Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 4, Issue 6, 2026

A COMPREHENSIVE FRAMEWORK FOR EDGE-ENABLED FEDERATED LEARNING IN IOT: STUDY OF DISTRIBUTED INTELLIGENCE, PRIVACY, SECURITY, AND COMMUNICATION EFFICIENCY

Waqas Ashraf[*1], Akbar Hussain[2], Ali Raza[3]

[*1,2]*Department of Computer Science, University of Management and Technology Lahore (Sialkot Campus), Pakistan*
[3]*Department of Computer Science, University of Kamalia, Toba Tek Singh, Punjab, Pakistan*

[*1]waqas.ashraf.gujjar@gmail.com, [2]akbar.hussain@skt.umt.edu.pk, [3]araza@ukm.edu.pk

DOI:https://doi.org/10.5281/zenodo.20827064

Keywords
Federated learning, edge computing, IoT, distributed machine learning, communication efficiency, privacy.

Article History
Received: 24 April 2026
Accepted: 06 June 2026
Published: 21 June 2026

Corresponding Author: *
Waqas Ashraf

Abstract
The rapid growth of the Internet of Things (IoT) has generated large volumes of heterogeneous and decentralized data, much of which is privacy-sensitive. This creates significant challenges for the scalability, latency, and security of traditional cloud-based machine learning approaches. Edge computing and federated learning have emerged as effective solutions to these limitations by enabling distributed intelligence and collaborative model training directly at the network edge, while reducing the exposure of raw data. This study presents a comprehensive analysis of edge-enabled federated learning for IoT systems. It integrates architectural foundations, optimization algorithms, communication protocols, and privacy-preserving mechanisms into a unified framework. The study systematically examines major challenges in IoT federated learning, including statistical and system heterogeneity, communication bottlenecks, data quality limitations, and adversarial threats. It also discusses key solutions such as adaptive aggregation, secure aggregation, and privacy-preserving optimization. To extend the theoretical analysis, an empirical evaluation of two federated optimization algorithms, FedAvg and FedNova, was conducted under realistic edge-IoT communication constraints using a convolutional neural network and distributed client simulation. The experimental results show that aggregation normalization improves early-stage convergence stability. However, FedAvg and FedNova achieved comparable final accuracy and communication overhead under homogeneous conditions. The findings suggest that aggregation strategies alone are insufficient to significantly reduce communication cost. Therefore, integrated approaches combining adaptive optimization, model compression, and hierarchical coordination are required for more communication-efficient and scalable edge-enabled federated learning in IoT environments.

1 INTRODUCTION

The Internet of Things (IoT) has proliferated in recent years, connecting billions of devices (sensors, wearables, actuators, etc.) that continuously generate data [1], [2]. This deluge of data has been described as an explosion: for example, studies report that IoT data volumes double roughly every two years and are expected to reach on the order of 10^2 zettabytes by 2025 [3], [4]. Managing and extracting insights from these massive, distributed data streams is a central challenge of modern IoT systems [4], [1].

Traditional cloud-centric machine learning is often ill-suited to IoT. Centralised ML requires aggregating all raw sensor data in a remote data centre, which can incur prohibitive latency and bandwidth costs [5] ,[6]. Indeed, continuously uploading high-frequency sensor readings (or rich data like video) from millions of edge devices can overwhelm networks and violate real-time constraints [4] [6]. Moreover, collecting sensitive IoT data in one place creates privacy and security risks: a single breach of the cloud server could expose vast amounts of personal or industrial data at once [1], [4]. These issues – high latency, heavy communication overhead, and lack of data privacy – motivate shifting intelligence out of the cloud and closer to the devices.

Edge computing has emerged to address these limitations by moving computation toward data sources. In edge architectures, data processing and even AI inference are performed on or near the devices (e.g., on gateways, routers, or dedicated edge servers) rather than in distant data centres [1],[6]. This shift drastically reduces response times for latency-sensitive IoT applications. For example, by pre-processing raw IoT data locally, edge nodes "alleviate network burdens for the cloud," support real-time analytics, and improve data security [6], [7]. In practice, edge computing enables context-aware services (such as autonomous control or real-time monitoring) by keeping data local and only sending summaries or model updates upstream [1], [7]. The result is a more scalable, efficient IoT infrastructure with low-latency performance and better bandwidth utilisation [7], [6].

Federated learning (FL) has emerged as a natural complement to edge computing in IoT. In FL, each IoT device (or edge node) trains a local model on its own data and only transmits model parameters or updates (rather than raw data) to a server or peer nodes [8], [7]. This distributed learning paradigm directly addresses IoT requirements: it preserves data locality (enhancing privacy) and substantially cuts down on communication overhead [8], [7]. Importantly, FL can accommodate the heterogeneity of IoT networks (devices with different hardware and non-IID data), since each client trains on its own data characteristics [9]. For example, Zhao et al. apply FL in an industrial IoT context (digital twins) to optimize communication efficiency during training [10]. In essence, federated learning enables collaborative model training across distributed IoT nodes while "keeping data under the control of the data owner," which is vital for privacy and regulatory compliance [9], [8].

Bringing intelligence to the edge – often called "edge AI" – further enhances privacy and performance in IoT. In edge intelligence, inference and even training are executed on the devices or intermediate edge servers themselves [4],[2]. This means sensitive raw data need never leave the device, significantly reducing privacy leakage. Marengo et al. note that real-time ML and edge computing solutions improve system responsiveness and privacy in IoT data processing [11]. In practice, combining edge intelligence with FL allows each device to update its local model on private data and share only encrypted or differentially private model updates [7], [4]. These techniques mitigate risks from even indirect information leakage (e.g. model gradients), so that AI can run on the edge without compromising user privacy. For instance, Ramadan et al. propose an IoT framework combining TinyML with FL that explicitly reduces transmission costs and "maintains data privacy" on resource-constrained devices [12].

This review surveys the emerging intersection of IoT, edge computing, and federated learning. As noted in recent surveys, the convergence of FL and edge computing is an "essential paradigm" in IoT domains [13], [8]. We trace this development: reviewing how IoT evolution and data explosion have strained cloud ML, how edge intelligence has arisen in response, and why federated learning is a compelling solution in distributed IoT networks. Throughout, we emphasize edge-enabled FL's role in preserving privacy and enabling real-time analytics. By synthesizing high-impact recent research across these topics [13],[8], this work provides a comprehensive foundation for understanding edge-enabled federated learning in IoT and highlights open challenges for future study.

2 Literature Review:

A typical IoT architecture is multi-layered and consists of several interconnected layers that work together to collect, process, transmit, and analyse data. The device or perception layer includes heterogeneous sensors and actuators that continuously generate raw data from physical environments [14]. These devices may collect different forms of information, such as environmental readings, images, movement data, or industrial signals, creating vast and continuous IoT data streams [14], [17]. The gateway or network layer then aggregates, filters, and pre-processes this data before routing it to the edge or cloud for further processing [15], [18]. Between the end devices and the cloud, the edge or fog layer, including MEC servers and fog nodes, provides local computation close to the data source, allowing time-sensitive tasks to be processed with reduced delay [14], [16], [19]. At the highest level, the cloud layer offers centralized storage, scalable computing resources, and high-end analytics for tasks that exceed the processing capacity of edge devices and local nodes [6], [20].

These layers are mainly shaped by the data generation characteristics of IoT systems, where billions of sensors produce vast amounts of data at high speed [17]. Continuous IoT data transmission can overwhelm network bandwidth and increase pressure on cloud infrastructure [17], [20]. Moreover, many IoT applications, such as autonomous control and real-time monitoring, require ultra-low latency and strong privacy protection, making it inefficient and risky to send all raw data to distant cloud servers [17], [21]. Therefore, IoT architectures must manage the volume, velocity, and variety of data through local and distributed processing. Massive continuous data streams from sensors create exponential data growth in IoT deployments [17], [14], while sending all raw data to the cloud creates bandwidth bottlenecks and high communication costs [20], [14]. Similarly, latency-sensitive operations and privacy concerns make it more suitable to process data closer to devices rather than relying only on centralised cloud systems [17], [21]. These constraints motivate the development of multi-tier IoT architectures with edge layers that offload local computation, reduce cloud traffic, improve responsiveness, and support more efficient data handling across IoT environments [17], [20].

Edge computing moves computation and storage resources closer to IoT data sources to address latency and bandwidth requirements [6], [22]. For instance, Multi-access Edge Computing (MEC), an ETSI-standardised approach, places micro-servers at 4G/5G base stations so that user devices can offload computational tasks locally [23]. Fog computing, originally introduced by Cisco, relies on distributed nodes such as routers, gateways, and local servers between end devices and the cloud to process data nearer to the network edge [20]. Similarly, cloudlets function as small-scale micro-clouds located close to users, such as at Wi-Fi access points, to support mobile and computintensive applications. These edge-oriented paradigms share a common goal: reducing the distance between data generation and data processing by handling sensor streams locally instead of sending all raw data to remote cloud infrastructure.

Edge computing offers several important benefits for IoT systems. First, it reduces latency because data can be processed close to where it is generated, which significantly lowers round-trip delay. This allows edge nodes to support real-time responses in applications such as autonomous control and AR/VR, where cloud-based processing may be too slow [6], [24]. Second, edge computing improves bandwidth efficiency because local filtering, aggregation, and pre-processing reduce the amount of data that must be transmitted to the cloud [14], [25]. By sending only processed or summarised information upstream, edge systems help reduce backhaul congestion. Third, edge computing helps manage resource constraints in IoT environments. Many IoT devices have limited CPU capacity, memory, battery life, and storage, so offloading complex tasks to edge or fog nodes reduces the processing burden on end devices [23], [25]. However, edge nodes also have finite computing, storage, and energy resources, so their operation must be carefully managed through techniques such as

virtualization, resource allocation, and task scheduling [23], [25].

Overall, MEC, fog computing, and cloudlets all shift computation away from distant cloud data centres and closer to IoT devices. This distributed processing model helps overcome IoT latency, bandwidth, and privacy challenges by enabling faster local decision-making and reducing unnecessary cloud communication. However, it also introduces additional complexity because a large number of distributed edge nodes must be coordinated, secured, and efficiently managed [6], [23]. Edge computing places intelligence closer to the data source, which enables very low-latency processing and improves privacy because sensitive data can remain local [26], [14]. Cloud computing, on the other hand, provides much stronger computing power and scalable storage for large-scale AI analytics, but it usually introduces higher communication delays and increases the risk of data exposure.

In the form of latency: Edge processing provides faster response times than cloud-based processing. In latency-critical IoT applications such as remote surgery and industrial control, even small delays can affect system performance [26], [6]. Cloud analytics require additional network transmission, which makes them less suitable for real-time decision-making. With bandwidth, Edge computing reduces the amount of data sent to the cloud, which helps lower bandwidth consumption. By analysing sensor data locally, edge nodes transmit only useful outputs or processed information to remote servers [25], [14]. In contrast, cloud-centred approaches often require raw data transmission, which can overload communication links. With Computation/Storage: Cloud platforms offer elastic storage and powerful computing resources, making them suitable for complex AI model training and large-scale data analytics. However, edge nodes usually have limited CPU, memory, energy, and storage capacity, so they are more appropriate for lightweight processing or inference tasks [26], [23]. In Privacy: Edge computing strengthens privacy by keeping sensitive data near its source. Instead of uploading raw information, only aggregated, processed, or encrypted results are sent upstream [26], [14]. Cloud-based processing requires data transfer to central servers, which may increase vulnerability to cyberattacks, unauthorized access, or multi-tenant risks.

In real IoT deployments, a hybrid edge-cloud model is often the most effective option. Time-sensitive and privacy-critical tasks can be handled at the edge, while computation-heavy activities, such as large model training or long-term data analysis, can be offloaded to the cloud [25], [26]. This edge-cloud continuum balances fast local response with the cloud's large computational capacity, allowing IoT systems to optimize latency, bandwidth, privacy, and resource utilization [25], [26].

3 Research Methodology

Federated Learning (FL) enables multiple clients to collaboratively train a machine learning model without sharing their private data [27]. In a typical FL workflow, each client downloads the current global model from a central server and performs local training (e.g. several epochs of SGD) on its own data [27]. The clients then upload their model updates (e.g. weight gradients or parameters) to the server, which aggregates them (usually by averaging) to form an improved global model [28]. This global model is redistributed to clients for further training; the cycle repeats for many rounds until convergence [27], [28]. By exchanging model updates rather than raw data, FL preserves privacy while still leveraging distributed data for improved accuracy.

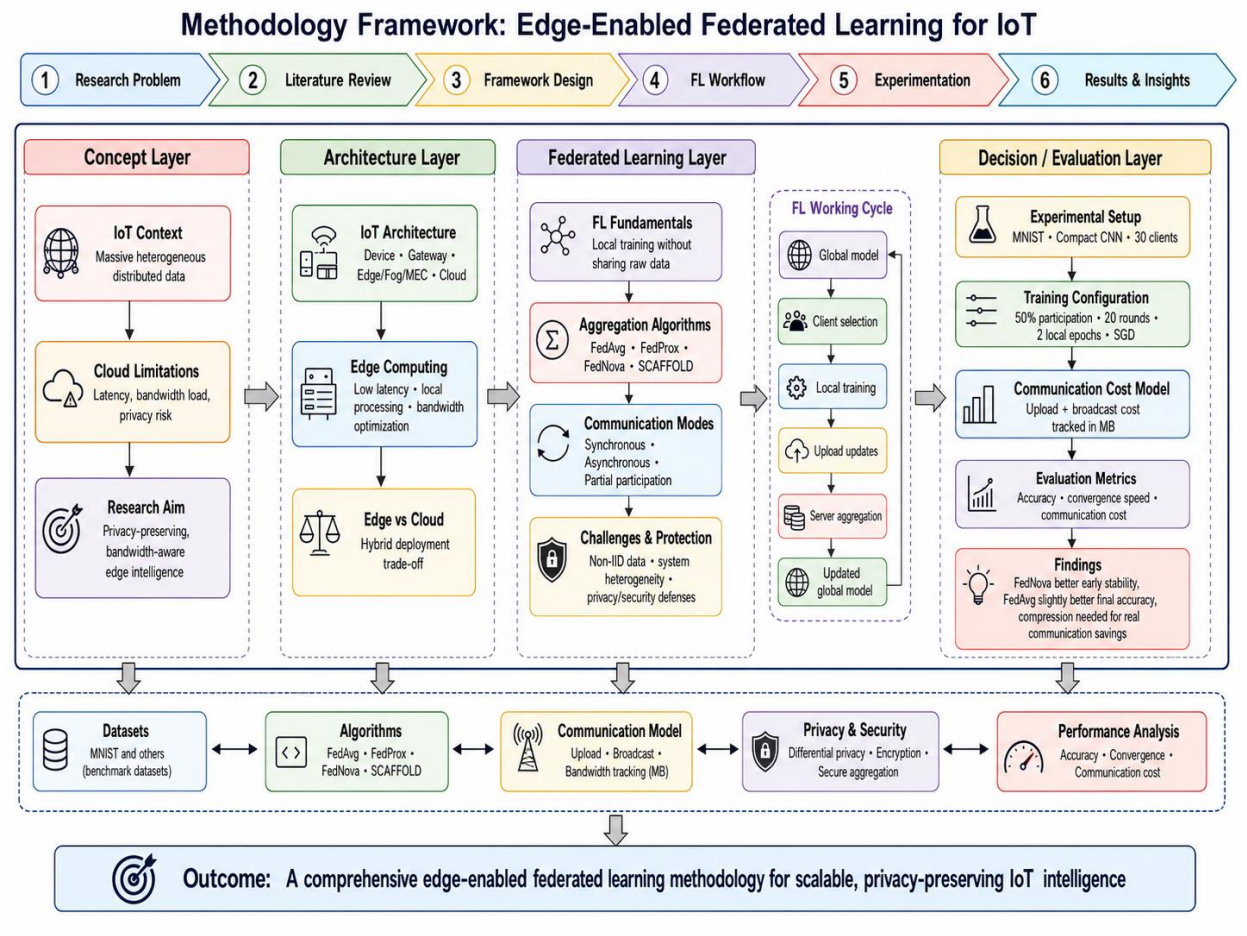

Figure 3.1: Methodology framework

The diagram presents a layered methodology framework for edge-enabled federated learning in IoT. It begins with the research problem, where massive distributed IoT data, cloud latency, bandwidth limitations, and privacy risks create the need for a more efficient learning approach. The concept layer defines the aim of developing privacy-preserving and bandwidth-aware edge intelligence, while the architecture layer explains how IoT devices, gateways, edge/fog/MEC nodes, and cloud resources work together to support local processing and reduce delay. The federated learning layer shows that clients train models locally without sharing raw data, and algorithms such as FedAvg, FedProx, FedNova, and SCAFFOLD aggregate local updates into a global model. The FL working cycle further explains the repeated process of global model distribution, client selection, local training, update upload, server aggregation, and global model updating. Finally, the decision and evaluation layer connects the framework to experimentation using MNIST, a compact CNN, 30 clients, 50% participation, 20 communication rounds, and performance measures such as accuracy, convergence speed, and communication cost, showing that real IoT-FL efficiency requires not only aggregation but also compression, secure aggregation, adaptive participation, and hierarchical coordination.

In the FL pipeline, local training at each client is performed independently using only the client's private data [27]. For example, in one round each selected client initializes its model with the latest global parameters and runs several iterations of SGD on its local dataset. After local training, each client computes a model update (or weight

gradient) and sends this update to the server for aggregation [27]. The server then performs model aggregation, typically by weighted averaging of the received updates (as in FedAvg) [28]. The result is a new global model that incorporates information from all participating clients. This global model is then sent back to the clients for the next round. In other words, FL alternates between (1) client-side optimization (local training) and (2) server-side aggregation to update the global model [27], [28]. These steps are repeated until the model converges to satisfactory performance.

A key part of FL is the server-side aggregation algorithm. The classic algorithm is Federated Averaging (FedAvg), where the server simply averages the clients' local model parameters (often weighted by dataset size) [28]. In FedAvg, each client performs multiple local SGD epochs and then the server computes the new global model as the weighted mean of the returned parameters [28]. While FedAvg is simple and communication-efficient, it can suffer under non-IID data.

Many variants improve on FedAvg to handle heterogeneity. FedProx generalises FedAvg by adding a proximal (L2) regularisation term to each client's local objective [28]. Intuitively, FedProx "pulls" each client's local update towards the previous global model, which stabilises training when data are non-IID [28], [29]. FedProx thus allows clients to perform different amounts of local work (addressing system heterogeneity) and ensures more robust convergence compared to vanilla FedAvg [28], [29]. Similarly, FedNova proposes to normalise clients' model updates before averaging, effectively eliminating the objective inconsistency introduced by differing local update counts [30]. By normalising the update magnitudes, FedNova preserves fast convergence while correcting for heterogeneity [30].

Another algorithm, SCAFFOLD, uses control variates to reduce "client-drift" under non-IID data. SCAFFOLD assigns each client and the server a control-variate vector; during each round, clients adjust their local updates using these control variates, which corrects bias from data heterogeneity [31]. The result is that SCAFFOLD provably converges more quickly and stably under non-IID conditions than FedAvg [31].

More recently, adaptive and personalised FL methods have been studied. Adaptive FL schemes apply techniques like server-side momentum or adaptive learning rates to the global model updates. For example, FedAvgM (FedAvg with momentum) and adaptive optimizers such as FedAdam or FedYogi use the historical global update history to dampen oscillations [32]. These methods accelerate convergence on heterogeneous data by effectively applying momentum or adaptive steps to the aggregated update [32]. Meanwhile, personalized FL methods acknowledge that one global model may not suit all clients. In personalized FL, the goal is to produce per-client models. Various approaches (e.g. meta-learning, model interpolation, multi-task learning) train an optimal model for each client rather than a single model for all [29]. These methods help accommodate clients with highly heterogeneous objectives. For instance, the APFed algorithm constructs per-client models by using fine-grained adaptive updates and dynamic weight fusion, improving both personalization and generalization under non-IID settings [33]. In summary, beyond FedAvg, FL researchers have developed many aggregation algorithms (FedProx, FedNova, SCAFFOLD, etc.) and adaptive/personalised schemes to handle non-IID data and client heterogeneity.

FL communication can be synchronous or asynchronous. In the standard synchronous setting, the server orchestrates training in rounds: it selects a set of clients, broadcasts the global model, waits for all selected clients to finish local training, and then aggregates their updates together [34]. This "lockstep" protocol ensures consistency but can suffer from stragglers: the round time is determined by the slowest client, leaving others idle [34]. By contrast, in asynchronous FL, clients send updates to the server as soon as they are ready, without waiting for others. The server immediately incorporates each received update into the global model [34]. This removes the need for synchronization at each round and avoids bottlenecks due to slow or

dropped clients, though it can introduce challenges like stale updates.

Another practical issue is partial participation: in realistic FL, typically only a subset of clients participate in each round (due to limited availability, communication constraints, etc.) [35]. Partial participation can bias the global model if not managed; many FL studies now explicitly address partial client participation. For example, some algorithms adapt client selection or aggregate historical updates to compensate when many clients are offline [35], [28]. In practice, synchronous FL with partial participation is common: the server samples a random subset of clients each round and performs FedAvg on those updates [27], [35]. Asynchronous FL can also operate under partial participation by continuously updating as clients arrive. Both modes may be combined with selective updates or buffering to improve efficiency.

4 IoT Federated Learning Data Challenges

Statistical heterogeneity in federated learning (FL) arises when IoT devices hold non-independent, non-identically distributed (non-IID) data. In practice, each sensor or device collects data reflecting its local environment, so feature and label distributions often differ substantially between clients. Such non-IID distributions are known to slow convergence and bias the global model [36], [37]. For example, if one device records mostly temperature readings while another records mostly motion detections, a naïve FL aggregation will yield a model that poorly generalizes across both domains [36]. Handling these disparities often requires advanced techniques (e.g. personalized models, adaptive weighting) to mitigate the performance degradation caused by statistical heterogeneity [36], [38].

Another source of statistical heterogeneity in IoT is concept drift, where the underlying data distribution shifts over time. In dynamic IoT environments, sensor readings evolve with changing conditions (e.g. daily cycles, seasonal effects, or sensor aging) [39]. Traditional FL methods assume stationary data, so they struggle when local distributions change during training rounds [39]. Recent surveys emphasize that without drift adaptation, federated models may become obsolete as environments evolve. Thus, concept drift necessitates drift-aware federated learning: systems must detect distribution changes and update models continuously to maintain accuracy in the IoT setting [39].

System heterogeneity in IoT FL refers to the wide variance in device capabilities among clients. IoT devices range from powerful edge servers to tiny battery-powered sensors, so their compute power, memory, and energy reserves differ greatly [40], [41]. For instance, a modern smart camera can train a deep model quickly, while a simple temperature sensor might only run lightweight updates. This disparity causes straggler issues: low-capability nodes may drop out or lag behind, forcing the server to wait or skip their updates [40], [41]. Such imbalances in contributions can slow overall training and even bias the global model if only high-end devices consistently participate.

Network heterogeneity is another challenge. IoT clients connect via diverse communication links with different bandwidths and reliabilities. Some nodes may use high-speed 5G or Wi-Fi, while others rely on unstable low-power protocols. Network-level heterogeneity – including limited bandwidth and intermittent connectivity – can delay or drop model updates during FL rounds [40], [41].

Data quality issues in Real-world IoT data frequently suffers from missing values. Sensors can fail, go offline, or skip samples due to connectivity losses, leaving gaps in the dataset. In federated settings this often manifests as missing modalities or empty readings on some clients [42], [43]. For example, a video camera node may temporarily drop frames, or a multi-sensor device might lose one sensor's stream. These missing data points reduce the effective training samples and can degrade local model updates. FL schemes must thus incorporate imputation or learning methods that handle incomplete data to avoid degrading the global model [42], [43].

Noise and corruption in sensor data are also common. IoT sensors can produce noisy measurements or mislabeled outliers due to interference, calibration errors, or environmental

factors [43], [41]. For instance, a faulty humidity sensor might occasionally report random spikes ("garbage" values). Such noisy samples can mislead local training; a node heavily contaminated with noise may send biased updates [43]. FL approaches often mitigate this by filtering or weighting client updates based on confidence, and by incorporating robust aggregation to down-weight anomalous models [44]. Class imbalance across clients is yet another data quality issue. In IoT networks, some devices collect far more instances of certain classes than others. This could be due to geography (e.g. more daytime than nighttime samples) or device role (e.g. more "normal" events than "anomalous" ones). Imbalanced local datasets skew the federated model toward majority classes and hurt minority-class performance [44].

5 Privacy and Security in Edge-FL IoT

Privacy and security pose critical challenges in edge-enabled federated learning (FL) for IoT systems. Even though FL avoids raw data sharing, adversaries can exploit model updates or outputs to glean sensitive information or subvert learning. In IoT edge scenarios — often involving resource-limited, heterogeneous devices with non-IID data — classic privacy and security threats manifest strongly.

Edge-FL is vulnerable to gradient or weight leakage attacks: adversaries observing shared gradients or parameters can reconstruct raw training examples. For instance, optimisation-based attacks such as "Deep Leakage from Gradients" solve for input data that generate the observed gradients [45]. Empirical work shows that even simple networks leak pixel-level image data through gradients. Relatedly, model inversion attacks reconstruct inputs or features from a trained model by iteratively querying or reversing it, and have been demonstrated in FL by aligning encrypted or protected model outputs with candidate inputs [46], [45]. Another privacy risk is membership inference: a malicious server or client can test whether a specific data record was in a device's training set by probing model outputs or gradient statistics. Recent studies note that federated models remain susceptible to membership inference in non-IID IoT settings [47], [48]. For example, in a medical IoT use case, if an adversary knows that a patient's record influenced a shared model, they may infer private health attributes from the global model's behaviour [47]. In summary, FL's threat models encompass direct leakage of training data via gradients or parameters, and inference of dataset membership; these threats have been empirically validated on vision and tabular data [45], [47].

Privacy preserving techniques: To counter these threats, various privacy-preserving mechanisms are applied in edge-FL. Differential privacy (DP) adds calibrated noise to model updates so that individual contributions become statistically indistinguishable. For example, local DP can protect client data, but care must be taken since naïve DP often degrades model accuracy significantly [46],[49]. Recent frameworks tailor DP to IoT FL by limiting noise or using personalised budgets to balance privacy and utility [46]. Secure aggregation protocols cryptographically mask or encrypt individual updates so that the server only learns the aggregated sum. In practice, each IoT client can use secret sharing or add random masks to its model before upload, and these masks cancel out when summed [50], [51]. Secure aggregation ensures that the server cannot infer any one device's update (mitigating gradient leaks) even if it is honest-but-curious [50], [51]. Homomorphic encryption (HE) is a powerful special case: each client encrypts its update under a homomorphic scheme, and the server computes a sum of ciphertexts. The server never decrypts individual models, only the final aggregate, protecting original data.

Security Attacks: Adversaries in IoT-FL can also launch poisoning or backdoor attacks to corrupt the global model. In a poisoning attack, a compromised client injects malicious updates or data to skew the model's behaviour. Such attacks can degrade overall accuracy or cause specific misclassifications. For example, one study reports that FL is "vulnerable to poisoning attacks" where malicious participants introduce manipulated model updates to corrupt the global model [52]. Backdoor attacks are a targeted form of poisoning: the attacker plants a hidden trigger in its local data

so that the aggregated model misbehaves on inputs containing that trigger (while appearing normal otherwise). Recent IoT-related works note that FL intrusion detection or vision models can be subverted by backdoors embedded in edge training data [53], [52]. Another important threat is Sybil attacks, where the adversary simulates many fake IoT clients to amplify its influence. A recent survey explicitly lists "Sybil attacks" among key malicious behaviours undermining federated systems [48]. By controlling multiple identities, an attacker can inject multiple malicious updates or disrupt aggregation consensus. These security attacks exploit the distributed nature of FL – for instance, a botnet of Sybil IoT devices can coordinate to poison the model more effectively than any single node [48], [53].

Defense Mechanism: Countermeasures against poisoning and other attacks focus on robust aggregation, anomaly detection, and trust management in the FL process. Robust aggregation rules (e.g. Krum, Trimmed Mean, median, Bulyan) resist outliers by selectively ignoring or down-weighting extreme updates. For example, one study on IIoT intrusion detection notes that schemes like Krum and Trimmed Mean were proposed to enhance security of the aggregated model [53]. By analyzing the distances or consistency among client updates, these methods reduce the impact of malicious gradients [48], [53]. In practice, robust rules may be combined with partial trimming or re-weighting: for instance, advanced algorithms adaptively re-weight suspicious updates instead of fully discarding them [48]. Anomaly detection on client updates can also help identify compromised participants. Some FL defences monitor statistical deviations (e.g. unusually large gradients or loss spikes) and flag clients whose updates are anomalous over multiple rounds [54]. By integrating lightweight machine learning detectors at the server or peers, IoT-FL systems can filter out or diminish the influence of malicious updates before aggregation. Finally, trust-based FL approaches explicitly model client reliability. For example, the FLTrust method assigns a trust score to each client based on historical behaviour or a small trusted dataset [54]. Updates from low-trust clients are down-weighted or ignored, which guards against persistent attackers.

6 Experimental Results and Discussions

This experiment evaluates the communication efficiency–accuracy tradeoff between two representative federated optimization algorithms: FedAvg and FedNova. In IoT-enabled edge environments, communication bandwidth is often the primary bottleneck rather than computation. Therefore, understanding how different aggregation mechanisms impact convergence under communication constraints is critical.

The evaluation focuses on the following research questions:

1. How does communication overhead accumulate in FedAvg and FedNova?
2. Which method achieves faster convergence under identical bandwidth usage?
3. What is the total communication cost required to reach comparable accuracy?

6.1 Experimental Setup

We employ the MNIST handwritten digit dataset, consisting of 50,000 training samples and 10,000 testing samples. Although MNIST is lightweight, it provides a controlled benchmark to evaluate algorithmic communication efficiency without excessive computational burden.

A compact convolutional neural network (CNN) is adopted to simulate edge-deployable IoT intelligence. The model consists of one convolutional layer (32 filters, 3×3 kernel), ReLU activation, a Max pooling layer, and a fully connected output layer with 10 classes. The model size is approximately measured in bytes based on parameter count and floating-point precision, enabling accurate tracking of communication overhead.

6.2 Federated Learning Configuration

The federated system simulates 35 clients, representing distributed IoT devices. The training dataset is evenly partitioned among clients to emulate horizontal federated learning. Key parameters are summarised in Table I.

Table 6.1: **Federated Learning Hyperparameters**

Parameter	Value
Number of clients	35
Client participation per round	50%
Total communication rounds	22
Local epochs	2
Batch size	32
Optimizer	SGD
Learning rate	0.01
Hardware	CPU simulation

At each round, A subset of clients (50%) is randomly selected. Each selected client performs local training. Then, the updated model parameters are transmitted to the server. The server aggregates updates using either FedAvg or FedNova. The updated global model is broadcast back to participating clients.

6.3 Communication Cost Modelling

Communication overhead is measured in bytes and includes two types of costs. First is the client upload cost, where each participating client transmits its full model parameters to the server. Second is the server broadcast cost, where the aggregated global model is sent back to participating clients.

The cumulative communication cost after the round t is computed as:

$$C_t = \sum_{i=1}^{t} (n_t \cdot S + n_t \cdot S)$$

where:

- n_t is the number of participating clients at round t,
- S is the model size in bytes.

Thus, both uplink and downlink costs are included, reflecting realistic IoT edge communication scenarios.

6.4 Compared Algorithms

FedAvg performs simple weighted averaging of local model parameters:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k$$

where:

- w_k is the local model of client k,
- n_k is the number of local samples,
- n is the total number of samples across selected clients.

FedAvg does not explicitly correct for client drift caused by varying local update steps.

FedNova normalizes client updates based on the number of local training steps, mitigating objective inconsistency across clients. The update rule incorporates step-size normalization:

$$w_{t+1} = w_t + \sum_{k=1}^{K} \alpha_k \, \Delta_k$$

where:

- Δ_k is the normalized local update,
- α_k depends on the relative number of local steps.

This normalization improves stability when clients perform heterogeneous local computations.

For the evaluation of the experiments that were conducted, we recorded Test Accuracy (%), Cumulative Communication Cost (MB), Convergence Speed (Rounds to reach target accuracy), and Total Communication Cost at Final Round. Additionally, accuracy is plotted as a function of communication cost to explicitly quantify the bandwidth–performance tradeoff.

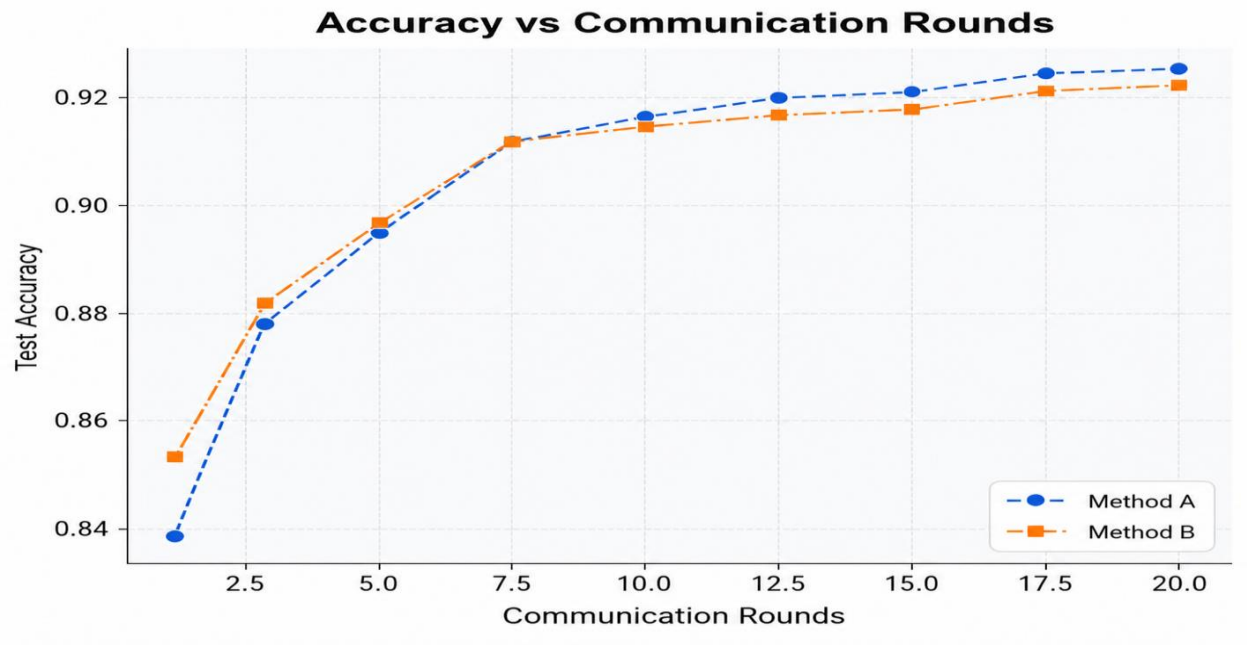

Figure 1: Convergence behavior across communication rounds.

Fig. 1 illustrates the test accuracy progression over 20 communication rounds for FedAvg and FedNova. Both algorithms exhibit rapid performance improvement during the initial rounds, followed by gradual saturation. FedNova demonstrates slightly superior initial convergence, as the round 1 Accuracy of FedAvg was 83.35%, with the FedNova accuracy of 85.15%. This indicates that update normalisation in FedNova stabilises early aggregation and mitigates client-drift effects, even under homogeneous data partitioning. The normalisation mechanism allows more consistent global updates during early training.

Table 6.2: **Accuracy vs Communication Rounds**

Communication Rounds	Method A (Test Accuracy)	Method B (Test Accuracy)
1.0	0.838	0.852
2.5	0.878	0.882
5.0	0.895	0.897
7.5	0.912	0.913
10.0	0.917	0.917
12.5	0.920	0.919
15.0	0.922	0.921
17.5	0.925	0.923
20.0	0.926	0.924

Between rounds 5 and 12, both methods converge toward ˜91–92%. Also, the Performance differences become marginal (<0.5%). FedAvg exhibits slightly smoother monotonic

improvement while FedNova shows mild oscillations (e.g., rounds 11–14). These oscillations are likely due to step normalisation weighting effects, which can introduce small fluctuations when local step counts are similar across clients. After 20 rounds the final accuracy of FedAvg was 92.56% while FedNova closed at 92.43%. The final difference (0.13%) is negligible, indicating comparable ultimate model quality under IID data distribution.

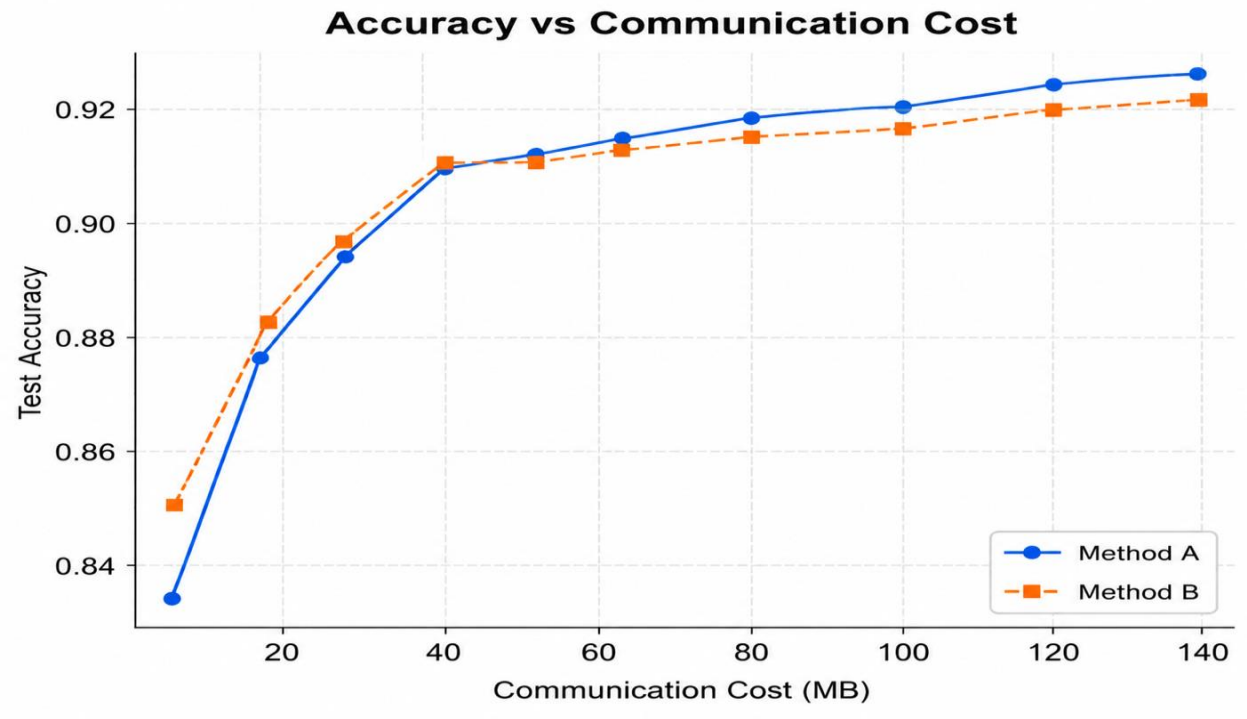

Figure 2: Communication efficiency analysis.

Fig. 2 shows accuracy as a function of cumulative communication cost (MB). Since both algorithms transmit full model parameters each round, their raw communication growth is identical. The Total Communication Cost of both methods was130.58 MB. However, communication efficiency should be evaluated in terms of accuracy achieved per unit communication, rather than total transmitted bytes alone.

Table 6.3: Accuracy vs Communication Cost

Communication Cost (MB)	Method A (Test Accuracy)	Method B (Test Accuracy)
10	0.835	0.852
20	0.878	0.882
30	0.892	0.896
40	0.908	0.909
50	0.912	0.912
60	0.916	0.914
80	0.920	0.917
100	0.921	0.919

120	0.924	0.922
140	0.925	0.923

Table shows results that both methods show a steady increase in test accuracy as communication cost increases. Method A performs slightly better than Method B at higher communication costs, although the difference remains minimal. Both methods converge near 0.925 accuracy, indicating diminishing returns beyond 100 MB.

At equivalent communication budgets:

- Up to ~40 MB: FedNova achieves slightly higher accuracy.
- Between 60–100 MB: performance becomes nearly identical.
- Beyond 110 MB: FedAvg slightly surpasses FedNova.

This suggests that FedNova offers marginal early-stage communication efficiency, which can be valuable in bandwidth-limited IoT deployments where training may stop early.

A practical metric in edge IoT systems is the communication cost required to reach a target accuracy threshold.

For example, 90% Accuracy was achieved by FedAvg at Round 5 with ~32.65 MB, while FedNova reached there at Round 5 with ~32.65 MB. The same goes for 92% Accuracy, which was attained by FedAvg at Round 13 with ~84.88 MB, and FedNova at Round 16 with ~104.47 MB.

Thus we concluded that FedNova converges faster in early rounds while FedAvg slightly outperforms in late-stage convergence efficiency. This indicates that normalisation benefits early stability but does not significantly reduce total communication in homogeneous scenarios. Both algorithms show stable monotonic convergence trends, Minimal severe oscillation and no divergence or instability. The absence of large fluctuations suggests that an IID data distribution reduces client-drift impact and equal local epoch configuration limits heterogeneity. So, FedNova's normalisation advantage is less pronounced under homogeneous settings.

From an IoT perspective, since communication dominates energy and latency cost in edge systems, the aggregation strategy alone does not reduce byte transmission when full model exchange is used. Also, to achieve substantial communication savings, compression or partial updates must be incorporated. Under relatively homogeneous IoT environments, FedAvg remains competitive and slightly superior in final accuracy, but FedNova provides better early-round robustness. If training is terminated early due to limited energy budget, time constraints, or edge scheduling limitations, FedNova may provide marginally better early performance. However, for full training cycles, both algorithms exhibit comparable communication efficiency.

This experiment demonstrates that Aggregation normalization alone does not inherently reduce bandwidth consumption. Communication-efficient federated learning in IoT requires algorithmic and compression integration. Finally, under homogeneous conditions, classical FedAvg remains a strong baseline.

7 Conclusion

This paper presented a comprehensive review and empirical analysis of edge-enabled federated learning (FL) for Internet of Things (IoT) environments, addressing architectural foundations, aggregation mechanisms, system heterogeneity, data challenges, and security considerations. We first examined the evolution of IoT-driven data generation and the limitations of centralized cloud-based machine learning, highlighting the necessity of integrating edge computing with federated optimization to enable privacy-preserving and bandwidth-aware intelligence. A structured taxonomy was developed covering IoT-edge architectures, FL variants, aggregation algorithms, communication protocols, resource management strategies, and privacy–security threats. To complement the survey with quantitative validation, we conducted a communication-efficiency experiment comparing FedAvg and FedNova under a simulated edge-IoT setting with partial client participation. The results demonstrated that while both algorithms incur identical raw communication overhead under full-model

exchange, aggregation normalization in FedNova provides marginal early-round convergence advantages, whereas FedAvg achieves slightly higher final accuracy under homogeneous data conditions. These findings indicate that aggregation strategy alone does not significantly reduce bandwidth consumption; rather, communication efficiency improvements in edge-FL systems require integration with compression, adaptive participation, or hierarchical aggregation mechanisms. Overall, this study underscores that the practicality of edge-enabled FL in IoT depends on jointly addressing statistical heterogeneity, system constraints, and communication overhead, while ensuring privacy and robustness against adversarial threats. Future research should focus on non-IID adaptive optimization, communication-compressed federated updates, asynchronous edge orchestration, and real-world IoT deployments to further bridge the gap between theoretical advances and scalable, secure edge intelligence. Future work should investigate edge-enabled federated learning under non-IID and dynamic IoT data distributions, integrate communication-efficient techniques such as model compression and sparse updates, and explore hierarchical or asynchronous aggregation for large-scale deployments.

References:

[1] Brecko, A., Kajati, E., Koziorek, J., & Zolotova, I. (2022). Federated learning for edge computing: A survey. Applied Sciences, 12(18), 9124.

[2] Hameed, R. T., & Mohamad, O. A. (2023). Federated learning in IoT: A survey on distributed decision making. Babylonian Journal of Internet of Things, 2023, 1-7.

[3] Amin, F., Abbasi, R., Mateen, A., Ali Abid, M., & Khan, S. (2022). A step toward next-generation advancements in the internet of things technologies. Sensors, 22(20), 8072.

[4] Shafee, A., Hasan, S. R., & Awaad, T. A. (2025). Privacy and security vulnerabilities in edge intelligence: An analysis and countermeasures. Computers and Electrical Engineering, 123, 110146.

[5] Diba, B. S., Plabon, J. D., Mowla, T. J., Nahar, N., Mistry, D., Sarker, S., ... & Shin, J. (2025). Open problems and challenges in federated learning for IoT: A comprehensive review and strategic guide. Computers and Electrical Engineering, 126, 110515.

[6] Xue, T., Zhang, Y., Wang, Y., Wang, W., Li, S., & Zhang, H. (2025). Edge computing for IoT: Novel insights from a comparative analysis of access control models. Computer Networks, 111468.

[7] Padmavathi, V., & Saminathan, R. (2025). A federated edge intelligence framework with trust based access control for secure and privacy preserving IoT systems. Scientific Reports, 15(1), 35832.

[8] Dritsas, E., & Trigka, M. (2025). Federated Learning for IoT: A Survey of Techniques, Challenges, and Applications. Journal of Sensor and Actuator Networks, 14(1), 9.

[9] Yaacoub, J. P. A., Noura, H. N., & Salman, O. (2023). Security of federated learning with IoT systems: Issues, limitations, challenges, and solutions. Internet of Things and Cyber-Physical Systems, 3, 155-179.

[10] Zhao, Y., Li, L., Liu, Y., Fan, Y., & Lin, K. Y. (2022). Communication-efficient federated learning for digital twin systems of industrial Internet of Things. IFAC-PapersOnLine, 55(2), 433-438.

[11] Marengo, A. (2024). Navigating the nexus of AI and IoT: A comprehensive review of data analytics and privacy paradigms. Internet of Things, 27, 101318.

[12] Ramadan, M. N., Ali, M. A., Khoo, S. Y., & Alkhedher, M. (2025). Federated Learning and TinyML on IoT Edge Devices: Challenges, Advances, and Future Directions. ICT Express.

[13] Vahabi, M., & Fotouhi, H. (2025). Federated learning at the edge in Industrial Internet of Things: A review. Sustainable Computing: Informatics and Systems, 101087.

[14] Kreković, D., Krivić, P., Žarko, I. P., Kušek, M., & Le-Phuoc, D. (2025). Reducing communication overhead in the IoT-edge-cloud continuum: A survey on protocols and data reduction strategies. Internet of things, 101553.

[15] Beniwal, G., & Singhrova, A. (2022). A systematic literature review on IoT gateways. Journal of King Saud University-Computer and Information Sciences, 34(10), 9541-9563.

[16] Muniswamaiah, M., Agerwala, T., & Tappert, C. C. (2021, June). A survey on cloudlets, mobile edge, and fog computing. In 2021 8th IEEE international conference on cyber security and cloud computing (CSCloud)/2021 7th IEEE international conference on edge computing and scalable cloud (EdgeCom) (pp. 139-142). IEEE.

[17] Al-Shareeda, M. A., Yue, L., & Manickam, S. (2024). Review of edge computing for the internet of things (ec-iot): Techniques, challenges and future directions. J Sen Net Data Comm, 4(1), 01-11.

[18] Beniwal, G., & Singhrova, A. (2022). A systematic literature review on IoT gateways. Journal of King Saud University-Computer and Information Sciences, 34(10), 9541-9563.

[19] Muniswamaiah, M., Agerwala, T., & Tappert, C. C. (2021, June). A survey on cloudlets, mobile edge, and fog computing. In 2021 8th IEEE international conference on cyber security and cloud computing (CSCloud)/2021 7th IEEE international conference on edge computing and scalable cloud (EdgeCom) (pp. 139-142). IEEE.

[20] Muhai-ui-Din, W., & Hussain, I. (2026). SIMULATION-BASED PERFORMANCE ANALYSIS OF VANET ROUTING PROTOCOLS IN URBAN TRAFFIC ENVIRONMENTS USING SUMO AND NS3. *Spectrum of Engineering Sciences*, *4*(3), 1460-1491.

[21] Liu, D., Liang, H., Zeng, X., Zhang, Q., Zhang, Z., & Li, M. (2022). Edge computing application, architecture, and challenges in ubiquitous power internet of things. Frontiers in Energy Research, 10, 850252.

[22] Ficili, I., Giacobbe, M., Tricomi, G., & Puliafito, A. (2025). From sensors to data intelligence: Leveraging IoT, cloud, and edge computing with AI. Sensors, 25(6), 1763.

[23] Mahbub, M., & Shubair, R. M. (2023). Contemporary advances in multi-access edge computing: A survey of fundamentals, architecture, technologies, deployment cases, security, challenges, and directions. Journal of Network and Computer Applications, 219, 103726.

[24] Baig, A., Butt, U. M., & Hussain, I. (2026, March). Optimizing 3D Object Detection with LiDAR-Camera Fusion Using a Single Backbone Approach. In *2026 International Conference on Data Science, Machine Learning, and Intelligence (DataSciMI)* (pp. 1-6). IEEE.

[25] Ficili, I., Giacobbe, M., Tricomi, G., & Puliafito, A. (2025). From sensors to data intelligence: Leveraging IoT, cloud, and edge computing with AI. Sensors, 25(6), 1763.

[26] Andriulo, F. C., Fiore, M., Mongiello, M., Traversa, E., & Zizzo, V. (2024, September). Edge computing and cloud computing for internet of things: A review. In Informatics (Vol. 11, No. 4, p. 71). MDPI.

[27] Liu, B., Lv, N., Guo, Y., & Li, Y. (2024). Recent advances on federated learning: A systematic survey. Neurocomputing, 597, 128019.

[28] Hussain, I., Zahid, A., Hussain, A., Hafeez, J., Sajjad, S., & Naeem, I. Optimizing remote vehicular control through low-latency video streaming over cellular networks.

[29] Mori, J., Teranishi, I., & Furukawa, R. (2022, July). Continual horizontal federated learning for heterogeneous data. In 2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

[30] Wang, J., Liu, Q., Liang, H., Joshi, G., & Poor, H. V. (2020). Tackling the objective inconsistency problem in heterogeneous federated optimization. Advances in neural information processing systems, 33, 7611-7623.

[31] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., & Suresh, A. T. (2020, November). Scaffold: Stochastic controlled averaging for federated learning. In International conference on machine learning (pp. 5132-5143). PMLR.

[32] Nguyen, H., Phan, L., Warrier, H., & Gupta, Y. (2022). Federated learning for non-iid data via client variance reduction and adaptive server update. arXiv preprint arXiv:2207.08391.

[33] Almansour, S., Yadav, K., Alkwai, L. M., Alghamdi, N. S., Viriyasitavat, W., & Dhiman, G. (2025). Adaptive personalized federated learning with lightweight depthwise convolutional bottleneck network for novel intrusion detection system in internet of vehicles. Scientific Reports, 15(1), 35604.

[34] Zuo, Y., Cox, B., Chen, L. Y., & Decouchant, J. (2024). Asynchronous multi-server federated learning for geo-distributed clients. arXiv preprint arXiv:2406.01439.

[35] Sen, M., Aparna, S., Agarwal, R., & Mohan, C. K. (2025). Overcoming Challenges of Partial Client Participation in Federated Learning: A Comprehensive Review. arXiv preprint arXiv:2506.02887.

[36] Barona López, L. I., & Borja Saltos, T. (2025). Heterogeneity challenges of federated learning for future wireless communication networks. Journal of Sensor and Actuator Networks, 14(2), 37.

[37] Solans, D., Heikkila, M., Vitaletti, A., Kourtellis, N., Anagnostopoulos, A., & Chatzigiannakis, I. (2024). Non-iid data in federated learning: A systematic review with taxonomy, metrics, methods, frameworks and future directions. arXiv preprint arXiv:2411.12377.

[38] Hussain, I., Zakaria, N. H., & Azzali, F. (2025). AN ENHANCED FUZZY LOGIC-BASED TECHNIQUE FOR SEAMLESS HANDOFF DECISION IN CELLULAR NETWORKS. *Spectrum of Engineering Sciences*, 1021-1031.

[39] Mahdi, O. A., Pardede, E., Bevinakoppa, S., & Ali, N. (2025). Federated Learning Under Concept Drift: A Systematic Survey of Foundations, Innovations, and Future Research Directions. Electronics, 14(22), 4480.

[40] Barona López, L. I., & Borja Saltos, T. (2025). Heterogeneity challenges of federated learning for future wireless communication networks. Journal of Sensor and Actuator Networks, 14(2), 37.

[41] Khajehali, N., Yan, J., Chow, Y. W., & Fahmideh, M. (2023). A comprehensive overview of IoT-based federated learning: Focusing on client selection methods. Sensors, 23(16), 7235.

[42] Wang, H., Liu, X., Zhong, X., Chen, L., Liu, F., & Zhang, W. (2025). Multimodal Online Federated Learning with Modality Missing in Internet of Things. arXiv preprint arXiv:2505.16138.

[43] Nazar, S., & Nordin, N. R. M. (2020, March). Impact of Teacher Training Programs on Teachers' Performance: A Case of In-Service English LanguageSecondary School Teachers in Pakistan. In *Linguistic Forum–A Journal of Linguistics* (Vol. 2, No. 1, pp. 19-22).

[44] Gokcen, A., & Boyaci, A. (2025). Robust Federated Learning with Confidence-Weighted Filtering and GAN-Based Completion under Noisy and Incomplete Data. arXiv preprint arXiv:2505.09733.

[45] Chen, J., Zhao, Y., Li, Q., Feng, X., & Xu, K. (2023). FedDef: Defense against gradient leakage in federated learning-based network intrusion detection systems. *IEEE Transactions on Information Forensics and Security*, *18*, 4561-4576.

[46] Gao, J., Tang, M., Wang, W., Routh, T., & Campbell, B. (2025, May). Atlas: Ensuring Accuracy for Privacy-Preserving Federated IoT Applications. In Proceedings of the ACM/IEEE 16th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2025) (pp. 1-11).

[47] Abbasi Tadi, A., Dayal, S., Alhadidi, D., & Mohammed, N. (2023). Comparative analysis of membership inference attacks in federated and centralized learning. *Information*, *14*(11), 620.

[48] Jimenez-Gutierrez, D. M., Falkouskaya, Y., Hernandez-Ramos, J. L., Anagnostopoulos, A., Chatzigiannakis, I., & Vitaletti, A. (2026). On the Security and Privacy of Federated Learning: A Survey with Attacks, Defenses, Frameworks, Applications, and Future Directions. *Information Fusion*, 104155.

[49] Badhib, A., Alshehri, S., & Cherif, A. (2025). IoT Authentication in Federated Learning: Methods, Challenges, and Future Directions. *Sensors*, *25*(24), 7619.

[50] Zhang, X., Luo, Y., & Li, T. (2025). A review of research on secure aggregation for federated learning. *Future Internet*, *17*(7), 308.

[51] Badhib, A., Alshehri, S., & Cherif, A. (2025). IoT Authentication in Federated Learning: Methods, Challenges, and Future Directions. *Sensors*, *25*(24), 7619.

[52] Thamilarasu, G., & Dunham, C. (2025). SpaceTime: A Deep Similarity Defense Against Poisoning Attacks in Federated Learning. *Big Data and Cognitive Computing*, *9*(12), 313.

[53] Pecherle, G. D., Győrödi, R. Ş., & Győrödi, C. A. (2025). Federated Learning-Based Intrusion Detection in Industrial IoT Networks. *Future Internet*, *18*(1), 2.

[54] Jimenez-Gutierrez, D. M., Falkouskaya, Y., Hernandez-Ramos, J. L., Anagnostopoulos, A., Chatzigiannakis, I., & Vitaletti, A. (2026). On the Security and Privacy of Federated Learning: A Survey with Attacks, Defenses, Frameworks, Applications, and Future Directions. *Information Fusion*, 104155.