in either the training or validation phases, and the learning process continues. Notably, the only observable effect is a slight increase in the loss at training time step 10, *i.e.,* when the aggregators fail and the system needs to re-stabilize, before it quickly resumes its downward trend.

network (Figure 8D). 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 depicted in performance charts (Figure 7) and visualized in the Alchemist simulator (Figure 8), demonstrate that the FBFL architecture successfully re-stabilizes into a new configuration. Specifically, the performance charts indicate only a minor increase in loss during the killing phase (global round 10), with no significant long-term drops in performance. Similarly, the Alchemist simulation shows a temporary transitional period following the removal of the aggregators, during which federations are not correctly formed (Figure 8E). However, the system adapts by electing new aggregators, and the configuration stabilizes once more (Figure 8F).

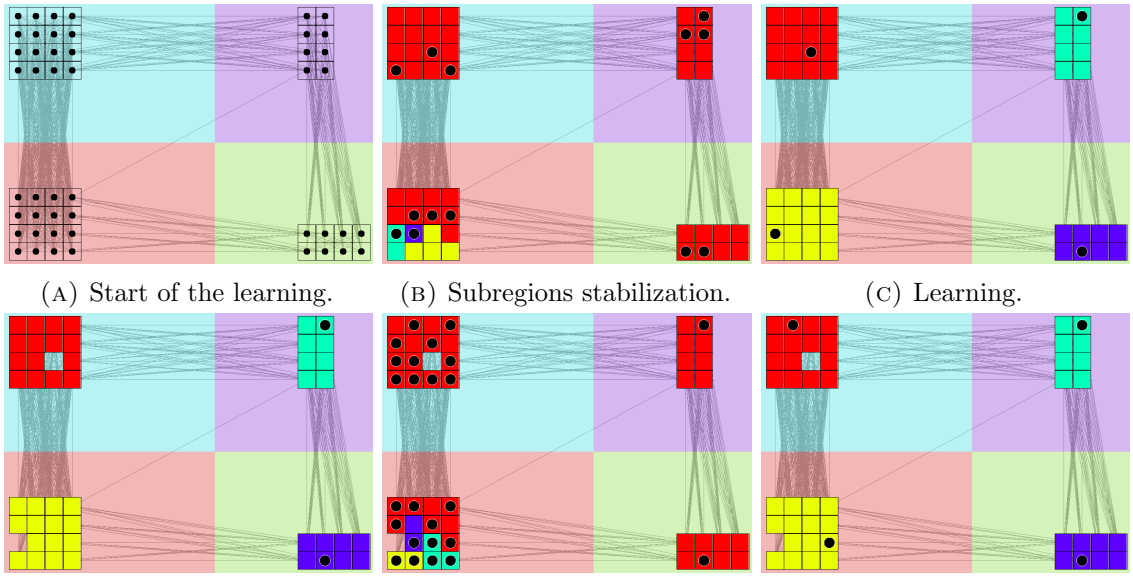
These findings highlight the robustness of the FBFL architecture, emphasizing its capability to recover from disruptions and maintain stable performance. This resilience is a critical feature for real-world applications where system stability under failure conditions is essential.

It can be observed that in unstable configurations, multiple aggregators are present in each subregion, whereas, once the system stabilizes, only a single aggregator remains per subregion.

7. LIMITATIONS

While the proposed approach demonstrates significant advantages in terms of adaptability and resilience, it is important to acknowledge several limitations that warrant consideration for future improvements.

Communication overhead and efficiency. FBFL inherently requires more frequent communication exchanges compared to traditional centralized approaches, since nodes must continuously exchange model updates with their neighbors: the gradient computation and collect-cast operations, while enabling self-organization, introduce additional message overhead that scales with network density. In resource-constrained environments or networks with limited bandwidth, this increased communication burden may hinder scalability and



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

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

Figure 8. Graphical representation of the simulation from the Alchemist simulator. In this scenario, 50 nodes (*i.e.,* the squares) are deployed in 4 different subregions. Background colors represent different data distribution, while nodes color represent their respective federation. Black dots inside nodes represent aggregators. It can be observed that in unstable configurations, multiple aggregators are present in each subregion, whereas, once the system stabilizes, only a single aggregator remains per subregion. After the learning has started and the systems has found a stable configuration, we randomly killed two aggregators node to simulate server failures. Notably, it is possible to see how the system is able to automatically re-stabilize.

energy efficiency; nevertheless, FBFL generally incurs lower overhead than fully peer-to-peer schemes, where every client shares its model with all others. Future work will explore communication-efficient variants, such as quantization techniques or sparse neural networks as discussed in recent works [DEA+25].

Spatial-only clustering assumptions. Our approach relies on the working hypothesis that spatial proximity serves as a proxy for data similarity. This assumption holds in many IoT settings (*e.g.,*traffic management, air-quality monitoring) but does not universally apply. In dense urban deployments, devices that are physically close may observe different phenomena (*e.g.,* indoor vs. outdoor sensors or distinct micro-environments), leading to suboptimal clustering. Similar issues arise in hospitals, where co-located devices may monitor patients with heterogeneous conditions and clinical protocols. The current implementation lacks mechanisms to detect and adapt to cases where spatial clustering does not align with data distribution patterns. Incorporating model-driven similarity metrics alongside spatial proximity could enhance the robustness of region formation, as discussed in subsequent works [DAF+24] where the potential field and leader election metrics also consider model

similarity based on cross-loss evaluation (without explicitly sharing data) leveraging a space-fluid approach [CMP+23].

Heterogeneous device capabilities. Our current approach assumes relatively homogeneous computational capabilities across devices. In practice, federated learning environments often involve devices with vastly different processing power, memory constraints, and battery life. The dynamic leader election mechanism does not currently account for device capabilities, potentially leading to resource-constrained devices being selected as aggregators, which could degrade overall system performance. However, future work could extend the leader election algorithm to consider device capabilities, ensuring that aggregators are selected based on their ability to handle the computational load.

8. Conclusions and Future Work

In this article, we presented *Field-based Federated Learning (FBFL)*, leveraging field-based coordination to address key challenges of highly decentralized and reliable federated learning systems. Our approach innovates by exploiting spatial proximity to handle non-IID data distributions and by implementing a self-organizing hierarchical architecture through dynamic leader election. FBFL demonstrates robust performance in both IID and non-IID scenarios while providing inherent resilience against node failures. Our results show that forming personalized learning subregions mitigates the effects of data skewness. Compared to state-of-the-art methods like Scaffold and FedProx, FBFL significantly outperforms in scenarios with heterogeneous data distributions.

Future work could explore advanced field coordination concepts, such as space fluidity [CMP+23], to enable dynamic segmentation of learning zones based on evolving trends in the phenomena being modeled, rather than relying on static, predefined assumptions Additionally, the potential for FBFL to support personalized machine learning models in decentralized environments offers promising applications across domains such as edge computing and IoT ecosystems. Finally, another avenue for improvement lies in integrating sparse neural networks to reduce resource consumption, making the approach more efficient and scalable for resource-constrained devices.

Acknowledgment

Lukas Esterle was supported by the Independent Research Fund Denmark through the FLOCKD project under the grant number 1032-00179B. Gianluca Aguzzi was 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

[ABB+25] Gianluca Aguzzi, Lorenzo Bacchini, Martina Baiardi, Roberto Casadei, Angela Cortecchia, Davide Domini, Nicolas Farabegoli, Danilo Pianini, and Mirko Viroli. A demonstrator for self-organizing robot teams. In Cinzia Di Giusto and António Ravara, editors, *Coordination Models and Languages - 27th IFIP WG 6.1 International Conference, COORDINATION 2025, Held as Part of the 20th International Federated Conference on Distributed Computing Techniques,*

DisCoTec 2025, Lille, France, June 17-19, 2025, Proceedings, volume 15731 of *Lecture Notes in Computer Science*, pages 230–244. Springer, 2025. doi:10.1007/978-3-031-95589-1_12.

[ACV22] Gianluca Aguzzi, Roberto Casadei, and Mirko 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*, pages 119–124. IEEE, 2022. doi:10.1109/ICDCSW56584.2022.00032.

[ACV25a] Gianluca Aguzzi, Roberto Casadei, and Mirko Viroli. Macroswarm: A field-based compositional framework for swarm programming. *Logical Methods in Computer Science*, Volume 21, Issue 3, August 2025. URL: http://dx.doi.org/10.46298/lmcs-21(3:13)2025, doi:10.46298/lmcs-21(3:13)2025.

[ACV25b] Gianluca Aguzzi, Roberto Casadei, and Mirko Viroli. Macroswarm: A field-based compositional framework for swarm programming. *Log. Methods Comput. Sci.*, 21(3), 2025. URL: https://doi.org/10.46298/lmcs-21(3:13)2025, doi:10.46298/LMCS-21(3:13)2025.

[AVD+19] Giorgio Audrito, Mirko Viroli, Ferruccio Damiani, Danilo Pianini, and Jacob Beal. On a higher-order calculus of computational fields. In Jorge A. Pérez and Nobuko Yoshida, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 39th IFIP WG 6.1 International Conference, FORTE 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17-21, 2019, Proceedings*, volume 11535 of *Lecture Notes in Computer Science*, pages 289–292. Springer, 2019. doi:10.1007/978-3-030-21759-4_17.

[CATvS17] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017. URL: http://arxiv.org/abs/1702.05373, arXiv:1702.05373.

[CMP+23] Roberto Casadei, Stefano Mariani, Danilo Pianini, Mirko Viroli, and Franco Zambonelli. Space-fluid adaptive sampling by self-organisation. *Log. Methods Comput. Sci.*, 19(4), 2023. URL: https://doi.org/10.46298/lmcs-19(4:29)2023, doi:10.46298/LMCS-19(4:29)2023.

[CPCC24] Angela Cortecchia, Danilo Pianini, Giovanni Ciatto, and Roberto Casadei. An aggregate vascular morphogenesis controller for engineered self-organising spatial structures. In *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2024, Aarhus, Denmark, September 16-20, 2024*, pages 133–138. IEEE, 2024. doi:10.1109/ACSOS61780.2024.00032.

[CPVN19] Roberto Casadei, Danilo Pianini, Mirko Viroli, and Antonio Natali. Self-organising coordination regions: A pattern for edge computing. In Hanne Riis Nielson and Emilio Tuosto, editors, *Coordination Models and Languages - 21st IFIP WG 6.1 International Conference, COORDINATION 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17-21, 2019, Proceedings*, volume 11533 of *Lecture Notes in Computer Science*, pages 182–199. Springer, 2019. doi:10.1007/978-3-030-22397-7_11.

[CVAP22] Roberto Casadei, Mirko Viroli, Gianluca Aguzzi, and Danilo Pianini. Scafi: A scala DSL and toolkit for aggregate programming. *SoftwareX*, 20:101248, 2022. URL: https://doi.org/10.1016/j.softx.2022.101248, doi:10.1016/J.SOFTX.2022.101248.

[DAEV24] Davide Domini, Gianluca Aguzzi, Lukas Esterle, and Mirko Viroli. Field-based coordination for federated learning. In Ilaria Castellani and Francesco Tiezzi, editors, *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*, volume 14676 of *Lecture Notes in Computer Science*, pages 56–74. Springer, 2024. doi:10.1007/978-3-031-62697-5_4.

[DAF+24] Davide Domini, Gianluca Aguzzi, Nicolas Farabegoli, Mirko Viroli, and Lukas 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*, pages 139–144. IEEE, 2024. doi:10.1109/ACSOS61780.2024.00033.

[DAV25] Davide Domini, Gianluca Aguzzi, and Mirko Viroli. Profed: a benchmark for proximity-based non-iid federated learning. *CoRR*, abs/2503.20618, 2025. URL: https://doi.org/10.48550/arXiv.2503.20618, arXiv:2503.20618, doi:10.48550/ARXIV.2503.20618.

[DCAV24] Davide Domini, Filippo Cavallari, Gianluca Aguzzi, and Mirko Viroli. Scarlib: Towards a hybrid toolchain for aggregate computing and many-agent reinforcement learning. *Sci. Comput. Program.*, 238:103176, 2024. URL: https://doi.org/10.1016/j.scico.2024.103176, doi:10.1016/J.SCICO.2024.103176.

[DEA+25] Davide Domini, Laura Erhan, Gianluca Aguzzi, Lucia Cavallaro, Amirhossein Douzandeh Zenoozi, Antonio Liotta, and Mirko Viroli. Sparse self-federated learning for energy efficient cooperative intelligence in society 5.0. *arXiv preprint arXiv:2507.07613*, 2025.

[DFA+26] Davide Domini, Nicolas Farabegoli, Gianluca Aguzzi, Mirko Viroli, and Lukas Esterle. Decentralized proximity-aware clustering for collective self-federated learning. *Internet of Things*, 35:101841, 2026. URL: https://www.sciencedirect.com/science/article/pii/S2542660525003555, doi:10.1016/j.iot.2025.101841.

[Dom24] Davide Domini. Towards self-adaptive cooperative learning in collective systems. In *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2024 - Companion, Aarhus, Denmark, September 16-20, 2024*, pages 158–160. IEEE, 2024. doi:10.1109/ACSOS-C63493.2024.00049.

[Est22] Lukas Esterle. Deep learning in multiagent systems. In *Deep Learning for Robot Perception and Cognition*, pages 435–460. Elsevier, 2022.

[FMO20] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/24389bfe4fe2eba8bf9aa9203a44cdad-Abstract.html.

[GCYR22] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *IEEE Trans. Inf. Theory*, 68(12):8076–8091, 2022. doi:10.1109/TIT.2022.3192506.

[GSH+24] Daniel Mauricio Jimenez Gutierrez, David Solans, Mikko A. Heikkilä, Andrea Vitaletti, Nicolas Kourtellis, Aris Anagnostopoulos, and Ioannis Chatzigiannakis. Non-iid data in federated learning: A survey with taxonomy, metrics, methods, frameworks and future directions. *CoRR*, abs/2411.12377, 2024. URL: https://doi.org/10.48550/arXiv.2411.12377, arXiv:2411.12377, doi:10.48550/ARXIV.2411.12377.

[HDJ21] István Hegedüs, Gábor Danner, and Márk Jelasity. Decentralized learning works: An empirical comparison of gossip learning and federated learning. *J. Parallel Distributed Comput.*, 148:109–124, 2021. URL: https://doi.org/10.1016/j.jpdc.2020.10.006, doi:10.1016/J.JPDC.2020.10.006.

[HYS+24] Wenke Huang, Mang Ye, Zekun Shi, Guancheng Wan, He Li, Bo Du, and Qiang Yang. Federated learning for generalization, robustness, fairness: A survey and benchmark. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(12):9387–9406, 2024. doi:10.1109/TPAMI.2024.3418862.

[KKM+20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha 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*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 2020. URL: http://proceedings.mlr.press/v119/karimireddy20a.html.

[KMA+21] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, and Arjun Nitin Bhagoji et. al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021. doi:10.1561/2200000083.

[LCB+10] Yann LeCun, Corinna Cortes, Chris Burges, et al. Mnist handwritten digit database, 2010.

[LDCH22] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *Proceedings of the International Conference on Data Engineering*, pages 965–978. IEEE, 2022. doi:10.1109/ICDE53745.2022.00077.

[LKSJ20] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *NeurIPS 2020*, 2020.

[LKZ+24] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. Vertical federated learning: Concepts, advances, and challenges. *IEEE Trans. Knowl. Data Eng.*, 36(7):3615–3634, 2024. doi:10.1109/TKDE.2024.3352628.

[LLM17] Alberto Lluch-Lafuente, Michele Loreti, and Ugo Montanari. Asynchronous distributed execution of fixpoint-based computational fields. *Log. Methods Comput. Sci.*, 13(1), 2017. doi:10.23638/LMCS-13(1:13)2017.

[LS20] Shadaj Laddad and Koushik Sen. Scalapy: seamless python interoperability for cross-platform scala programs. In Guido Salvaneschi and Nada Amin, editors, *SPLASH '20: Conference on Systems, Programming, Languages, and Applications, Software for Humanity, Virtual Event, USA, November, 2020*, pages 2–13. ACM, 2020. doi:10.1145/3426426.3428485.

[LSZ+20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze, editors, *Proceedings of the Third Conference on Machine Learning and Systems, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020. URL: https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html.

[LYG+25] Entuo Liu, Wentong Yang, Yonggen Gu, Wei Long, Szabó István, and Linhua Jiang. A survey of clustering federated learning in heterogeneous data scenarios. *Journal of Computing and Electronic Information Management*, 16(3):17–22, 2025.

[LZSL20] Lumin Liu, Jun Zhang, Shenghui Song, and Khaled B. Letaief. Client-edge-cloud hierarchical federated learning. In *2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7-11, 2020*, pages 1–6. IEEE, 2020. doi:10.1109/ICC40277.2020.9148862.

[MADB22] Yuanqiu Mo, Giorgio Audrito, Soura Dasgupta, and Jacob Beal. Near-optimal knowledge-free resilient leader election. *Autom.*, 146:110583, 2022. URL: https://doi.org/10.1016/j.automatica.2022.110583, doi:10.1016/J.AUTOMATICA.2022.110583.

[MDAV25] Nicolò Malucelli, Davide Domini, Gianluca Aguzzi, and Mirko Viroli. Neighbor-based decentralized training strategies for multi-agent reinforcement learning. In Jiman Hong, Sebastiano Battiato, Christian Esposito, Juw Won Park, and Adam Przybylek, editors, *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing, SAC 2025, Catania International Airport, Catania, Italy, 31 March 2025 - 4 April 2025*, pages 1250–1257. ACM, 2025. doi:10.1145/3672608.3707923.

[MMR+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017. URL: http://proceedings.mlr.press/v54/mcmahan17a.html.

[MZL04] Marco Mamei, Franco Zambonelli, and Letizia Leonardi. Co-fields: A physically inspired approach to motion coordination. *IEEE Pervasive Comput.*, 3(2):52–61, 2004. doi:10.1109/MPRV.2004.1316820.

[NPP+23] Dinh C. Nguyen, Quoc-Viet Pham, Pubudu N. Pathirana, Ming Ding, Aruna Seneviratne, Zihuai Lin, Octavia A. Dobre, and Won-Joo Hwang. Federated learning for smart healthcare: A survey. *ACM Comput. Surv.*, 55(3):60:1–60:37, 2023. doi:10.1145/3501296.

[NSU+18] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats 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*, pages 1–8. ACM, 2018. doi:10.1145/3286490.3286559.

[PBV16] Danilo Pianini, Jacob Beal, and Mirko Viroli. Improving gossip dynamics through overlapping replicates. In *International conference on coordination languages and models*, pages 192–207. Springer, 2016.

[PCAC24] Cèdric Prigent, Alexandru Costan, Gabriel Antoniu, and Loïc Cudennec. Enabling federated learning across the computing continuum: Systems, challenges and future directions. *Future Gener. Comput. Syst.*, 160:767–783, 2024. URL: https://doi.org/10.1016/j.future.2024.06.043, doi:10.1016/J.FUTURE.2024.06.043.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[PMV13] Danilo Pianini, Sara Montagna, and Mirko Viroli. Chemical-oriented simulation of computational systems with ALCHEMIST. *J. Simulation*, 7(3):202–215, 2013. URL: https://doi.org/10.1057/jos.2012.27, doi:10.1057/JOS.2012.27.

[RDF25] Ljubomir Rokvic, Panayiotis Danassis, and Boi Faltings. Lazy but effective: Collaborative personalized federated learning with heterogeneous data. *CoRR*, abs/2505.02540, 2025. URL: https://doi.org/10.48550/arXiv.2505.02540, arXiv:2505.02540, doi:10.48550/ARXIV.2505.02540.

[SNR20] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. Federated learning with cooperating devices: A consensus approach for massive iot networks. *IEEE Internet Things J.*, 7(5):4641–4654, 2020. doi:10.1109/JIOT.2020.2964162.

[VAB+18] Mirko Viroli, Giorgio Audrito, Jacob Beal, Ferruccio Damiani, and Danilo Pianini. Engineering resilient collective adaptive systems by self-stabilisation. *ACM Transaction on Modelling and Computer Simulation*, 28(2):16:1–16:28, March 2018. URL: http://doi.acm.org/10.1145/3177774, doi:10.1145/3177774.

[VBD+18] Mirko Viroli, Jacob Beal, Ferruccio Damiani, Giorgio Audrito, Roberto Casadei, and Danilo Pianini. From field-based coordination to aggregate computing. In Giovanna Di Marzo Serugendo and Michele Loreti, editors, *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*, volume 10852 of *Lecture Notes in Computer Science*, pages 252–279. Springer, 2018. doi:10.1007/978-3-319-92408-3_12.

[VBT17] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized collaborative learning of personalized models over networks. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 509–517. PMLR, 2017. URL: http://proceedings.mlr.press/v54/vanhaesebrouck17a.html.

[War89] Charles W. Warren. Global path planning using artificial potential fields. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, USA, May 14-19, 1989*, pages 316–321. IEEE Computer Society, 1989. doi:10.1109/ROBOT.1989.100007.

[WLL+20] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/564127c03caab942e503ee6f810f54fd-Abstract.html.

[WN21] Tobias Wink and Zoltán Nochta. An approach for peer-to-peer federated learning. In *Proceedings of the International Conference on Dependable Systems and Networks Workshops*, pages 150–157. IEEE, 2021. doi:10.1109/DSN-W52860.2021.00034.

[XGS+21] Jie Xu, Benjamin S. Glicksberg, Chang Su, Peter B. Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *J. Heal. Informatics Res.*, 5(1):1–19, 2021. URL: https://doi.org/10.1007/s41666-020-00082-4, doi:10.1007/S41666-020-00082-4.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL: http://arxiv.org/abs/1708.07747, arXiv:1708.07747.

[XYDH25] Jian Xu, Meilin Yang, Wenbo Ding, and Shao-Lun Huang. Stabilizing and improving federated learning with highly non-iid data and client dropout. *Appl. Intell.*, 55(3):216, 2025. URL: https://doi.org/10.1007/s10489-024-05956-3, doi:10.1007/S10489-024-05956-3.

[YAG+19] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan H. Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*,

volume 97 of *Proceedings of Machine Learning Research*, pages 7252–7261. PMLR, 2019. URL: http://proceedings.mlr.press/v97/yurochkin19a.html.

[YLCT19] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):12:1–12:19, 2019. doi:10.1145/3298981.