future internet

MDPI

Article

Energy-Efficient Federated Learning for Internet of Things: Leveraging In-Network Processing and Hierarchical Clustering

M. Baqer

Department of Computer Engineering, College of Information Technology, University of Bahrain, Sakhair P.O. Box 32038, Bahrain; mbaqer@uob.edu.bh

Abstract: Federated learning (FL) has emerged as a promising solution for the Internet of Things (IoT), facilitating distributed artificial intelligence while ensuring communication efficiency and data privacy. Traditional methods involve transmitting raw sensory data from IoT devices to servers or base-stations for processing, resulting in significant communication overhead. This overhead not only increases energy consumption but also diminishes device longevity within IoT networks. By focusing on model updates rather than raw data transmission, FL reduces the volume of data communicated to the base-station; however, FL still faces challenges due to the multiple communication rounds required for convergence. This research introduces an innovative approach that leverages the in-network processing capabilities of IoT devices by integrating a hierarchical clustering routing protocol with FL. This approach enhances energy efficiency through single-round pattern recognition, minimizing the need for multiple communication rounds to achieve convergence. It is envisaged that the proposed approach will prolong the lifespan of IoT devices and maintain high accuracy in event detection, all while ensuring robust data privacy.

Keywords: federated learning; internet of things; sensor networks; artificial intelligence; machine learning; distributed learning; nearest neighbor; pattern recognition; in-network processing; communication efficiency

check for updates

Academic Editors: Wei Wang, Junxin Chen and Jinyu Tian

Received: 16 November 2024
Revised: 15 December 2024
Accepted: 24 December 2024
Published: 26 December 2024

Citation: Baqer, M. Energy-Efficient Federated Learning for Internet of Things: Leveraging In-Network Processing and Hierarchical Clustering. *Future Internet* **2025**, *17*, 4. https://doi.org/10.3390/fi17010004

Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Internet of Things (IoT) is a novel paradigm that has become intertwined with the fabric of our daily life [1]. Advancement of IoT is bringing us ever closer to the realization of ubiquitous computing and pervasive networks [2–5]. Initially, IoT was proposed to refer to radio frequency identification (RFID) tags [6], but today, the IoT domain includes devices such as RFID tags, sensors, actuators, and smartphones, amongst many other devices; they have become available and ubiquitous in every aspect of our lives, and these devices are influencing our decisions, behavior, and lifestyle. Generally, IoT devices are characterized by being small hardware devices that comprise four key enabling components, namely processing, memory storage, sensor(s), and communication components [6]. There are an almost unlimited number of applications for IoT, such as surveillance, environment monitoring, e-health, smart cities, lifestyle, transportation, learning, logistics, industrial manufacturing, agriculture, military, and automobile applications amongst many others applications [6,7].

Wireless sensor networks and wireless IoT devices represent a specific subset of IoT devices. While they share many characteristics with general IoT devices, this subset faces additional challenges due to their deployment in inaccessible locations and at scales where recharging or replacing batteries is labor-intensive and inefficient. As a result, applications

and approaches tailored to these devices must prioritize energy efficiency to enhance the longevity of the network [8,9].

The capability of IoT devices to collect sensory data from the physical world and share them with other IoT devices or human users is increasingly bridging the gap between the physical and digital realms [3,10–17]. The affordability of these devices has significantly enhanced the availability and accessibility of sensory data [17].

Moreover, heterogeneous IoT devices can collaborate seamlessly, enabling machine-to-machine (M2M) interactions that occur without human intervention. This collaboration allows for the processing and aggregation of data, transforming locally collected sensory data into globally context-aware insights about the physical world [1,2,17–19]. The interoperability of these interconnected devices fosters a higher degree of intelligence within the network [1].

Nonetheless, the IoT paradigm presents new challenges, as IoT devices often face limitations in performance, resources, and cost [19]. To address these challenges, IoT applications must be efficient in utilizing the limited processing power, memory, and energy available in these devices [1]. Collected sensory data can be stored, processed, and communicated to derive meaningful insights [11]. However, storage and processing capabilities are constrained by the available resources on the IoT devices, which can be further limited by battery life and the energy required for data communication [11]. Additionally, communication within IoT networks can be unreliable and significantly slower than within traditional communication networks; thus, assuming that all nodes are avaliable and active in all scenarios many not be valid at all times [20].

Moreover, the large-scale deployment of IoT devices places substantial demands on backbone infrastructures, such as base-stations, servers, and cloud computing resources [21]. Raw sensory data must be processed to transform local observations into globally meaningful information and patterns. However, transmitting raw data through the network to centralized processing and storage can be overwhelming, especially given the vast volumes of sensory data generated by extensive IoT deployments [22]. In this context, geographically dispersed IoT devices can collaborate in parallel to optimize the use of their limited resources. Advancements in technology have made edge computing increasingly popular [23]. However, the limited resources of IoT devices often render the implementation of traditional machine learning and deep learning approaches impractical. These methods typically excel in complex environments where significant computational power, memory, and large volumes of data are available [24]. Additionally, machine learning and deep learning techniques are generally centralized, requiring both data and models to be stored at a central location [20,24]. Moreover, transferring large volumes of data to a centralized location can be communication-intensive and may compromise data privacy while also negatively impacting the lifespan of IoT devices and the overall network. As a result, traditional machine learning and deep learning approaches may not be well suited to IoT applications [25].

Federated learning (FL) and data fusion represent two distinct methodologies for managing data in IoT networks, each with unique strengths and limitations. FL prioritizes privacy and decentralization, enabling IoT nodes to locally train models and exchange only model updates. This approach reduces communication overhead and preserves sensitive data, making it well suited for energy-constrained and privacy-critical applications. Conversely, data fusion involves aggregating raw sensory data from multiple nodes at a central location, which can achieve high accuracy but at the expense of higher energy consumption, increased communication costs, and limited scalability in large-scale deployments [26–28].

A key distinction lies in data handling; FL keeps raw data local, whereas data fusion relies on transferring raw data to a central server for analysis. This centralized approach

introduces significant communication overhead, particularly in resource-constrained IoT networks. FL overcomes this challenge by transmitting lightweight model updates instead of bulk raw data, making it a more efficient option for modern IoT applications requiring distributed intelligence. While FL aligns with the needs of dynamic and heterogeneous IoT environments, data fusion remains advantageous for traditional setups where centralized processing and immediate access to raw data are essential.

Recently, FL has emerged as a promising distributed data processing solution [20,22,29–31]. Since its introduction by Google, FL has garnered significant attention for its ability to operate in a decentralized manner, primarily focusing on preserving data privacy [32]. In FL, IoT devices collaborate to create a powerful model that benefits all participants by sharing insights without compromising individual privacy. Each device can learn from other devices, thereby expanding its knowledge beyond its own limited local dataset [22]. Results have shown that the performance of FL is comparable to traditional centralized machine learning methods [25]. FL emphasizes several key aspects, including scalability, communication efficiency, device heterogeneity, statistical heterogeneity, and data privacy [20,33]. However, many machine learning architectures are resource-intensive, requiring substantial memory, processing, and communication power [22]. Additionally, traditional CPUs and GPUs may not be suitable for IoT devices, which are often untethered and battery-powered [22]. Consequently, training deep learning models on IoT devices is typically time-consuming and inefficient [22].

FL algorithms are designed to enable multiple nodes to collaboratively train a global model in a massively distributed manner [22,34]. Generally, FL decouples model training from the need to directly access raw local data, allowing nodes to maintain the privacy of their data. Instead of exchanging local data, nodes communicate model updates to the server after performing local training [20,22,25,35]. As a result, the global model is developed and refined at the server. FL approaches encounter several challenges, including the high dimensionality of model updates, persistent high communication costs, the heterogeneity of IoT devices, varying computational capabilities, varied power consumption, data quality, and data security and privacy [20,36]. Large-scale deployments of IoT devices that collect diverse sensory data can lead to significant communication costs and bottlenecks when all devices tend to transmit data to the cloud or base-station [20]. Data interoperability is another critical challenge in implementing FL within IoT networks, as each device may collect data in different structures, sizes, formats, and types [24]. Consequently, IoT devices typically refrain from sending unprocessed raw data to the base-station. Instead, sensory data are collaboratively processed among the IoT devices before being forwarded to the base-station or cloud. FL has been proposed to improve many IoT applications. In healthcare, FL can be utilized to accelerate patient diagnosis and treatment without compromising the privacy of patients. In these scenarios, patient, hospitals, and healthcare policy makers do not need to exchange their healthcare records with each other; instead, FL can train IoT devices and smartphones locally and communicate results to a server for a global consensus [37].

Seamless services such as the delivery of traffic and energy can benefit from FL to improve smart city applications [38]. FL offers smart city applications a medium for decentralized data processing that can preserve the privacy of users and IoT devices' data. A smart traffic system was proposed in [39], where smart vehicles learn an FL model based on a traffic sign dataset; the application allows selected vehicles to collaborate in a decentralized manner to recognize unlabeled traffic data while not sharing their data with third parties. Moreover, in [40] the locations of chargers are shared amongst electric vehicles without revealing private information. Videos from smartphones, bus cameras, CCTV cameras, and videos from other IoT devices are utilized in an FL video manage-

ment application, where each IoT device performs local video analytics and shares results only. Although FL and IoT have been studied extensively in recent years, their communication overhead and energy consumption implications remain unexplored. Many FL algorithms are communication-intensive and as a result are not suitable for resource-constrained, battery-operated untethered IoT devices. These limitations motivate us to suggest energy-efficient and in-network processing approaches to improve the performance of FL applications.

FL has been proposed as a transformative approach through which to enhance various IoT applications. In particular, FL is highly compatible with healthcare systems, where data privacy is paramount due to the sensitive nature of healthcare records. Patient data are often distributed across hospitals and healthcare facilities, making FL an ideal solution. By leveraging FL, patient diagnosis and treatment processes can be accelerated without compromising privacy. In these scenarios, patients, hospitals, and policy makers do not need to exchange raw healthcare records. Instead, FL enables IoT devices and smartphones to train models locally, sharing only aggregated results with a central server to achieve a global consensus [37].

Beyond healthcare, FL also benefits smart city applications, such as traffic management and energy delivery [38]. Smart city sensing is a critical component of any smart city system, and FL provides a decentralized framework for data processing while safeguarding the privacy of users and IoT devices. For example, [39] proposed a smart traffic system where vehicles collaboratively train an FL model using a traffic sign dataset. This approach allows selected vehicles to recognize unlabeled traffic data in a decentralized manner without sharing sensitive information with third parties. Similarly, [40] demonstrated how electric vehicles can share charger location data without revealing private details.

FL has also been applied to video management in smart cities. Videos captured by smartphones, bus cameras, CCTV cameras, and other IoT devices are analyzed locally, with only the results shared for further processing. In [41], a federated learning model was proposed for spatial–temporal prediction to address data heterogeneity while preserving user privacy. The authors introduced a global spatial–temporal pattern graph that integrates historical data and a personalized learning mechanism to extract heterogeneity across cities. Moreover, [42] explored FL for smart city applications, emphasizing its potential to handle complex spatial–temporal datasets while maintaining data security.

This paper presents a novel energy-efficient federated learning approach for IoT devices. It focuses on the deployment and implementation of FL in IoT environments. The proposed approach, termed TFNN, integrates the low-energy adaptive clustering hierarchy (LEACH) protocol with k-Nearest Neighbor (kNN) classification to effectively recognize event patterns. Additionally, this study introduces an intermediate level of data processing at the cluster head. This approach leverages in-network processing and storage capabilities to enhance performance and extend the longevity of the network. Instead of converging and reaching results over multiple communication-intensive rounds, this paper proposes a single-round memory recall approach. Despite the increasing popularity of FL applications, many existing methods do not fully take advantage of the in-network processing capabilities inherent in IoT networks. In this approach, instead of transmitting all data to a base-station or cloud for processing, global consensus is achieved through in-network hierarchical aggregation, where data are processed as they traverse from the device to the base-station via the cluster head.

2. Background

This paper highlights the benefits of FL, self-clustering, and in-network processing, while utilizing kNN to collaboratively recognize patterns with physically sparse IoT devices.

The following sections provide background information on FL, LEACH, kNN, and voting approaches. FL integrates with IoT networks by enabling a decentralized framework that leverages local processing capabilities at the IoT nodes. Each IoT node, equipped with minimal computational resources, trains a local model using its collected data and transmits only model updates (e.g., weights or gradients) to a central server or cluster head. This process avoids transmitting raw data, addressing both energy efficiency and privacy concerns. A typical implementation involves the following steps:

1. **Data Collection and Local Training:** IoT nodes collect data from their environment (e.g., temperature, motion) and perform local training using lightweight machine learning models, such as decision trees or small neural networks.
2. **Model Aggregation:** Cluster heads, selected using energy-efficient clustering protocols such as LEACH, aggregate model updates from nodes within their cluster. This reduces communication overhead by limiting transmissions to short distances.
3. **Global Model Update:** The aggregated updates are sent to a central base-station or edge server, which refines the global model. This model is then redistributed to IoT nodes for the next training round.

For example, in an environmental monitoring application, sensors distributed across a region can collaboratively train a model to predict air quality. Each sensor locally processes its data and sends only model updates to the cluster head, which aggregates them before forwarding to a base-station for final updates. This hierarchy ensures energy-efficient operation and scalability.

The TFNN approach further simplifies FL integration by combining in-network data aggregation and classification. By leveraging a kNN classifier at the cluster head level, TFNN enables single-round classification, bypassing the need for multiple iterations. This hierarchical design demonstrates the practical feasibility of FL in resource-constrained IoT networks. This integration ensures that sensor networks can achieve distributed intelligence while maintaining energy efficiency, privacy, and low latency, making FL a practical solution for modern IoT applications.

2.1. Federated Learning

An FL approach for iterative model averaging, known as FedAvg, was proposed in [35]. FedAvg operates using a subset of clients rather than the entire network. Each client trains its model on local data and sends the model update to a centralized server, avoiding the need to transmit raw data. The server then aggregates the received model updates through averaging to create a new global model. This process is repeated for several rounds until the convergence criterion is met.

While FedAvg decreases the data shared compared to traditional centralized approaches, it still incurs significant communication costs, as multiple rounds are required to reach consensus. Local updates may also cause FedAvg to drift from the solutions of the global objective [43–47]. Furthermore, FedAvg faces scalability challenges and may struggle to adapt to the dynamic conditions of IoT networks. Finally, its reliance on the standard arithmetic mean for model aggregation makes it vulnerable to corruption and outliers [48,49].

An FL scenario for an IoT network comprises N nodes (device) and a centralized location for final processing—a base-station or sink node. FL can be mathematically formulated as a distributed optimization problem for the global loss function f_{FL}, where $\overset{min}{\omega}$ $\frac{1}{k}\sum_{n=1}^{N}\sum_{k=1}^{k_n} f_k(\omega)$, among which ω is the global model weights and total data points stored in each device, respectively. FedAvg is being chosen to showcase the FL mathematical model [35]. As described earlier, FedAvg is based in principle on averaging the local stochastic gradient decent (SGD), as shown in Algorithm 1 [35]. The server chooses a subset

of nodes to participate in the process and distributes the recent model to the all selected nodes. Each node updates its own local model upon receiving the global model from the server and performs the SGD Algorithm. The nodes forward their updated local model to the server, and subsequently, the server develops new global models by calculating a weighted total of all obtained local models [49]. In the beginning, the global model (ω_0) is initialized. In each round (t), nodes are randomly selected by the server by controlling the fraction of clients (C) on each round to form a subset (S_t), and the current global model ω_t is distributed to all nodes in S_t. All selected nodes in S_t operate in parallel, where each node updates its own local model with the shared model $\omega_t^k \leftarrow \omega_t$. In parallel, each node computes its local gradient of descent, as shown in Algorithm 1, step 13. Clients communicate their local mode ω_{t+1}^k to the server, whereby the server then computes the weighted sum of all received local models to generate the new global model ω_{t+1}.

Algorithm 1 FedAvg Algorithm.

1: **Aggregation Server**
2: Initialize ω_0
3: **for** each round t=1,2,.. **do**
4: **for** each round t=1,2,.. **do**
5: $\omega^t \leftarrow$ ClientUpdate(i,ω_t)
6: $\omega_{t+1} \leftarrow \sum_{i=1}^{n} \omega_{t+1}$
7: **end for**
8: **end for**
9: ClientUpdate(i,ω): //Run on client i
10: **for** each local epoch e from 0,...,$E-1$ **do**
11: **for** each minibatch b of size B **do**
12: $\omega_{e+1} \leftarrow \omega_e - \eta \nabla f_i(\omega_e;b)$
13: **end for**
14: **end for**
15: return ω_e to server

In [50], the authors propose federated proximal (FedProx), a weighted aggregation approach that enhances robustness and efficiency in managing the heterogeneity of IoT devices. FedProx incorporates a proximal regularization term into the objective function, which penalizes deviations from the global model. This encourages local updates to remain close to the global model while still adapting to local data. However, FedProx has several limitations, including the following:

- **Increased Complexity**: the computational of proximal terms can lead to longer training durations.
- **Hyperparameter Sensitivity**: poor choices for the proximal term coefficient can adversely affect model performance and convergence.
- **Communication Overhead**: the approach requires multiple rounds of communication between clients and the server to reach consensus.
- **Reliance on Global Model Quality**: the effectiveness of the proximal terms is contingent on the quality of the global model.

Given these challenges, careful tuning and a deep understanding of the specific application context are essential for optimizing performance.

Federated buffered asynchronous aggregation (FedBuff) is another FL approach that employs ensemble-based aggregation [51]. FedBuff is designed to enhance efficiency and performance in scenarios with limited connectivity and variable data distribution. Instead of sending model updates to the server after every communication round, nodes can buffer several updates, reducing the frequency of communication and improving adaptability to non-IID data. Despite its advantages, FedBuff also has limitations, including the following:

- **Increasaed Latency**: buffering can introduce delays in model updates.
- **Local Memory Constraints**: nodes may face challenges due to limited memory capacity.
- **Outdated Local Models**: buffered updates may lead to stale model information.
- **Risk of Overfitting**: extended local training without server feedback can result in overfitting.

Nonetheless, it can be noticed that FL approaches suffer from high communication overhead and the need to run a large number of communication rounds (steps) to reach the desired accuracy. Improving accuracy is found to be proportional to the number of communication rounds. Additionally, in general, FL algorithms assume that nodes are deployed with direct communication access to the server. Therefore, the FL algorithm does not perform any in-network communication and thus does not take advantage of en-route processing capabilities to further improve the efficiency of FL algorithms. Table 1 compares and contrasts between the three FL algorithms mentioned in this paper, namely FedAvg, FedProx, and FedBuff.

Table 1. Comparison of FedAvg, FedProx, and FedBuff federated learning algorithms.

Feature	FedAvg	FedProx	FedBuff
Proposed by	McMahan et al., 2017 [35]	Li et al., 2020 [50]	Nguyen et al., 2022 [51]
Short Description	Averages local model updates from clients to train a global model.	Modifies the optimization objective to handle non-IID data and client heterogeneity. Adds a proximal term to the objective to mitigate heterogeneity issues.	Introduces buffered aggregation to optimize communication and adapt to dynamic environments. Utilizes buffered aggregation to optimize communication and handle client variability.
Number of Rounds/Trips	Moderate to high; struggles with non-IID data and slow convergence in heterogeneous setups.	Fewer rounds due to buffering mechanisms mitigating asynchronously and stragglers.	Similar to FedAvg; may slightly reduce rounds by addressing heterogeneity through regularization.
Accuracy	High on IID data; decreases with non-IID data due to unbalanced contributions.	Improved over FedAvg in non-IID settings due to better synchronization.	Better than FedAvg for non-IID data; uses proximal terms to handle data heterogeneity effectively.
Scalability	Scales well in ideal conditions but sensitive to communication and device heterogeneity.	Better scalability; asynchronous updates and buffering reduce overhead.	Similar to FedAvg; slightly more computation due to proximal term but handles heterogeneity better.

The authors in [51] introduced FedBuff, a federated learning algorithm that enhances communication efficiency and robustness in asynchronous settings by utilizing a buffering mechanism for aggregating client updates. The method is particularly effective in federated learning scenarios with 100 devices where client updates may be delayed or asynchronous. By storing and aggregating updates in a buffer, FedBuff allows for more efficient synchronization and reduces the number of rounds needed to achieve convergence. In their experiments with non-IID data on the CIFAR-10 dataset, FedBuff required around 250–300 rounds to reach 50% or above accuracy, which is faster than FedAvg, which needed more than 400 rounds in similar settings. This improvement is due to the buffering technique that mitigates the impact of straggling clients and varying client latencies.

In comparison, FedProx, another federated learning method designed to handle data heterogeneity, stabilizes training by adding a proximal term to the loss function, penalizing updates that diverge too far from the global model. While FedProx generally requires more computational effort due to the proximal regularization, it improves accuracy in non-IID settings by providing a more stable convergence path. FedProx often reaches 50% or more accuracy in 300–350 rounds for non-IID data, making it slower than FedBuff but generally more stable in highly heterogeneous environments. Therefore, while FedBuff excels in communication efficiency, FedProx offers robustness to client and data heterogeneity at the cost of additional computational overhead.

This study introduces an intermediate processing level where data are handled in-network at the cluster head level, aiming to reduce communication requirements and the number of rounds needed for convergence. As mentioned earlier, FL is a decentralized approach that emphasizes local data processing, enabling model updates to be exchanged among devices without sharing raw data [24]. FL operates on data that are often characterized as unbalanced, massively distributed, and non-IID [24]. Typically, FL training and recognition processes occur at two primary levels: the node level and the cloud level (or base-station level). Many studies emphasize the importance of enhancing edge (node-level) processing to decrease the number of communication rounds required [35,52,53]. In mesh ad-hoc network environments, where IoT devices can communicate with one another, leveraging in-network processing capabilities can further minimize the amount of data transmitted across the network, as demonstrated in this study.

2.2. LEACH

LEACH is a cluster-based routing protocol designed for sensor networks that utilizes in-network hierarchical routing and incorporates three main concepts: (i) localized coordination and control for cluster setup and operations, (ii) rotation and randomization of cluster heads, and (iii) data aggregation, as described in Algorithm 2 [54]. By forming clusters, LEACH enhances the longevity of the network through in-network aggregation, enabling close-range communication among nodes within a cluster while reserving long-range communication for interactions with the base-station. Additionally, the adaptive clustering and rotation of cluster heads help to balance node utilization across the sensor network, thereby improving overall energy efficiency.

Algorithm 2 k-NN Algorithm.

Input
D: a set of training samples $\{(x_1,y_1),\ldots,((x_n,y_n)\}$
k: the number of nearest neighbors
for each training sample $(x_i,y_1) \in D$ **do**
    Compute $d(x,x_i)$
end for
let $N \subseteq D$ be the set of training samples with the k smallest distance
return the majority lable of the sample in N

LEACH operates under the assumption that all nodes start with the same energy level and that functioning as a cluster head consumes approximately the same amount of energy for each node. This sensor network approach works well for IoT applications characterized by a large number of nodes with limited energy resources, particularly in scenarios where data transmission distances can be minimized and where periodic data monitoring is required to maintain efficient and reliable operations.

The operation of LEACH can be divided into two phases: the setup phase and the steady phase. During the setup phase, the network is organized into non-overlapping clus-

ters, each with its own cluster head. LEACH maintains dynamic formation and selection of clusters and cluster heads. The selection of cluster heads occurs independently across the network, utilizing a mechanism that prevents any cluster head from being permanently assigned (i.e., static). Each node generates a random number (n) between 0 and 1, and nodes with an n value lower than a predefined threshold function $T(n)$ are selected as cluster heads. This approach ensures that each node has the opportunity to become a cluster head at least once. The threshold function $T(n)$ is computed as follows:

$$T(n) = \begin{cases} \frac{p}{1-p(r \mod \frac{1}{p})} & n \in G \\ 0 & \text{others} \end{cases} \quad (1)$$

In this context, G represents the set of nodes that have not served as cluster heads in the last $\frac{1}{p}$ rounds, while r denotes the current round, and p indicates the desired percentage of cluster heads. The threshold $T(n)$ ensures that once a node is selected as a cluster head, it will not be chosen again for the next $\frac{1}{p}$ rounds.

The subsequent phase, known as the cluster head advertisement phase, begins when the selected cluster heads use a CSMA MAC protocol to announce their presence to neighboring nodes, prompting non-cluster head nodes to activate their receivers. Once this phase concludes, non-cluster head nodes select the cluster to which they will belong for the current round, taking into account the signal strength of the cluster head's advertisement. Choosing the cluster head with the strongest signal is believed to minimize the energy required for communication within the cluster. In cases of a draw, the selection of a cluster head is resolved randomly. Furthermore, each node informs the cluster head of its intention to join the cluster, marking this stage as the setup phase.

The following stage is the steady stage, during which each cluster head develops a TDMA schedule that allows cluster members to transmit their data and aggregate the received information. As a result, LEACH facilitates the aggregation of data collected from IoT devices in a distributed manner without intervention from the base-station. This approach reduces energy communication and distributes energy consumption among the nodes, thereby enhancing the network's overall lifetime.

2.3. k-Nearest Neighbors

Since Fix and Hodges introduced the non-parametric method for pattern classification known as kNN, its strengths have become evident [55,56]. kNN is valued for its simplicity, lack of a formal training phase, ability to handle a large number of classes, avoidance of parameter over-fitting, and reliance on distance measurement [55]. In kNN, the training process can be viewed as a single-shot memorization phase, contrasting sharply with the computationally intensive training required for neural networks.

kNN assumes that all instances correspond to points in the n-dimensional space $\mathbb{R}^n$. The nearest neighbors of an instance can be computed in by using any of the popular distance measurements, such as Euclidean distance. When f is the true concept function that gives the correct class value $f(x)$ for each instance x, and instance x can also be $< a_1(x), a_2(x), \ldots, a_n(x) >$ and $a_r(x)$ denote the value of the rth attribute of instance x. We can describe an arbitrary instance by vector $< x, f(x) >$. The Euclidean distance between two instances x_i and x_j is denoted as $d(x_i, x_j)$ and can be calculated by Equation (2).

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \quad (2)$$

The kNN algorithm is described by the pseudo-code in Algorithm 2. kNN is a supervised learning algorithm, where a training set D is employed to train the model to

recognize patterns. Each instance in D comprises a pair of (x_i, y_i) which are a pattern and its label. K represents the number of nearest neighbors to be considered when labeling the test patterns. The algorithm computes $d(x, x_i)$ for each training sample (x_i, y_i), and the algorithm calculates the distance between the target sample x (the sample for which we want to determine the majority label) and the current training sample x_i. The specific distance metric may use any of the distance measurement methods, such as Euclidean distance and Manhattan distance, amongst others. After computing the distance between the target sample x and each training sample x_i, the algorithm selects the k samples from D that have the smallest distances. These k samples are from the set N, which represents the k^{th} nearest neighbors of the target sample x. Finally, the algorithm determines the majority label among the samples in N. The majority label is the label that appears most frequently among the k^{th} nearest neighbors. This majority label is then returned as the output of the algorithm.

In the next section, we propose encapsulating the aforementioned concepts in a tiny federated nearest neighbor approach to provide decentralized energy-efficient event recognition for wireless sensor networks.

3. Tiny Federated Nearest Neighbor

This paper proposes federated nearest neighbor (TFNN), an approach which aims to provide an energy-efficient FL solution for sensor networks, taking into account the memory, processing, and communication constraints of IoT nodes. Instead of forwarding raw sensory data to a central base-station for processing, the network utilizes in-network processing to analyze data and develop models locally. This approach proposes significantly reducing the energy consumed by minimizing direct transmissions of raw sensory data to the base-station. TFNN leverages the LEACH protocol, enabling IoT devices to autonomously form clusters and elect cluster heads. Each cluster head then aggregates the sensory data from its associated nodes to update the model. Additionally, this decentralized approach means that the base-station is not required to manage the network or monitor the status of active and inactive IoT nodes directly.

While kNN can work with both types of data, its performance is generally more predictable and effective with IID data. With non-IID data, careful consideration of data characteristics and distance metrics is essential to optimize its performance [57]. Algorithm 3 outlines TFNN approach, wherein IoT devices collaboratively transform their local data into a unified global event recognition framework through a decision fusion process. Each IoT device in the network retains training patterns and executes kNN locally on the test (event) patterns, as detailed in Algorithm 2. In this local context, the class label for the event (test) pattern x is established through majority voting among the IoT devices. When an event is detected, each IoT device independently executes kNN, thereby contributing to the collective classification process. Each IoT node will find the k^{th} closest training pattern to the event pattern and then perform majority voting to find the winning class label. Let $f_i(x)$ be the class label of the i^{th} neighbor of x. The identity function $\delta(c,f_i)$ output is 1 if a match is found $(f_i\ (x) = c)$, and otherwise 0. The function $g(c) = \sum_i \delta(c, f_i(x))$ is used to count the neighbors with label c. The winning class will be the class label receiving the majority of votes (i.e., 50% + ϵ) using $c^* = argmax_c g(c)$.

As a result, each IoT node will classify the event pattern and find the best matching class label from the training set. In this local context, the class label for the event (test) pattern x is determined through majority voting among the IoT devices. When an event is detected, each IoT device independently executes kNN, contributing to the overall classification process. Each IoT device identifies the Kth closest training pattern to the event pattern and then performs majority voting to establish the winning class label.

Next, majority voting is conducted within each cluster to aggregate the local class label results from all IoT devices belonging to that cluster. Each IoT device transmits its winning class label to the cluster head, which then processes the collected labels through majority voting to update the voting model, as described earlier. Finally, each cluster head communicates its aggregated model (the results from the cluster) to the base-station. The base-station performs a final model $(\hat{y}_f)$ update by applying majority voting to the received models from the cluster heads, resulting in the winning class label for the event pattern P_t.

TFNN can benefit from utilizing a blockchain-based federated learning framework (FedCFB) to improve security, particularly data integrity and trustworthiness [58]. Traditional FL faces challenges like model poisoning, data heterogeneity, and free-riding behavior, which undermine its reliability. FedCFB addresses these by integrating blockchain technology to ensure tamper-proof storage of data and models, along with smart contracts for secure and automated processes. The framework employs a multi-stage approach, including malicious model detection, fair client contribution evaluation using an improved Shapley Value method, and a decentralized architecture for secure aggregation and reward distribution. The study in [58] shows that FedCFB is able to maintain model accuracy and fairness, even under adverse conditions. Its use of blockchain ensures accountability and transparency in client contributions and reward allocation, making it particularly suitable for applications requiring secure and collaborative data processing.

4. Evaluation

Energy consumption, delay and throughput, and accuracy are the main factors to be considered when evaluating algorithms for IoT. The following subsections describe energy consumption, delay and throughput, and accuracy models.

4.1. Energy Consumption

Energy consumption is the most significant constraint in IoT networks, directly affecting both operational efficiency and network longevity. In particular, communication processes are the largest consumers of energy. The TFNN approach enhances the lifespan of the network by incorporating energy-efficient techniques from the LEACH protocol. Organizing the network into clusters with designated cluster heads and randomizing their selection, energy consumption is balanced across all nodes. Moreover, processing raw data at the local node level and aggregating results at the cluster level reduce the amount of data communicated to the base-station. It is important to consider that the topology and placement of IoT nodes, as well as their distance from the base-station, significantly influence energy usage for data transmission. In TFNN, the locally processed sensory data are analyzed using kNN at the node level, after which each cluster head collects and aggregates the class label results from the associated IoT nodes before forwarding them to the base-station. Therefore, the energy consumption related to data communication can be understood by taking into consideration two key aspects, transmission of data (E_{Tx}) and receiving data (E_{Rx}), as in the following formulas:

$$E_{Tx}(m, d) = (E_{elec} * m) + (\epsilon_{amp} * m * d^2) \tag{3}$$

$$E_{Rx}(m) = E_{elec} * m \tag{4}$$

where E_{elec} represents the energy dissipation in Joules (J) per bit (b) required to operate the wireless transmitter circuitry, typically consuming 50 nJ/b. Additionally, the energy consumed by the transmitter amplifier generally consumes 100 pJ/m^2 during the receiving waiting period, which is required to achieve an acceptable $\frac{E_b}{N_q}$ ratio. To transmit m-bits

message over a distance d, we can use Formula (3). Moreover, Formula (4) illustrates that receiving messages incurs non-negligible energy costs.

The cost of communication is significantly influenced not only by the distance between nodes but also by the size (number of bits per message (m)) and the total number of transmitted messages. The clustering achieved by the LEACH protocol helps minimize the number of messages sent directly to the base-station, as it restricts direct communication to cluster heads only. Consequently, the total energy consumed for communication from the network to the base-station, as represented in Formula (3), can be expressed as $\sum_{n=1}^{CH} E_{Tx}(m_n, d_n)$. Additionally, the clustering algorithm depends on the proximity of nodes to each other, and as a result, the majority of the communications will occur within clusters with small distances (d), which will consequently affect the overall energy E_{Tx} calculated by Formula (3). Each cluster head will need to wait and listen for n node transmissions, and each node will consume communication energy when sending its local result to the cluster head. Therefore, the energy consumption of $cluster_i$ is affected by the number of nodes associated with each cluster and derived from Formulas (3) and (4) as follows:

$$E_{Rx_cluster_i} = n_{cluster_i} * E_{elec} * m \tag{5}$$

$$E_{tx_cluster_i}(m, d) = \sum_{i=1}^{n \in cluster_i} ((E_{ele} * m) + (\epsilon * m * d_i^2)) \tag{6}$$

The waiting energy for the i^{th} cluster head refers to the time required for n nodes in cluster i to transmit their messages to the cluster head. In Formula (6), we assume that each node sends messages of the same size m, and that the distance between the cluster head and the i^{th} node is denoted as d_i.

4.2. Delay and Throughput

In addition to energy efficiency, delay and throughput are critical factors impacted by FL in IoT networks. FL can introduce delay due to its iterative communication rounds, where each node n performs local training and sends model updates to a central base-station server. The total delay can be expressed as $T \times R$, where T represents the transmission delay for each round, and R is the total number of communication rounds required for convergence. TFNN is an optimized single-shot approach which can significantly reduce this delay by achieving model consensus in one round.

Throughput is defined as the effective amount of data processed per unit time and is often improved in FL. This is because FL reduces the data payload from raw data D_{raw} to smaller model updates D_{model}, such that $D_{\text{model}} \ll D_{\text{raw}}$. This reduction minimizes network congestion and enhances bandwidth utilization B. The throughput can be approximated as follows:

$$T_p = \frac{D_{\text{model}}}{\Delta t}, \tag{7}$$

where Δt is the transmission time per update. Throughput and delay performance are key aspects, where FL and TFNN differ significantly. In FL, throughput is improved by reducing the data payload from raw data D_{raw} to smaller model updates D_{model}, such that $D_{\text{model}} \ll D_{\text{raw}}$. This leads to efficient utilization of bandwidth B. However, FL typically requires multiple communication rounds (R) for model convergence, leading to cumulative delays proportional to $T \times R$, where T is the transmission delay per round.

In contrast, TFNN addresses this delay by leveraging single-round hierarchical data aggregation and decision making. Using clustering protocols like LEACH, TFNN combines local data within clusters and performs kNN classification in a single communication round.

This minimizes the total delay to approximately $T_{\mathrm{TFNN}} = T$, which is significantly lower than the multi-round delay in FL.

4.3. Accuracy

The second factor to consider when evaluating the TFNN algorithm is recognition accuracy. As demonstrated in Algorithm 3, the TFNN algorithm performs federated kNN among the nodes in the network. At the local node level, the accuracy of each node's kNN can achieve performance better than the Bayes error rate. Formula (7) shows the theoretical minimum error, which represents the best performance that any classifier can achieve when the underlying probability distribution of the class is known; E for kNN can be estimated as follows:

$$E = \int \min_{c \in C} P(C = c|X = x)P(X = x)dx \tag{8}$$

Formula (7) shows the integral sum over the entire input space, where C is the set of possible classes, $P(C{=}c \mid X{=}x)$ is the posterior probability of class c given the feature vector x, and $P(X \mid x)$ is the probability density function of the feature vector x. Furthermore, Formula (7) demonstrates that the error rate does not exceed twice that of the Bayesian optimal classifier.

Algorithm 3 Tiny federated nearest neighbor (TFNN) algorithm.

Network Initialization
$N \leftarrow \{n_1, n_2, n_3, \ldots n_k\}$
$(CH,CLUSTER) \leftarrow \{(ch_1, cluster_1),(ch_2, cluster_2),(ch_3, cluster_3), \ldots, (ch_j, cluster_j)\}$
% Autonomously partition the network by forming j non-overlapping clusters and elect a cluster head (ch_i) for each cluster ($cluster_i$) using LEACH
$cluster_i \subseteq N$, where $cluster_i \cap cluster_k = \phi$ for all $1 \leqslant i \neq k \leqslant j$
Base-Station Aggregation
% each cluster aggregates the local kNN results of nodes associated with the cluster
for every cluster head $\mathrm{ch}_i \in \mathrm{CH}$ **do**
    $\hat{y}_i^{ch_i} \leftarrow$ **ClusterHead_Update**(ch_i, P_t)
end for
$\hat{y}_f \leftarrow$ **argmax** $\sum_m^{i=1} \delta(c, \hat{y}_i^{ch_i})$

ClusterHead_Update(*ch*,*P*)
for each node $\mathrm{n}_i \in \mathrm{cluster}_i$ **do**
    $\hat{y}^j = \mathrm{NodeKNN}(P)$ % return the winning class label
end for
return $\mathrm{argmax}_c \sum_{j=1}^{m} \delta(c, \hat{y}^j)$

TFNN employs majority voting to aggregate local classifications and determine a global label for the event pattern. Each node n in cluster i has an individual accuracy p, where $0 \leqq p \leqq 1$. In this majority voting system, an event pattern is classified when more than half of the nodes assign the correct label. Consequently, the accuracy A of the majority voting can be calculated as follows:

$$A = \sum_{n=1}^{k} \begin{pmatrix} n \\ k \end{pmatrix} * p^k * (1-p)^{(n-k)} \tag{9}$$

where k is the number of nodes with the correct class label, and n is the total number of nodes. In majority voting, the condition $k > \frac{n}{2}$ must be met. The term $\begin{pmatrix} n \\ k \end{pmatrix}$ represents the binomial coefficient, which calculates the number of ways to choose k nodes with

correct votes from n voting nodes. Consequently, the error of the majority vote (E) can be calculated as follows:

$$E = 1 - A = \sum_{n=1}^{k} \binom{n}{k} * p^{n-k} * (1-p)^k \tag{10}$$

Formulas (8) and (9) facilitate the calculation of the accuracy and error of the majority vote based on the individual voting accuracy p of each node and the total number of voting nodes n. These formulas apply under the assumption of two label classes, meaning that nodes can either vote for 0 or 1. For multi-class majority voting, we consider a scenario with c classes, where $c > 2$. The accuracy A of the majority voting can be expressed as follows:

$$A = \sum_{k_1+k_2+\ldots+k_c=n}^{k} \binom{n}{k_1, k_2, \ldots, k_c} \prod p_i^k \tag{11}$$

Let $\sum_{i=1}^{c} k_i \leq \frac{n}{2}$, and the value of A is calculated using multinomial coefficients, which represent the number of ways to choose k_i voting nodes for each class i, ensuring that the total sum of all k_i is equal to n. The multi-class error (E) of the majority vote can be expressed as follows:

$$E = 1 - A \tag{12}$$

The number of voting nodes (n) in the majority voting model significantly influences both A and E in selecting the correct class label. As the number of voting nodes increases, the probability of the majority voting accurately also rises. In fact, the accuracy of majority voting is directly proportional to the number of participating nodes. Consequently, a greater number of nodes enhances the likelihood that the majority will make the correct decision.

Conversely, E is inversely proportional to the number of participating nodes. Thus, with more voters involved in the process, the probability of an incorrect majority vote decreases. This holds true for both binary and multi-class classifications; as the number of voting nodes (n) increases, the terms involving n in the binomial and multinomial coefficients grow, resulting in higher A and lower E for the majority vote.

The number of nodes participating in the voting process significantly influences A and E of the majority vote. Consequently, confidence (α) can be modeled as a function of the number of voting nodes (n), which helps to quantify the certainty of the majority vote result. For binary classification, the value of α can be determined using the following formula:

$$\alpha = 2 * (1 - \varphi(\left|\frac{n}{2} - k\right|, n, p)) \tag{13}$$

where φ is the cumulative distribution function (CDF) of the standard normal distribution, k represents the number of voters for the majority class, n is the total number of voting nodes, and p denotes the individual accuracy of the voting nodes. The value of α ranges from 0 to 1. Similarly, the formula for multi-class classification can be derived as follows:

$$\alpha = 2 * (1 - \varphi\left|\frac{n}{2} - \sum k_i\right|, n, \sum p_i)) \tag{14}$$

where k_i is the number of voting nodes for class label i and p is the individual accuracy of the voting nodes for that class. The value of α represents the confidence level, which is determined by how close the sum of the voters for the majority class is to half the total number of voters. Consequently, the further the sum is from $\frac{n}{2}$, the higher the confidence. The following updates to Formulas (9)–(13) incorporate this confidence as follows:

$$\hat{A} = A * \alpha \tag{15}$$

$$\hat{E} = E * (1 - \alpha) \tag{16}$$

TFNN may not utilize all the nodes in the IoT network. Therefore, the following formula helps us to understand the accuracy achieved when using the entire network compared to the accuracy obtained when utilizing only a subset of nodes:

$$A(N) = (1 - \varphi(\left|\frac{N}{2} - k\right|, N, 0.5)) \tag{17}$$

$$A(n) = (1 - \varphi(\left|\frac{n}{2} - k\right|, n, 0.5)) \tag{18}$$

For multi-class classification, the following formulas are applied, where $k_{s,i}$ is the number of votes for class i in the sample n.

$$A(N) = 1 - \varphi(\left|\frac{N}{2} - \sum k_i\right|, N, \frac{1}{c})) \tag{19}$$

$$A(n) = 1 - \varphi(\left|\frac{n}{2} - \sum k_{s,i}\right|, n, \frac{1}{c})) \tag{20}$$

where N is the set of all nodes in the network and $n \subseteq N$. For both binary and multi-class classification, the difference between using all nodes and using a subset of nodes is shown as follows:

$$A_{diff} = A(N) - A(n) \tag{21}$$

Equation (20) shows that as the sample n increases, $A(n)$ will approach $A(N)$, and the difference between the accuracy of using a sample subset and all nodes will diminish.

5. Results

This study is motivated by the recognition of event patterns that may occur in nature. TFNN is designed as a robust algorithm that autonomously adapts to the diverse dynamics of IoT networks without human intervention, while enhancing the network's longevity in recognizing event patterns. This study focuses on exploring the energy consumption and accuracy of the TFNN algorithm.

The Modified National Institute of Standards and Technology (MNIST) digit dataset was utilized, where each IoT device in the network is trained on a selection of training patterns from this dataset [59]. The MNIST dataset consists of a large collection of handwritten digits ranging from 0 to 9 and is categorized into two main subsets: training and test sets. A sample of handwritten digits is shown in Figure 1. In this simulation, the training dataset is partitioned, with each IoT device receiving equal number of random, non-overlapping set of training patterns corresponding to the selected digit. During the simulation, test patterns are randomly selected from the MNIST test set, and the same pattern is presented to all IoT nodes in each test epoch (round).

TFNN is simulated using MATLAB, and its performance is compared to a scenario where any node in the network communicates its local results directly to the base-station [60]. The simulation utilizes the LEACH algorithm, which organizes the network topology by selecting cluster heads and forming clusters, while each active node implements the kNN algorithm to recognize event patterns locally.

Additionally, the simulation considers communication performance, allowing active nodes within a cluster to communicate directly with their cluster head, while nodes that fail to join any cluster can communicate directly with the base-station. Each cluster head aggregates the local kNN results from its members using majority voting; thus, cluster members are not permitted to communicate directly with the base-station. Subsequently, each cluster head relays its winning pattern class label to the base-station. At the base-station, a second

round of majority voting is performed, where the collected class labels from active nodes not belonging to any cluster, along with those from the cluster heads, are aggregated to determine the final winning class label for the event pattern. The simulation assumes a field size of 100 m × 100 m, with the base-station located outside the field at coordinates (50, 200). The base-station is positioned significantly far from the network to amplify the impact of communication to the base-station in this simulation setup. Other nodes are deployed randomly, as illustrated in Figure 2. The number of IoT nodes deployed can be adjusted to simulate networks of varying sizes, specifically utilizing two configurations in this study: 50 and 100 nodes. In the case of a network of 50 IoT nodes, depicted in Figure 2, the setup includes the base-station and 50 IoT nodes organized into clusters of varying sizes. In each round, the clusters are re-created according to the LEACH algorithm, resulting in a different number of clusters of various sizes in each round.

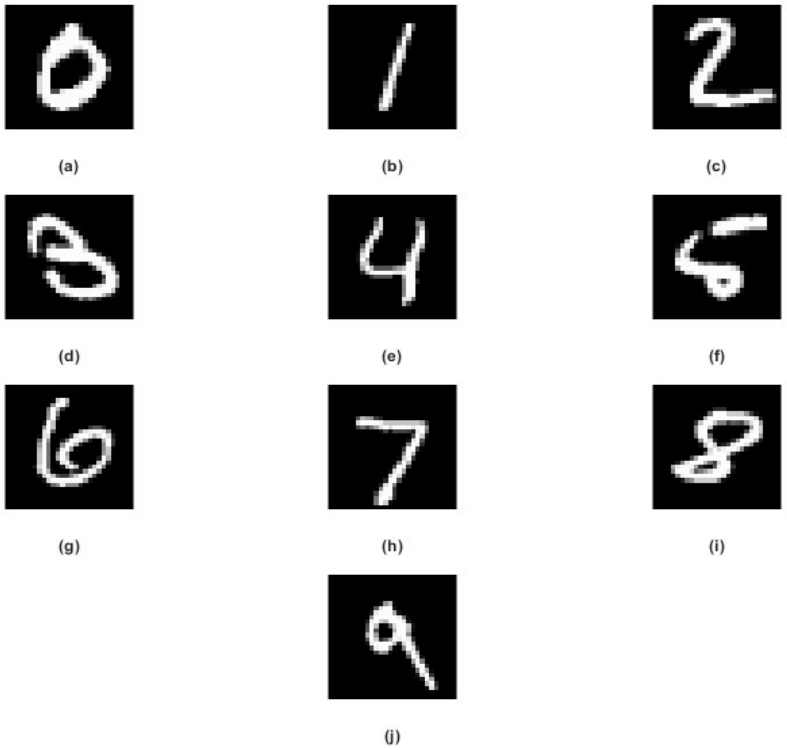


Figure 1. Figures (**a**–**j**) present a selection of handwritten digits from 0–9 of the MNIST dataset, respectively.

As previously mentioned, communication energy is the primary consumer of energy resources for IoT nodes. This energy consumption is primarily associated with waiting to receive messages, waiting for a node's turn to communicate with the destination, and the energy cost of transmitting messages. Two key factors significantly influence communication energy: the number of communicated bits and the distance from the node to the destination. These factors are incorporated into the simulation to evaluate performance. In the simulation, the base-station is assumed to be connected to a continuous energy source, while all other nodes and cluster heads are equipped with equal amounts of energy, utilizing the same type of battery. In each round, the energy consumption of each node is calculated, and each node ceases to operate once its energy is depleted.

TFNN enhances the network's lifetime compared to the direct-to-base-station communication approach. The autonomous hierarchical clustering configuration enabled by TFNN allows cluster heads to communicate directly with the base-station, while all other communications occur within the cluster. Members of each cluster send their local results to their cluster head for processing. Each cluster consists of nodes that are in close proximity to

one another, resulting in generally short distances between the cluster head and its cluster members. In contrast to the direct communication approach, which requires long-distance, energy-intensive transmissions to the base-station, such communications are limited to cluster heads and nodes that are not part of any cluster. By ensuring that the majority of data communications occur within the cluster, TFNN significantly reduces energy consumption among the nodes. This reduction prolongs the network's operational lifespan, allowing it to detect patterns for longer periods without the need to replace or recharge batteries. Figure 3 illustrates that the number of active nodes experiences a significant drop around round 400. However, the network continues to operate and detect patterns, with the number of active nodes gradually declining after round 800 when using TFNN. Consequently, TFNN nearly doubles the network's lifetime.

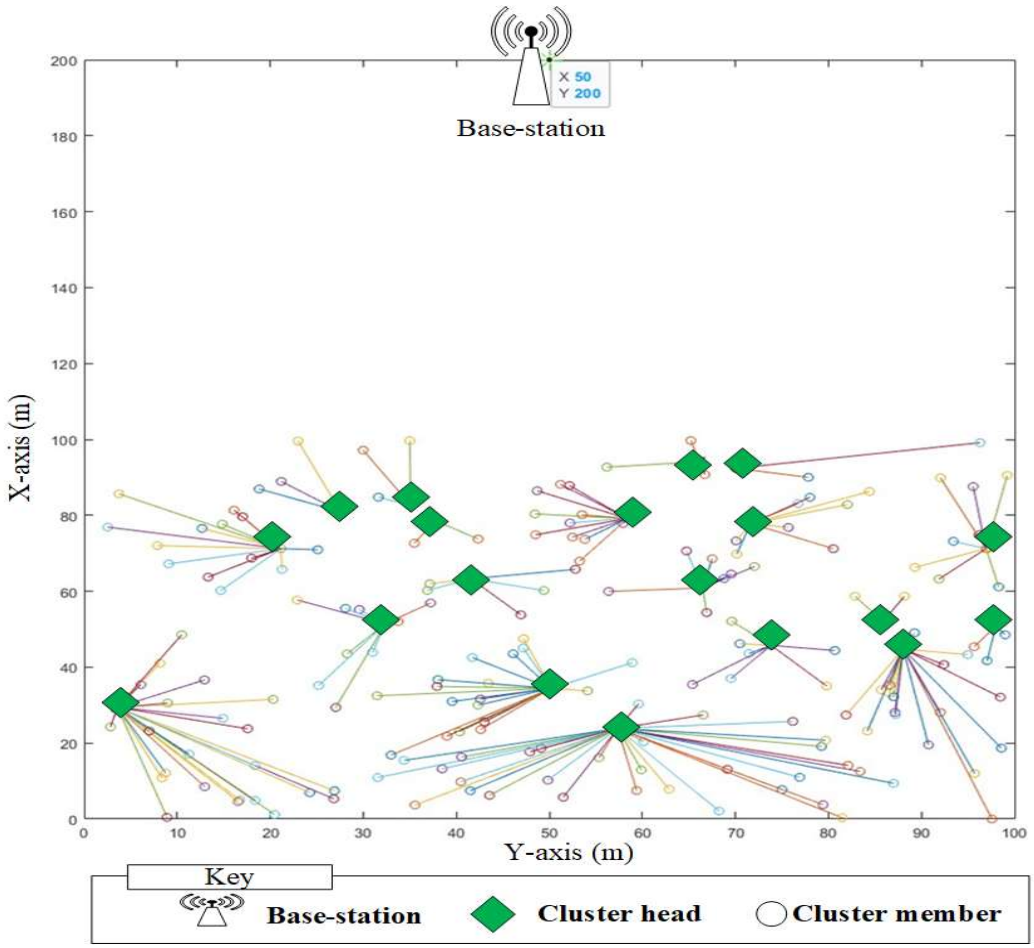


Figure 2. Network setup featuring 50 nodes deployed within a 100 m × 100 m field, with the base-station positioned at (50, 200). In this round, 19 clusters are formed.

The simulation results presented in Figure 4 establish a connection between the accuracy and lifetime of the network. The figure compares the accuracy of event recognition against the network's lifetime (measured in rounds) while testing MNIST patterns from 0 to 9 using both TFNN and direct communication approaches. Each node was trained with 200 random patterns from the MNIST training dataset. The results indicate comparable performance across different test patterns. However, when direct communication is employed instead of TFNN, accuracy begins to decline around round 400 for all pattern classes. The above figure demonstrates that nodes utilizing direct communication deplete their battery power more quickly than those using TFNN, as direct communication to the base-station is more energy-intensive. Consequently, the accuracy of nodes using direct

communication decreases rapidly as they start to go offline due to battery depletion, which is exacerbated by the high energy consumption associated with direct communication.

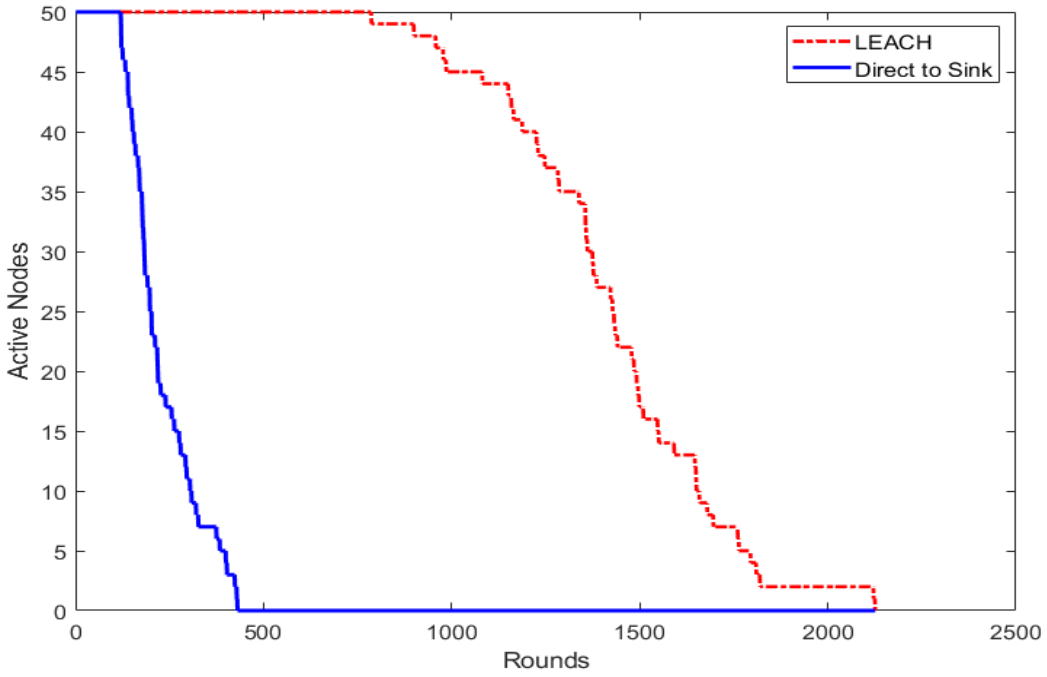


Figure 3. Number of active nodes while using TFNN and direct-to-sink communications.

FedAvg, FedProx, and FedBuff rely on multiple communication rounds to detect patterns and achieve specific accuracy thresholds. According to the study in [51], these algorithms require an average of 231, 228, and 31.9 rounds, respectively, to detect each pattern. This multi-round process significantly impacts the network's lifetime, as each communication round consumes resources such as battery power, memory, and processing capacity. Consequently, these algorithms may be unsuitable for deployment in IoT networks with strict resource constraints.

The communication-intensive nature of these algorithms suggests that an IoT network employing any of them would be capable of detecting only a limited number of patterns before depleting its energy reserves and going offline. Based on the required number of communication rounds, such a network may operate for only a short duration before the nodes exhaust their battery power and cease functioning. Integrating LEACH with FedAvg, FedProx, or FedBuff could mitigate this issue by reducing the frequency of direct communication with the base-station, thereby conserving energy.

The communication-intensive nature of these algorithms suggests that an IoT network employing any of them would detect only a limited number of patterns before depleting its energy reserves and going offline. Based on the required number of communication rounds, such a network may operate for only a short duration before nodes exhaust their battery power and cease functioning. Integrating LEACH, a hierarchical clustering routing protocol, with FedAvg, FedProx, or FedBuff could help reduce the frequency of direct communication with the base-station, conserving energy and potentially improving network longevity.

However, these algorithms still rely on iterative model updates distributed over multiple communication rounds to recognize each pattern, which can exacerbate resource consumption in resource-constrained IoT networks. Moreover, evaluating how hierarchical routing protocols like LEACH influence fairness in model training and data privacy preservation within federated learning could provide deeper insights into their effectiveness. In contrast, the single-shot TFNN approach requires fewer communication rounds, recog-

nizing patterns in each round and inherently preserving energy. Its consistent performance across rounds also highlights its potential as an energy-efficient and privacy-conscious alternative to traditional FL algorithms.

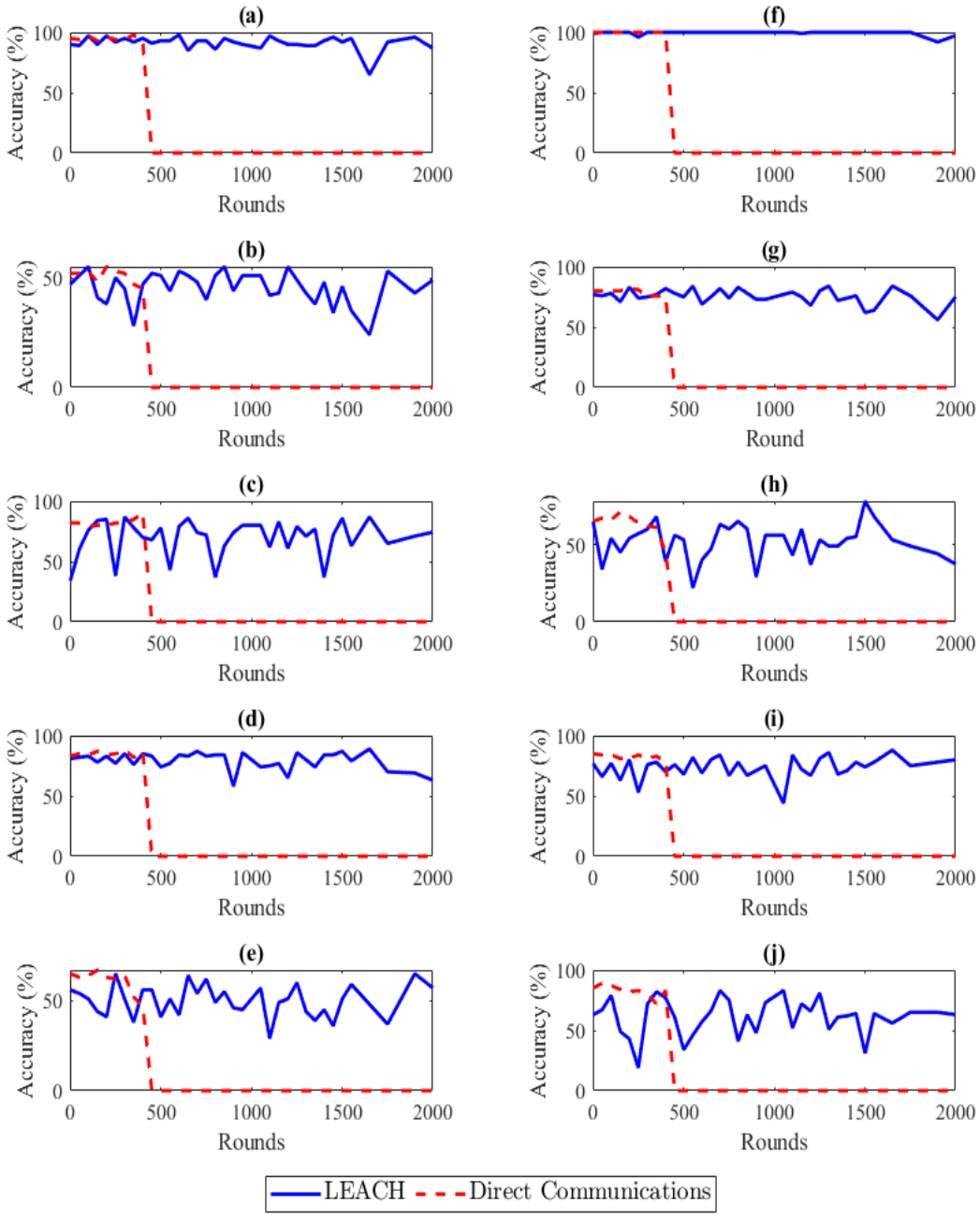


Figure 4. Accuracy of event recognition when using LEACH in comparison to direct communications to the base-station, while setting $K = 5$ for handwritten digits 0–9, as shown in sub-figures (**a**–**j**), respectively.

In contrast, TFNN offers a single-shot approach to pattern detection. Results from Figure 3 demonstrate that an IoT network using TFNN remains active for 800 rounds before experiencing any node failures. Unlike the multi-round FL algorithms, TFNN recognizes a pattern in each round, making it significantly more efficient. Furthermore, the pattern recognition performance of TFNN remains consistent across a large number of rounds, as its built-in energy efficiency techniques extend the network's operational longevity and enhance its capacity to recognize patterns over time, as shown in Figure 4.

6. Conclusions

In conclusion, the TFNN approach demonstrates significant advantages in enhancing the longevity and efficiency of IoT networks. By autonomously adapting to the diverse dynamics of these networks in a decentralized manner, TFNN minimizes the energy consumption associated with communication, which is a critical factor for battery-powered IoT devices. TFNN is a single-shot pattern recognition approach, and thus TFNN does not incur the communication and energy overhead associated with the multiple rounds needed to converge and recognize patterns. This study leveraged the MNIST digit dataset to evaluate the performance of TFNN in recognizing event patterns. The results illustrate that TFNN, utilizing the LEACH algorithm for hierarchical clustering, effectively reduces the energy costs associated with long-distance communications to the base-station. By allowing intra-cluster communications and limiting direct transmissions to the base-station to cluster heads and unclustered nodes, the algorithm significantly enhances operational efficiency. This approach not only prolongs the network's lifespan but also maintains a high level of accuracy in event detection. Ultimately, the findings underscore the potential of TFNN to support the sustainable operation of IoT networks, enabling them to function effectively over extended periods while maintaining acceptable levels of pattern recognition accuracy.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Atzori, L.; Iera, A.; Morabito, G. The Internet ofyyy Things: A survey. *Int. J. Comput. Telecommun.* **2010**, *54*, 2787–2805.
2. Baqer, M.; Kamal, A. S-Sensors: Integrating physical world inputs with social networks using wireless sensor networks. In Proceedings of the 2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Melbourne, Australia, 7–10 December 2009.
3. Estrin, D.; Culler, D.; Pister, K.; Sukhatme, G. Connecting the physical world with pervasive networks. *IEEE Pervasive Comput.* **2002**, *1*, 59–69. [CrossRef]
4. Baqer, M.; Balbis, L. A Sampling Sensor Nodes to Extend the Longevity of Wireless Sensor Networks. *Int. J. Sensors, Wirel. Commun. Control* **2019**, *9*, 53–63. [CrossRef]
5. Baqer, M.; Al Mutawah, K. Random node sampling for energy efficient data collection in wireless sensor networks. In Proceedings of the 2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, 2–5 April 2013; pp. 467–472.
6. Ashton, K. That 'Internet of Things' Thing. *RFID J.* **2009**, *22*, 97–114.
7. Elijah, O.; Rahman, T.A.; Orikumhi, I.; Leow, C.Y.; Hindia, M.N. An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges. *IEEE Internet Things J.* **2018**, *5*, 3758–3773. [CrossRef]
8. Patil, K.; Banerjee, S. Brief Overview on Wireless Sensor Network Technologies in the Internet of Things (IoT). *Int. J. Eng. Manag. Res.* **2023**, *13*, 1–8.
9. Arampatzis, T.; Lygeros, J.; Manesis, S. A Survey of Applications of Wireless Sensors and Wireless Sensor Networks. In Proceedings of the 2005 IEEE International Symposium on Mediterranean Conference on Control and Automation, Limassol, Cyprus, 27–29 June 2005.
10. Vermesan, O.; Friess, P.; Guillemin, P.; Gusmeroli, S.; Sundmaeker, H.; Bassi, A.; Jubert, I.S.; Mazura, M.; Harrison, M.; Eisenhauer, M.; et al. Internet of Things Strategic Research Roadmap. In *Internet of Things: Global Technological and Societal Trends from Smart Environments and Spaces to Green ICT*; River Publishers: Gistrup, Denmark, 2022; pp. 9–52.
11. Sethi, P.; Sarangi, S.R. Internet of Things; Architectures, Protocols, and Applications. *J. Electr. Comput. Eng.* **2017**, *2017*, 9324035. [CrossRef]
12. Akyildiz, I.; Su, W.S.; Sankarasubramaniam, Y.; Cayirci, E. A Survey on Sensor Networks. *IEEE Commun. Mag.* **2002**, *19*, 102–114. [CrossRef]

13. Mainwaring, A.; Culler, D.; Polastre, J.; Szewczyk, R.; Anderson, J. Wireless Sensor Networks for Habitat Monitoring. In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02), Atlanta, GA, USA, 28 September 2002.
14. Kim, S.; Pakzad, S.; Culler, D.; Demmel, J.; Fenves, G.; Glaser, S.; Turon, M. Wireless Sensor Networks for Structural Health Monitoring. In Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06), Boulder, CO, USA, 31 October–3 November 2006.
15. Bonnet, P.; Gehrke, J.; Seshadri, P. Querying the Physical World. *IEEE Pers. Commun. Mag.* **2000**, *7*, 10–15. [CrossRef]
16. Kramp, T.; Kranenburg, R.V.; Lange, S. Introduction to the Internet of Things. In *Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 1–10.
17. Abdul-Qawy, P.P.J.A.S.; Magesh, E.; Srinivasulu, T. The Internet of Things (IoT): An Overview. *Int. J. Eng. Res. Appl.* **2015**, *5*, 71–82.
18. Mashal, I.; Alsaryrah, O.; Chung, T.Y.; Yang, C.Z.; Kuo, W.H.; Agrawal, D.P. Choices for interaction with Things on Internet and Underlying Issues. *Ad Hoc Netw.* **2015**, *28*, 68–90. [CrossRef]
19. Khan, R.; Khan, S.U.; Zaheer, R.; Khan, S. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In Proceedings of the 10th International Conference on Frontiers of Information Technology, Islamabad, Pakistan, 17–19 December 2012.
20. Lim, W.Y.B.; Luong, N.C.; Hoang, D.T.; Jiao, Y.; Liang, Y.C.; Yang, Q. Federated Learning in Mobile Edge Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* **2020**, *22*, 2031–2063. [CrossRef]
21. Adi, E.; Anwar, A.; Baig, Z.; Zeadally, S. Machine Learning and Data Analytics for the IoT. *Neural Comput. Appl.* **2020**, *32*, 16205–16233. [CrossRef]
22. Zhang, T.; Gao, L.; He, C.; Zhang, M.; Krishnamachari, B.; Avestimehr, A.S. Federated Learning for the Internet of Things: Applications, Challenges, and Opportunities. *IEEE Internet Things Mag.* **2022**, *5*, 24–29. [CrossRef]
23. Lopez, P.G.; Montresor, A.; Epema, D.; Datta, A.; Higashino, T.; Lamnitchi, A.; Barcellos, M.M.; Felber, P.; Riviere, E. Edge-centric Computing: Vision and Challenges. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 37–42. [CrossRef]
24. Lazzarini, R.; Tianfield, H.; Charissis, V. Federated Learning for IoT Intrusion Detection. *AI* **2023**, *4*, 509–530. [CrossRef]
25. Hu, K.; Li, Y.; Xia, M.; Wu, J.; Lu, M.; Zhang, S.; Weng, L. Federated Learning: A Distributed Shared Machine Learning Method. *Complexity* **2021**, *2021*, 8261663. [CrossRef]
26. Al-Jarrah, M.A.; Yaseen, M.A.; Al-Dweik, A.; Dobre, O.A.; Alsusa, E. Decision Fusion for IoT-Based Wireless Sensor Networks. *IEEE Internet Things J.* **2020**, *7*, 1313–1326. [CrossRef]
27. Qin, X.; Gu, Y. Data fusion in the Internet of Things. *Procedia Eng.* **2011**, *15*, 3023–3026. [CrossRef]
28. Krishnamurthi, R.; Kumar, A.; Gopinathan, D.; Nayyar, A.; Qureshi, B. An overview of IoT sensor data processing, fusion, and analysis techniques. *Sensors* **2020**, *20*, 6076. [CrossRef] [PubMed]
29. He, C.; Annavaram, M.; Avestimehr, S. Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020.
30. Niknam, S.; Dhillon, H.S.; Reed, J.H. Federated Learning for Wireless Communications: Motivation, Opportunities, and Challenges. *IEEE Commun. Mag.* **2020**, *58*, 46–51. [CrossRef]
31. Tahir, M.; Ali, M.I. On the Performance of Federated Learning Algorithms for IoT. *Internet Things* **2022**, *3*, 273–284. [CrossRef]
32. Shaheen, M.; Farooq, M.S.; Umer, T.; Kim, B.S. Applications of Federated Learning; Taxonomy, Challenges, and Research Trends. *Electronics* **2022**, *11*, 670. [CrossRef]
33. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
34. Liu, L.; Zhang, J.; Song, S.; Letaief, K.B. Client-Edge-Cloud Hierarchical Federated Learning. In Proceedings of the IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020.
35. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
36. Imteaj, A.; Thakker, U.; Wang, S.; Li, J.; Amini, M.H. A Survey on Federated Learning for Resource-Constrained IoT Devices. *IEEE Internet Things J.* **2022**, *9*, 1–24. [CrossRef]
37. Nguyen, D.C.; Ding, M.; Pham, Q.V.; Pathirana, P.N.; Le, L.B.; Seneviratne, A.; Li, J.; Niyato, D.; Poor, H.V. Federated Learning Meets Blockchain in Edge Computing: Opportunities and Challenges. *IEEE Internet Things J.* **2021**, *8*, 12806–12825. [CrossRef]
38. Gharaibeh, A.; Salahuddin, M.A.; Hussini, S.J.; Khreishah, A.; Khalil, I.; Guizani, M.; Al-Fuqaha, A. Smart Cities: A Survey on Data Management, Security, and Enabling Technologies. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 2456–2501. [CrossRef]

39. Albaseer, A.; Ciftler, B.S.; Abdallah, M.; Al-Fuqaha, A. Exploiting Unlabeled Data in Smart Cities using Federated Edge Learning. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 1666–1671.
40. Kumar, A.; Braud, T.; Tarkoma, S.; Hui, P. Trustworthy AI in the Age of Pervasive Computing and Big Data. In Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Austin, TX, USA, 23–27 March 2020; pp. 1–6.
41. Li, W.; Wang, S. Federated meta-learning for spatial-temporal prediction. *Neural Comput. Appl.* **2022**, *34*, 10355–10374. [CrossRef]
42. Jiang, J.C.; Kantarci, B.; Oktug, S.; Soyata, T. Federated Learning in Smart City Sensing: Challenges and Opportunities. *Sensors* **2020**, *20*, 6230. [CrossRef]
43. Karimireddy, S.P.; Kale, S.; Mohri, S.R.M.; Stich, S.; Suresh, A.T. Scaffold: Stochastic Controlled Averaging for Federated Learning. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020.
44. Malinovskiy, G.; Kovalev, D.; Gasanov, E.; Condat, L.; Richtarik, P. From Local SGD to Local Fixed-Point Methods for Federated Learning. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020.
45. Pathak, R.; Wainwright, M.J. Fedsplit: An algorithmic framework for fast federated optimization. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20), Vancouver, BC, Canada, 6–12 December 2020.
46. Charles, Z.; Konečnỳ, J. Convergence and Accuracy Trade-offs in Federated Learning and Meta-learning. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Virtual, 13–15 April 2021.
47. Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; Poor, H.V. Tackling the objective inconsistency problem in heterogeneous federated optimization. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20), Vancouver, BC, Canada, 6–12 December 2020.
48. Collins, L.; Hassani, H.; Mokhtari, A.; Shakkottai, S. FedAvg with Fine Tuning: Local Updates Lead to Representation Learning. In Proceedings of the 36th International Conference on Neural Information Processing Systems, New Orleans, LA, USA, 10–15 December 2024.
49. Aledhari, M.; Razzak, R.; Parizi, R.M.; Saeed, F. Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Access* **2020**, *8*, 140699–140725. [CrossRef] [PubMed]
50. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated Optimization in Heterogeneous Networks. In *Machine Learning and Systems*; Dhillon, I., Papailiopoulos, D., Sze, V., Eds.; MLSys: Austin, TX, USA, 2020; Volume 2, pp. 429–450.
51. Nguyen, J.; Malik, K.; Zhan, H.; Yousefpour, A.; Rabbat, M.; Malek, M.; Huba, D. Federated Learning with Buffered Asynchronous Aggregation. In Proceedings of the 25th International Conference on Artificial Intelligence and Statistics, Virtual, 28–30 March 2022; Volume 151, Proceedings of Machine Learning Research; pp. 3581–3607.
52. Huang, L.; Yin, Y.; Fu, Z.; Zhang, S.; Deng, H.; Liu, D. LoAdaBoost: Loss-based AdaBoost Federated Machine Learning with Reduced Computational Complexity on IID and non-IID Intensive Care Data. *PLoS ONE* **2020**, *15*, e0230706. [CrossRef] [PubMed]
53. Yao, C.H.X.; Sun, L. Two-stream Federated Learning: Reduce the Communication Costs. In Proceedings of the IEEE Visual Communications and Image Processing (VCIP), Taichung, Taiwan, 10–12 December 2018.
54. Heinzelman, W.R.; Chandrakasan, A.; Balakrishnan, H. Energy-efficient Communication Protocol for Wireless Microsensor Networks. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI, USA, 4–7 January 2000.
55. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
56. Fix, E.; Hodges, J.L. An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation. *Int. Stat. Rev.* **1951**, *57*, 233–247.
57. Lu, Z.; Pan, H.; Dai, Y.; Si, X.; Zhang, Y. Federated Learning With Non-IID Data: A Survey. *IEEE Internet Things J.* **2024**, *11*, 19188–19209. [CrossRef]
58. Chen, L.; Zhao, D.; Tao, L.; Wang, K.; Qiao, S.; Zeng, X.; Tan, C.W. A Credible and Fair Federated Learning Framework Based on Blockchain. *IEEE Trans. Artif. Intell.* **2024**, *1*, 1–15. [CrossRef]
59. Deng, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142. [CrossRef]
60. *MATLAB*; version 23.2.0.2459199 (R2023b); The MathWorks Inc.: Natick, MA, USA, 2023.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.