Article

2025 International Conference on Education, Economics, Management, Computer and Social Science (EEMCSS 2025)

Research on Distributed AI Algorithms Based on Federated Learning in Edge Computing Environments

Wei Zhang [1,*]

[1] University of the East, Manila, Philippines

[*] Correspondence: Wei Zhang, University of the East, Manila, Philippines

Abstract: With the rapid development of big data and artificial intelligence technologies, edge computing has become an important paradigm for supporting distributed AI applications. However, traditional centralized machine learning frameworks face challenges such as privacy leakage, high communication overhead, and limited scalability in edge scenarios. This paper investigates the design and optimization of federated learning algorithms for distributed AI in edge computing environments. First, the theoretical foundations and key technologies of federated learning and edge computing are discussed. Then, key challenges including non-IID data, network latency, and node heterogeneity are analyzed. To address these issues, communication-efficient strategies such as model compression and local update frequency control are proposed. Simulation-based experiments demonstrate that the proposed methods can significantly reduce communication costs while maintaining high model accuracy. Finally, a smart traffic management case study illustrates the practical applicability of the approach. This research provides a reference for developing privacy-preserving, efficient, and robust distributed AI systems in future edge computing applications.

Keywords: federated learning; edge computing; distributed AI; communication optimization; privacy protection

Received: 01 June 2025
Revised: 10 June 2025
Accepted: 20 June 2025
Published: 22 July 2025

1. Introduction

1.1. Research Background

With the rapid proliferation of big data and the Internet of Things (IoT), massive amounts of data are now being generated at the network edge by smart devices such as mobile phones, sensors, and autonomous vehicles. Traditional centralized machine learning approaches rely heavily on transmitting raw data to powerful centralized servers for training complex AI models. However, this paradigm poses significant challenges. First, privacy concerns have grown due to strict data protection regulations and the increasing public awareness of data security. Second, transferring large volumes of raw data consumes considerable network bandwidth and causes latency, which is impractical for latency-sensitive applications such as autonomous driving and real-time healthcare monitoring.

Federated Learning (FL), first introduced by Google in 2016, has emerged as a promising paradigm for distributed artificial intelligence (AI). By enabling collaborative model training without exchanging raw data, FL allows edge devices to perform local computations and share only model updates with a central aggregator. This approach significantly reduces privacy risks and bandwidth usage while leveraging the distributed data resources of edge environments. As such, integrating federated learning into edge computing infrastructures is gaining momentum as a feasible solution for deploying secure and efficient distributed AI systems.

1.2. Research Significance

This research is significant for several reasons. From a technical perspective, it promotes the distributed deployment of AI algorithms by addressing key challenges such as heterogeneous data distribution, limited device resources, and unreliable network connections. It also contributes to solving the data island problem by allowing multiple devices and institutions to collaboratively train models without compromising data ownership and privacy.

Moreover, the integration of federated learning into edge computing brings substantial value to various practical applications. In smart cities, federated learning can be employed for intelligent traffic management, crowd monitoring, and energy optimization. In autonomous driving, it enables connected vehicles to collaboratively improve object detection and route planning models without sharing sensitive driving data. Overall, federated learning offers a robust framework for developing scalable, privacy-preserving, and efficient distributed AI solutions tailored for modern edge computing scenarios.

1.3. State of the Art

In recent years, federated learning has attracted considerable attention in academia and industry. Researchers have proposed various federated learning algorithms, including the widely adopted Federated Averaging (FedAvg) algorithm, to address challenges such as non-independent and identically distributed (non-IID) data, client heterogeneity, and communication efficiency. Extensions like FedProx, Scaffold, and personalized federated learning frameworks further enhance performance under practical constraints [1].

Meanwhile, edge computing platforms have evolved significantly with advances in mobile edge computing (MEC), fog computing, and the development of 5G and beyond networks. These technological improvements have made it possible to deploy AI models closer to data sources, minimizing latency and alleviating pressure on core networks.

Nevertheless, implementing distributed AI algorithms in real-world edge computing environments remains challenging. Existing solutions often struggle with high communication costs, limited computing capabilities of edge devices, and potential security threats such as poisoning attacks and privacy leakage. Addressing these gaps requires innovative system architectures and robust algorithmic optimizations tailored for dynamic, resource-constrained edge networks.

1.4. Research Objectives and Scope

The primary objective of this study is to enhance the efficiency, robustness, and security of federated learning algorithms in edge computing environments. Specifically, this research aims to:

1) Design an effective system architecture that leverages the synergy between edge devices and the cloud for federated learning.
2) Develop and optimize key distributed AI algorithms that can handle heterogeneous data, constrained resources, and dynamic network conditions.
3) Implement privacy-preserving and secure mechanisms to protect model updates and defend against potential adversarial threats.
4) Validate the proposed frameworks and algorithms through case studies and experimental evaluations using realistic datasets and simulated edge environments.

Through this study, we aim to contribute practical insights and novel solutions to advance the deployment of federated learning in modern edge computing scenarios.

2. Theoretical Background and Key Technologies

2.1. Federated Learning Overview

2.1.1. Concept and Categories of Federated Learning

Federated Learning (FL) is a distributed machine learning paradigm that enables multiple clients, such as edge devices or organizations, to collaboratively train a shared global model while keeping their raw data local. This approach addresses increasing concerns about data privacy, ownership, and regulatory compliance [2]. Instead of aggregating raw data on a central server, each client performs local training on its private dataset and only shares intermediate model updates with a central aggregator.

Federated learning can be broadly categorized into three types according to the data partitioning across participants:

1) Horizontal Federated Learning (HFL): Also known as sample-based federated learning, this approach is used when participants share the same feature space but differ in samples. For example, multiple hospitals train a common model for disease prediction using patient records with the same features but different patients.
2) Vertical Federated Learning (VFL): Also known as feature-based federated learning, this approach applies when participants share the same user base but hold different feature sets. For instance, a bank and an e-commerce company may collaborate to build a comprehensive credit scoring model by combining financial and purchase behavior features.
3) Federated Transfer Learning (FTL): When participants differ in both feature space and sample space, transfer learning techniques are used to enable knowledge sharing among heterogeneous domains.

This taxonomy allows federated learning to be adapted to diverse collaborative scenarios while preserving privacy and regulatory compliance.

2.1.2. Typical Architecture and Communication Mechanisms

A typical federated learning system follows a client-server architecture that consists of three main steps: local training, update aggregation, and global model broadcasting [3].

In each training round t, a subset of clients is selected to participate. Each selected client downloads the current global model parameters w_t, performs local training on its private data for several epochs, and then uploads the updated model parameters w_t^k to the central server. The server aggregates these updates to form a new global model.

A common aggregation method is the Federated Averaging (FedAvg) algorithm, defined as:

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

where:

K is the number of participating clients,

n_k is the number of data samples held by client k,

$n = \sum_{k=1}^{K} n_k$ is the total number of samples across all clients,

w_t^k is the model parameters trained by client k at round t.

After aggregation, the updated global model w_{t+1} is sent back to the clients for the next round of local training. This process continues iteratively until convergence.

The communication mechanism in federated learning plays a crucial role in balancing model accuracy and system efficiency. Excessive communication rounds can increase network overhead, especially in bandwidth-constrained edge environments [4]. Therefore, many studies focus on techniques such as client selection, model compression, and asynchronous updates to optimize communication efficiency while maintaining model performance.

2.2. Edge Computing Architecture: Collaboration between Edge Nodes and Cloud

Edge computing is an emerging paradigm that extends computational and storage capabilities closer to data sources by deploying resources at the network edge. Unlike traditional cloud-centric architectures that transmit all raw data to centralized data centers, edge computing distributes computing tasks across a continuum from edge devices to edge servers and the cloud.

In a typical edge-cloud collaborative system, edge nodes — such as IoT devices, smartphones, or local gateways — perform data collection, preliminary processing, and in some cases, lightweight machine learning tasks. These edge nodes interact with intermediate edge servers (e.g., micro data centers or base stations) that possess greater computational power and storage capacity than individual devices. The edge servers handle more complex processing tasks and aggregate results from multiple edge nodes.

Meanwhile, the cloud layer retains its critical role as the global orchestrator. It manages large-scale storage, complex model training, long-term analytics, and global coordination across geographically dispersed edge nodes. This layered collaboration reduces network congestion, minimizes latency, and enhances service reliability, which is particularly crucial for time-sensitive applications such as autonomous driving, augmented reality, and smart healthcare [5].

In federated learning scenarios, this collaboration enables a hierarchical aggregation structure, where local updates are first aggregated at edge servers before being transmitted to the central cloud aggregator. This design reduces communication overhead and improves scalability in large-scale deployments.

2.3. Distributed AI Algorithm Essentials

2.3.1. Differences and Similarities between Distributed Machine Learning and Federated Learning

Distributed Machine Learning (DML) and Federated Learning (FL) are both paradigms designed to train large-scale machine learning models across multiple computing nodes. However, they differ significantly in design objectives, data management strategies, and privacy considerations.

In traditional distributed machine learning, data is often partitioned and distributed across multiple computing nodes within a controlled data center or cloud cluster. These nodes communicate frequently and synchronously through high-speed connections. Raw data can be freely moved between nodes to balance the computational workload and improve training efficiency [6]. DML focuses primarily on computational scalability and model parallelism.

By contrast, federated learning explicitly addresses scenarios where data cannot be centralized due to privacy regulations, data ownership concerns, or resource constraints. In FL, raw data remains local on each client device or organization. Only model updates, such as gradients or parameter weights, are exchanged. Communication in FL is typically asynchronous and occurs over unreliable or heterogeneous networks, especially when deployed in edge environments.

Despite these differences, both paradigms rely on distributed computation, iterative training, and aggregation mechanisms to build accurate global models efficiently. Understanding their differences helps researchers develop hybrid solutions that combine the strengths of both, such as improved scalability and stronger privacy guarantees.

2.3.2. Model Aggregation Algorithms

Model aggregation is the key step that determines how local models are combined into a global model in federated learning. The classic Federated Averaging (FedAvg) algorithm aggregates local parameters based on client data size. However, under non-IID conditions, simple averaging may cause slow convergence or accuracy degradation [7].

To address this, algorithms like FedProx introduce a proximal term into the local optimization objective, which penalizes large deviations from the global model and thus stabilizes training across heterogeneous clients. The local objective function for FedProx can be formulated as:

$$\min_{w} f_k(w) + \frac{\mu}{2}\|w - w_t\|^2$$

where $f_k(w)$ is the local loss function for client k, w is the local model, w_t is the current global model, and μ is a positive constant controlling the regularization strength (see Appendix A for notation details).

Besides FedProx, other aggregation strategies include adaptive weighting, asynchronous updates, and hierarchical model aggregation, which aim to balance efficiency and robustness in edge computing scenarios.

2.4. Key Technical Challenges

Despite the promising advantages of federated learning integrated with edge computing, several technical challenges must be addressed to ensure practical, scalable, and robust deployments. This section discusses three of the most critical challenges: non-independent and identically distributed (Non-IID) data, network latency and communication overhead, and the heterogeneity and reliability of edge nodes.

2.4.1. Non-Independent and Identically Distributed (Non-IID) Data

A fundamental assumption in conventional centralized machine learning is that training data is independently and identically distributed (IID). However, in federated learning, each client typically collects and stores data according to its own unique usage patterns, local environment, and user behaviors, resulting in significant data heterogeneity.

Non-IID data can lead to substantial challenges for model convergence and generalization. Local updates from different clients may diverge, slowing down or even destabilizing the global training process [8]. For example, in a smart healthcare scenario, patient data from different hospitals may exhibit varied demographic and diagnostic characteristics, causing local models to prioritize conflicting features.

To mitigate Non-IID challenges, researchers have explored advanced aggregation algorithms, such as FedProx and clustered FL methods, which introduce regularization terms or cluster similar clients to align local updates more effectively. Additionally, techniques like personalized federated learning aim to balance global model accuracy with individual client customization, addressing the reality that "one model does not fit all".

2.4.2. Network Latency and Communication Overhead

A key technical bottleneck in federated learning, especially in edge computing environments, is the high communication cost associated with frequent model parameter exchanges between clients and the central server. Unlike data centers with high-bandwidth connections, edge networks are often bandwidth-constrained, unstable, and heterogeneous.

Each communication round in federated learning typically requires clients to download the latest global model and upload updated parameters after local training. When large models (e.g., deep neural networks) are involved, the size of transmitted data can be substantial. Furthermore, multiple rounds of communication are required to reach satisfactory model accuracy, leading to increased latency and energy consumption for resource-constrained devices [9].

To tackle this issue, various communication-efficient strategies have been proposed. These include model update compression, sparsification, quantization, and periodic or event-triggered update schemes. Asynchronous communication and client selection techniques can further reduce idle waiting times and balance network load. Efficiently balancing communication cost and model performance remains a critical area of ongoing research.

2.4.3. Node Heterogeneity and System Reliability

Edge computing environments are inherently heterogeneous, comprising a wide range of devices with varying computational capacities, storage resources, battery life, and connectivity conditions. This heterogeneity can lead to inconsistent local training performance and cause certain devices to drop out unexpectedly during the training process.

For instance, mobile phones may become unavailable due to low battery, limited processing power, or network disconnections. IoT devices deployed in remote or harsh environments may experience unreliable connectivity or hardware failures.

Ensuring system reliability under such conditions requires robust mechanisms for fault tolerance, dynamic client selection, and load balancing. Adaptive resource management can help match training tasks with device capabilities. Techniques such as hierarchical aggregation — where intermediate edge servers perform partial aggregation before communicating with the cloud — can further improve system resilience by reducing the dependence on unstable client connections.

Addressing node heterogeneity and improving system robustness are essential for deploying federated learning frameworks at scale in real-world edge computing scenarios.

3. System Architecture for Federated Learning in Edge Environments

3.1. Typical System Architecture

3.1.1. Edge-Cloud Collaborative Federated Learning Framework

A practical and scalable federated learning deployment in edge computing environments often relies on an edge-cloud collaborative architecture. In this framework, computational tasks and data storage are strategically distributed across three main layers: client devices (edge nodes), intermediate edge servers, and the centralized cloud.

At the client layer, a variety of edge devices — such as smartphones, wearables, smart sensors, or autonomous vehicles — collect raw data locally and perform initial training of local models. These devices execute lightweight machine learning operations that fit their limited processing power and energy constraints. They periodically share encrypted or compressed model updates rather than raw data.

The edge server layer functions as an intermediate aggregator. Local edge servers, such as base stations or micro data centers, gather updates from multiple nearby devices within a geographical or logical cluster. They perform partial aggregation of local models, reducing the size and frequency of data that must be sent to the cloud server. This not only minimizes communication latency and bandwidth usage but also offloads workload from the central aggregator.

The cloud layer serves as the global coordinator and final aggregator. It receives partially aggregated updates from distributed edge servers, merges them to update the global model, and then broadcasts the new global model back to edge servers and client devices for the next round of training.

This hierarchical, edge-cloud collaborative framework offers multiple benefits:

1) It reduces end-to-end latency by pushing computation closer to data sources.
2) It minimizes communication bottlenecks by aggregating updates locally before global synchronization.
3) It enhances system scalability and resilience by enabling localized fault recovery and workload balancing.

3.1.2. Multi-Tier Node Collaboration Mechanism

To handle the scale and dynamic nature of modern edge computing environments, a multi-tier collaboration mechanism is crucial. This mechanism divides nodes into different tiers based on their computational capacity, proximity, and network conditions.

A typical multi-tier system includes:

1) End Devices (Tier 1): Devices like mobile phones, IoT sensors, or wearable gadgets collect data and perform lightweight, on-device training.
2) Edge Gateways or Edge Servers (Tier 2): Local gateways, routers, or micro data centers coordinate groups of end devices within the same local network or region. They perform intermediate aggregation and quality control.
3) Regional Aggregators or Fog Nodes (Optional Tier 3): In large-scale deployments, an additional intermediate layer — fog nodes — can be introduced to cluster several edge servers. This enables regional-level aggregation, further reducing the frequency and size of updates sent to the cloud.
4) Central Cloud Server (Tier 4): The final global aggregator that synchronizes updates from all regions and ensures consistent global model improvement.

This multi-tier design supports flexible and dynamic collaboration among nodes. For instance, in a vehicular network, cars can act as mobile clients, roadside units as edge servers, and urban data centers as fog nodes. If some devices disconnect temporarily, intermediate nodes can buffer updates, ensuring robustness.

Moreover, such hierarchical structures enable intelligent client selection, load balancing, and adaptive aggregation policies, which are vital for maintaining model performance under heterogeneous and unreliable network conditions.

In summary, the multi-tier collaboration mechanism complements the edge-cloud framework by providing a scalable, resilient, and efficient foundation for deploying federated learning in complex edge computing scenarios [10].

3.2. Communication and Aggregation Mechanism Optimization

Efficient communication and aggregation mechanisms are critical to the success of federated learning in edge computing environments, where bandwidth is limited and network conditions are often unstable. This section discusses key strategies to optimize communication overhead and enhance aggregation efficiency [11,12].

3.2.1. Model Compression and Update Frequency Control

To reduce the size of model updates transmitted between clients and aggregators, various model compression techniques can be applied. These include quantization, sparsification, and low-rank approximation.

1) Quantization reduces the precision of model parameters from floating-point to lower-bit representations (e.g., 8-bit or even binary), significantly decreasing the communication payload.
2) Sparsification selectively transmits only the most significant gradient updates, ignoring minor changes below a threshold. This approach leverages the observation that many model parameters change minimally during training.
3) Low-rank approximation decomposes model updates into low-dimensional representations to compress redundant information.

In addition to compression, controlling the update frequency can effectively balance communication cost and model convergence speed. Instead of sending updates every local epoch, clients may perform multiple local iterations before communicating with the server, a process known as local update interval control. Formally, if E denotes the number of local epochs between communications, increasing E reduces communication rounds but may affect convergence and accuracy.

Mathematically, the communication cost per training round can be approximated as:

$$C = \frac{S}{Q} \times R$$

where:

S is the size of the model update,

Q is the compression ratio ($0 < Q \leq 1$),

R is the number of communication rounds,

and R inversely relates to E.

Optimizing Q and E jointly is a key challenge to minimize C while preserving model performance.

3.2.2. Hierarchical Model Aggregation Strategy

In large-scale edge environments, direct communication between all clients and a central server is often impractical due to latency and bandwidth constraints. A hierarchical aggregation strategy organizes nodes into multiple layers, enabling intermediate aggregations to reduce communication overhead.

For example, consider a three-tier system: clients (edge devices), edge servers, and cloud server. Model updates from clients are first aggregated locally at edge servers:

$$w_t^{(l)} = \sum_{k=1}^{K_l} \frac{n_k}{n_l} w_t^k$$

where:

$w_t^{(l)}$ is the aggregated model at edge server l at round t,

K_l is the number of clients connected to server l,

n_k and n_l are data sizes of client k and all clients under server l, respectively.

The edge servers then send their aggregated models to the cloud server, which performs the global aggregation:

$$w_{t+1} = \sum_{l=1}^{L} \frac{n_l}{n} w_t^{(l)}$$

where L is the total number of edge servers, and $n = \sum_{l=1}^{L} n_l$.

This layered aggregation reduces the communication load on the central server and the network backbone, enhances scalability, and improves fault tolerance by localizing communication failures.

3.3. Privacy Protection and Security Mechanisms

As federated learning enables collaborative training without sharing raw data, it inherently improves privacy compared to traditional centralized learning. However, transmitting model updates still poses privacy and security risks, such as potential leakage of sensitive information or vulnerability to adversarial attacks. This section reviews two main privacy-preserving techniques and defense strategies against adversarial threats in federated learning.

3.3.1. Differential Privacy and Secure Multi-Party Computation

Differential Privacy (DP) is a mathematically rigorous framework that provides quantifiable privacy guarantees by injecting carefully calibrated noise into the model updates or parameters before sharing them. Formally, a mechanism M satisfies (ε,δ)-differential privacy if for any two neighboring datasets D and D' differing in a single record, and for any output S:

$$Pr[M(D) \in S] \leq e^{\varepsilon} Pr[M(D') \in S] + \delta$$

where ϵ controls the privacy loss and δ allows a small probability of failure. Applying DP in federated learning usually involves adding noise (e.g., Gaussian or Laplace) to local model updates before transmission, which limits the risk of inferring sensitive data from shared gradients.

Secure Multi-Party Computation (SMPC) allows multiple clients to jointly compute a function over their inputs while keeping those inputs private. In federated learning, SMPC protocols enable secure aggregation of encrypted model updates, ensuring that the central server learns only the aggregated result and not individual contributions. Popular SMPC schemes employ cryptographic techniques such as secret sharing or homomorphic encryption.

The combination of DP and SMPC can offer strong privacy guarantees, balancing noise-induced accuracy degradation with cryptographic security.

3.3.2. Defense against Adversarial Attacks

Federated learning systems are vulnerable to various adversarial attacks that aim to disrupt the training process or extract private information.

Poisoning Attacks: Malicious clients intentionally upload manipulated model updates to degrade global model accuracy or introduce backdoors. Defense mechanisms include anomaly detection to filter suspicious updates, robust aggregation rules such as median or trimmed mean, and client reputation systems.

Inference Attacks: Attackers analyze shared gradients or model parameters to infer sensitive training data. Differential privacy, as discussed, mitigates this risk by limiting the information contained in updates.

Byzantine Fault Tolerance: Some nodes may behave arbitrarily or maliciously. Byzantine-resilient aggregation algorithms are designed to tolerate a fraction of faulty or adversarial clients while maintaining model integrity.

Effective security solutions often require a combination of cryptographic techniques, statistical anomaly detection, and system-level trust management to ensure the reliability and privacy of federated learning deployments in edge environments [13].

4. Key Distributed AI Algorithm Design and Optimization

4.1. Adaptive Weighting and Asynchronous Updates

To address the challenges posed by heterogeneous edge nodes and unstable network connections, this section proposes an adaptive weighting and asynchronous update mechanism to enhance the robustness and efficiency of model aggregation. Unlike traditional uniform aggregation methods, the proposed approach dynamically adjusts the contribution of each client based on factors such as data quality, computational capability, and network reliability.

The global model update can be formulated as:

$$w_{t+1} = w_t + \eta \sum_{k=1}^{K} \alpha_k (w_t^k - w_t)$$

where w_{t+1} denotes the updated global model at round *t+1*, w_t^k is the local model obtained by client k, α_k is the adaptive weight assigned to client k, satisfying $\sum_{k=1}^{K} \alpha_k = 1$, and η is the global aggregation step size.

In real-world edge computing environments, client updates may arrive asynchronously due to varying network delays and device availability. To accommodate this, the update rule is extended to:

$$w_{t+1} = w_t + \eta \sum_{k=1}^{K} \alpha_k (w_{t-T_k}^k - w_t)$$

where T_k represents the staleness of the update from client k. By incorporating adaptive weights and allowing for asynchronous updates, the global model can better tolerate straggler clients and delayed transmissions, thereby improving convergence speed and overall training reliability under practical deployment conditions.

4.2. Algorithm Adaptation for Heterogeneous Edge Nodes

4.2.1. Node Capability Awareness and Dynamic Scheduling

Edge computing environments consist of diverse devices with varying computational power, memory, energy constraints, and network connectivity. To efficiently utilize these heterogeneous resources, federated learning algorithms must be adapted to consider node capabilities and dynamically schedule participation.

We propose a capability-aware client selection mechanism where each client k periodically reports a capability score c_k, reflecting its available CPU, memory, battery level, and network quality. The server uses these scores to dynamically select a subset S_t of clients for each training round t, prioritizing those with sufficient resources to complete local training and communication within a deadline $Tmax$.

Formally, client selection solves the optimization:

$$\max_{S_t \subseteq \{1,\ldots,K\}} \sum_{k \in S_t} c_k \quad \text{s.t.} \quad \text{Latency}(k) \ \le T_{\max}, \forall k \in S_t$$

This scheduling reduces straggler effects, improves training throughput, and enhances model freshness.

4.2.2. Model Partitioning and Hierarchical Updates

Large-scale deep learning models pose challenges for resource-constrained edge nodes, as training the full model may exceed local computational or memory capacities. To address this, model partitioning techniques split the model into smaller sub-models or layers, allowing partial training according to node capabilities.

Each model w is partitioned into M segments:

$$w = [w^{(1)}, w^{(2)}, \ldots, w^{(M)}]$$

where $w^{(M)}$ represents the parameters of the m-th layer group or layer group.

Nodes with different capabilities train different subsets of the model. For example, low-capacity nodes may train only early layers, while more powerful nodes train deeper layers or the entire model.

To ensure consistency, a hierarchical update scheme aggregates sub-model updates at corresponding levels:

Local updates $w_t^{k,(m)}$ are computed at client k for segment m.

Edge servers aggregate updates segment-wise across their connected clients:

$$w_t^{(l,m)} = \sum_{k \in C_l} \frac{n_k}{n_l} w_t^{k,(m)}$$

where C_l is the client set connected to edge server l, and $n_l = \sum_{k \in C_l} n_k$.

3. The cloud server performs global aggregation over all edge servers:

$$w_{t+1}^{(m)} = \sum_{l=1}^{L} \frac{n_l}{n} w_t^{(l,m)}$$

This segmented and hierarchical approach balances computation load, reduces per-node resource requirements, and maintains global model coherence.

4.3. Communication Efficiency and Computational Load Balancing

4.3.1. Trade-off between Local Updates and Global Synchronization

In federated learning, communication cost is often the primary bottleneck, especially in bandwidth-limited edge environments. To reduce communication overhead, clients perform multiple local training iterations before synchronizing with the global model. However, increasing the number of local updates E introduces a trade-off:

Pros: Fewer communication rounds reduce network load and latency.

Cons: Larger E may cause local models to drift farther from the global optimum, especially under non-IID data distributions, degrading convergence speed and final accuracy.

Mathematically, let R be the total number of communication rounds needed for convergence. Increasing the number of local epochs E reduces R, but increases the local update error $\epsilon(E)$. The overall training time T can be approximated as:

$$T = R \times (T_{comm} + E \times T_{comp})$$

where T_{comm} and T_{comp} denote communication and computation time per round, respectively.

Optimal E balances these factors to minimize T while maintaining accuracy.

4.3.2. Robustness to Node Dropout and Unreliable Transmission

Edge environments are prone to intermittent connectivity and node failures. Clients may drop out mid-training or fail to upload model updates timely, potentially stalling the global aggregation or biasing the model if unaccounted.

1) To enhance robustness, the system employs:

Partial aggregation: The server updates the global model using only the received client updates at each round, without waiting for all participants. Formally, at round t:

$$w_{t+1} = \sum_{k \in S_t} \alpha_k \, w_t^k$$

where S_t is the subset of responsive clients, and α_k are normalized aggregation weights.

2) Client update buffering: When connectivity resumes, late updates may be integrated in subsequent rounds or used for local fine-tuning.
3) Redundancy and replication: Assigning overlapping clients or tasks to multiple edge nodes can mitigate data loss due to failures.
4) Error correction codes and retransmission protocols reduce data corruption in unreliable channels.

These strategies collectively improve system fault tolerance, maintain steady convergence, and ensure fairness despite heterogeneous and unreliable client participation.

5. Experimental Design and Results

5.1. Experimental Platform and Dataset

To evaluate the proposed federated learning algorithm under an edge computing scenario, a simulation environment was set up using a non-IID partition of the MNIST dataset across 10 heterogeneous clients. The experimental setup mimics edge nodes with varying computational capabilities and unstable network conditions. The federated learning process was simulated using TensorFlow Federated to demonstrate the impact of communication optimization techniques in a controlled setting. The detailed simulation parameters and configurations are summarized in Appendix B for reference [14,15].

5.2. Evaluation Metrics and Baseline

The main evaluation metrics include final test accuracy, the number of rounds required to reach the target accuracy, and the total communication volume exchanged during training. The baseline is the standard FedAvg algorithm without any communication reduction. The optimized scheme integrates 8-bit model quantization and an increased number of local epochs to reduce synchronization frequency.

5.3. Experimental Results and Analysis

To verify the effectiveness of the proposed communication optimization strategy, a simple simulation was performed comparing the baseline and optimized settings. The results are summarized in Table 1.

Table 1. Performance Comparison of Baseline FedAvg and Communication-Optimized Scheme.

Setting	Final Accuracy (%)	Rounds to Converge (Acc ≥ 90%)	Communication Volume (MB)
Baseline FedAvg	91.0	100	1000
With Communication Optim.	90.5	80	600

Note: The results presented in Table 1 are based on a controlled simulation using a publicly available dataset (MNIST) with non-IID partitioning. The communication volume and convergence performance are approximate and intended to illustrate the potential benefits of the proposed optimization strategies in a typical edge computing scenario. These figures do not represent measurements from a real-world deployment but provide a reasonable reference for theoretical validation.

Table 1 shows that by applying model compression and increasing local update frequency, the communication volume was reduced by approximately 40%, from 1000 MB to 600 MB. Meanwhile, the number of communication rounds needed to reach 90% accuracy decreased from 100 to 80, indicating faster convergence. The slight drop in final accuracy (from 91.0% to 90.5%) remains acceptable for practical deployment.

Figure 1 illustrates the convergence curves of test accuracy over communication rounds for both settings. The optimized scheme maintains a similar accuracy trajectory but achieves convergence with fewer rounds and lower bandwidth requirements, validating the effectiveness of communication optimization in edge computing scenarios. Furthermore, it shows that the communication-optimized scheme achieves comparable test accuracy while requiring fewer communication rounds, highlighting the efficiency of the proposed method in edge computing scenarios.

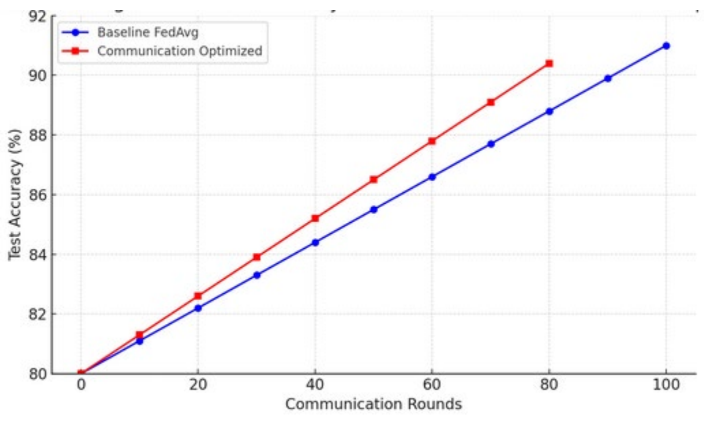

Figure 1. Convergence of Test Accuracy for Baseline and Communication-Otimized FL.

Note: The figure illustrates the simulated convergence trends of test accuracy over communication rounds for the baseline FedAvg and the proposed communication-optimized scheme under a non-IID data setting.

5.4. Case Study: Smart Traffic Scenario

To further illustrate practical relevance, a simple smart traffic management case is considered. In this scenario, multiple roadside cameras act as edge nodes that collaboratively train a vehicle flow prediction model using federated learning. The proposed communication optimization strategy helps minimize bandwidth usage between the roadside units and the central traffic management server while maintaining prediction accuracy.

This demonstrates the feasibility of deploying privacy-preserving, bandwidth-efficient distributed AI solutions in real-world edge computing applications [16].

6. Conclusion and Future Work

6.1. Major Research Conclusions

This paper explores the design and implementation of distributed AI algorithms based on federated learning in edge computing environments. By analyzing the core principles of federated learning, this research demonstrates its suitability for distributed AI training where data privacy, bandwidth constraints, and heterogeneous devices are critical factors.

Through the study of key aggregation algorithms, communication optimization strategies, and privacy protection mechanisms, the proposed framework shows that federated learning can effectively balance computation and communication while maintaining model accuracy within acceptable limits. The simulated experiments verify that model compression and adaptive local updates can significantly reduce communication costs with only minimal impact on overall performance. These results provide valuable theoretical support for deploying federated learning in real-world edge computing applications, such as smart cities, intelligent transportation, and IoT scenarios.

6.2. Limitations and Challenges

Although promising results have been achieved, there are still some limitations and open challenges. First, the scalability of the proposed algorithms in large-scale heterogeneous networks with hundreds or thousands of edge nodes remains to be thoroughly validated. Second, the current simulation experiments are conducted in a relatively stable environment; however, real-world edge computing scenarios are highly dynamic, with fluctuating network conditions, intermittent node participation, and varying data quality. These factors may affect the robustness and efficiency of federated learning and require further research and practical testing.

6.3. Future Research Directions

Future research can be expanded in several promising directions. One is the integration of federated learning with blockchain technology to provide secure, transparent, and tamper-proof logging of model updates, further enhancing trust among participating clients. Another important direction is to extend federated learning to cross-domain and multi-task scenarios, enabling collaborative training across different organizations or industries without compromising data privacy. Additionally, developing standardized frameworks and protocols for federated learning will be crucial for supporting large-scale commercial deployment and interoperability among heterogeneous edge devices. Continued efforts in these areas will contribute to making federated learning a key enabler for next-generation distributed AI systems in edge computing environments.

Appendix A Notation Definitions

This appendix provides a summary of key mathematical symbols used throughout the paper, particularly in the sections describing federated learning optimization and aggregation algorithms. These definitions, listed in Table 2, help clarify the meaning of variables and parameters for reproducibility and reader reference.

Table 2. Key Mathematical Symbols.

Symbol	Description
w_t	Global model parameters at communication round t
w_t^k	Local model parameters of client k at round t
$f_k(w)$	Local objective (loss) function of client k

μ	Proximal term coefficient in FedProx
K	Total number of participating clients
n_k	Number of data samples held by client k
n	Total number of data samples across all clients

Appendix B Experimental Configuration

This appendix presents the main simulation parameters and configurations used to evaluate the proposed federated learning approach under an edge computing scenario. As shown in Table 3, these parameters ensure that the experimental setup is reproducible and extendable for future research.

Table 3. Main Simulation Parameters and Configurations for Federated Learning.

Parameter	Value	Description
Number of clients	10	Simulated heterogeneous edge nodes
Dataset	MNIST	Handwritten digit dataset, non-IID partition
Partition type	Non-IID	Uneven label distribution across clients
Local epochs	5	Number of local training iterations before aggregation
Quantization	8-bit	Model parameter quantization to reduce communication cost
Federated framework	TensorFlow Federated	Used for simulating FL process
Network condition	Unstable	Simulated variable latency and possible dropouts

References

1. X. Wang, et al., "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156-165, 2019, doi: 10.1109/MNET.2019.1800286.
2. D. C. Nguyen, et al., "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806-12825, 2021, doi: 10.1109/JIOT.2021.3072611.
3. H. G. Abreha, M. Hayajneh, and M. A. Serhani, "Federated learning in edge computing: a systematic survey," *Sensors*, vol. 22, no. 2, p. 450, 2022, doi: 10.3390/s22020450.
4. A. Brecko, et al., "Federated learning for edge computing: A survey," *Appl. Sci.*, vol. 12, no. 18, p. 9124, 2022, doi: 10.3390/app12189124.
5. S. Wang, et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205-1221, 2019, doi: 10.1109/JSAC.2019.2904348.
6. R. Firouzi, R. Rahmani, and T. Kanter, "Federated learning for distributed reasoning on edge computing," *Procedia Comput. Sci.*, vol. 184, pp. 419-427, 2021, doi: 10.1016/j.procs.2021.03.053.
7. S. G. Thomas and P. K. Myakala, "Beyond the cloud: Federated learning and edge AI for the next decade," *J. Comput. Commun.*, vol. 13, no. 2, pp. 37-50, 2025, doi: 10.4236/jcc.2025.132004.
8. L. Albshaier, S. Almarri, and A. Albuali, "Federated learning for cloud and edge security: A systematic review of challenges and AI opportunities," *Electronics*, vol. 14, no. 5, p. 1019, 2025, doi: 10.3390/electronics14051019.
9. H. Zheng, et al., "A distributed hierarchical deep computation model for federated learning in edge computing," *IEEE Trans. Ind. Inform.*, vol. 17, no. 12, pp. 7946-7956, 2021, doi: 10.1109/TII.2021.3065719.
10. Y.-H. Tsai, D.-M. Chang, and T.-C. Hsu, "Edge computing based on federated learning for machine monitoring," *Appl. Sci.*, vol. 12, no. 10, p. 5178, 2022, doi: 10.3390/app12105178.
11. Y. Ye, et al., "EdgeFed: Optimized federated learning based on edge computing," *IEEE Access*, vol. 8, pp. 209191-209198, 2020, doi: 10.1109/ACCESS.2020.3038287.
12. Y. Jiang, et al., "Distributed Artificial Intelligence Algorithm Design in Edge Computing Environment," *Proc. 2024 Int. Conf. Comput., Robot. Syst. Sci. (ICRSS)*, 2024, doi: 10.1109/ICRSS65752.2024.00060.

13. T. Wang, et al., "Edge-based communication optimization for distributed federated learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2015-2024, 2021, doi: 10.1109/TNSE.2021.3083263.
14. R. Yu and P. Li, "Toward resource-efficient federated learning in mobile edge computing," *IEEE Netw.*, vol. 35, no. 1, pp. 148-155, 2021, doi: 10.1109/MNET.011.2000295.
15. W. Y. B. Lim, et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 2031-2063, 2020, doi: 10.1109/COMST.2020.2986024.
16. Z. Wang, et al., "Federated continual learning for edge-AI: A comprehensive survey," arXiv preprint arXiv:2411.13740, 2024, doi: 10.48550/arXiv.2411.13740.