Journal of computer science and software applications
https://www.mfacademia.org/index.php/jcssa
ISSN:2377-0430
Vol. 5, No. 5, 2025

Federated Learning for Privacy-Preserving Edge Intelligence: A Scalable Systems Perspective

Elowen Price
University of Southern Queensland, Toowoomba, Australia
elowen.price@usq.edu.au

Abstract:

The rapid proliferation of edge devices and the exponential growth of user-generated data have accelerated the demand for intelligent systems that operate in distributed, resource-constrained, and privacy-sensitive environments. Federated Learning (FL) has emerged as a promising solution to this challenge by enabling collaborative model training across decentralized devices without transferring raw data to a central server. This paper presents a comprehensive systems-level framework for deploying scalable and privacy-preserving FL on heterogeneous edge platforms. We propose a modular architecture that integrates adaptive model compression, dynamic client selection, and secure gradient aggregation under bandwidth and compute constraints. Our design emphasizes fault tolerance, communication efficiency, and adversarial robustness while maintaining inference performance comparable to centralized training. Extensive experiments on CIFAR-10, HAR, and speech datasets using Raspberry Pi and NVIDIA Jetson devices show that our system achieves up to 38% reduction in communication cost and 26% training speed-up, with only a 1.7% accuracy loss compared to centralized baselines. We further demonstrate the system's resilience to client dropout and adversarial data poisoning. This work contributes a practical, extensible platform for real-world FL deployment and offers insights into building future intelligent edge infrastructures.

Keywords:

Federated learning, edge intelligence, system architecture, privacy preservation, distributed optimization, secure aggregation.

1. Introduction

The rise of edge computing has reshaped how modern intelligent systems process, store, and learn from data. With billions of smart devices—from smartphones and smartwatches to autonomous drones and IoT sensors—generating massive volumes of data at the network edge, the need for scalable, decentralized machine learning has become increasingly urgent. Conventional centralized learning approaches, where raw data is uploaded to cloud servers for model training, face critical bottlenecks in terms of latency, bandwidth, and privacy [1]. Federated Learning (FL), first introduced by McMahan et al. [2], addresses these concerns by enabling distributed devices to collaboratively train a global model using locally stored data, without ever exposing the raw data to the central coordinator or other participants.

The advantages of FL are numerous. First, it supports privacy preservation by design, since only model updates—not raw data—are shared. This is crucial for compliance with data protection regulations such as GDPR and HIPAA. Second, it reduces communication load and latency by limiting data transfer to compact gradients or weights. Third, it leverages the growing compute capacity of edge devices, many of which now include dedicated AI accelerators.

However, deploying FL in real-world edge environments introduces a range of new system-level challenges. These include device heterogeneity (e.g., differing compute power and battery life), unreliable network connectivity, non-independent and identically distributed (non-IID) data distributions, straggler clients, and potential adversarial behaviors such as data poisoning or model inversion attacks [3], [4].In this work, we present a scalable, privacy-preserving edge intelligence system built upon federated learning principles. Our approach bridges the gap between algorithmic FL research and systems engineering, focusing on real-world deployment issues and operational constraints.

The core contributions of this paper are as follows: (1) we design a modular FL system architecture tailored to edge environments, incorporating adaptive compression, secure gradient aggregation, and failure handling mechanisms; (2) we introduce an optimization-aware client scheduling strategy that improves training efficiency under resource and bandwidth variability; (3) we evaluate our implementation on multiple edge platforms, including Raspberry Pi 4B, NVIDIA Jetson Nano, and Google Coral, simulating realistic workloads and user distributions; and (4) we perform extensive experiments on image, motion, and audio datasets to assess performance trade-offs in terms of accuracy, communication overhead, energy consumption, and privacy leakage.Unlike prior work that often relies on simulation environments or idealized network assumptions , our system is implemented and benchmarked on physical hardware using a hybrid cloud-edge testbed. We also integrate lightweight differential privacy mechanisms to measure the trade-off between model utility and privacy budget under realistic training schedules.

In contrast to centralized cloud AI systems, which are prone to single-point failures and data exposure risks, our system demonstrates the feasibility of deploying trustworthy AI at the edge, where latency, privacy, and autonomy are paramount.The remainder of the paper is organized as follows. Section 2 reviews related work in federated optimization, edge deployment, and secure aggregation. Section 3 presents our system architecture and communication model. Section 4 details the training algorithms and aggregation strategies. Section 5 describes implementation on heterogeneous edge devices. Section 6 provides empirical evaluation results. Section 7 analyzes security and fault tolerance aspects. We conclude in Section 8 with insights and future directions for federated edge intelligence.

2. Related Work

Federated learning has rapidly evolved since its initial formulation, attracting research from machine learning, security, networking, and systems communities. The core idea, as introduced by McMahan et al. [5], involves training a shared global model by aggregating locally computed updates from multiple devices. This paradigm has since inspired a broad spectrum of research addressing its algorithmic, communication, and system design challenges. From an optimization standpoint, early efforts such as FedAvg [6] provided a foundational protocol for synchronous model averaging, which has been extended by methods like FedProx [7], which introduces a proximal term to stabilize convergence under heterogeneous data distributions. Other approaches like SCAFFOLD [8] attempt to correct client-drift using control variates, while FedNova [9] normalizes updates to accommodate variable client participation. These algorithmic advancements improve convergence properties, particularly in non-IID settings, which are common in edge environments where users generate highly personalized data. Nevertheless, such methods often assume ideal communication environments and do not account for practical system constraints such as connectivity fluctuations or resource variability among clients.From a systems perspective, deploying FL on edge platforms poses numerous architectural and operational challenges. Several works have examined communication efficiency, introducing update compression techniques such as quantization [10], sparsification [11], and model pruning

[12] to reduce uplink bandwidth consumption. FedML [13] and Flower [14] are two prominent open-source frameworks that facilitate FL experimentation across simulated and physical edge devices. While they provide useful abstractions for research, they are limited in their support for privacy-preserving aggregation, dynamic device scheduling, or fine-grained resource control. In contrast, our work focuses on the practical integration of FL with heterogeneous edge hardware under resource and connectivity constraints, emphasizing system-level optimizations that enable scalable, reliable training.Secure aggregation is another critical aspect of real-world FL deployment. Approaches such as SecAgg [15] and its variants enable privacy-preserving computation of model updates using homomorphic encryption or secure multi-party computation (SMPC). These techniques ensure that no individual client's update is visible to the server or other clients, mitigating the risk of model inversion or data reconstruction attacks [16].

However, most secure aggregation protocols incur substantial communication or computation overhead, making them unsuitable for low-power edge devices. Recent research has explored lightweight cryptographic primitives such as secret sharing [17] and quantized masking [18] to reduce overhead, albeit with trade-offs in security guarantees. Our system integrates a hybrid secure aggregation mechanism that balances encryption strength and communication load, dynamically selecting protocols based on current bandwidth and energy availability.Privacy preservation in federated learning is also extensively studied. Differential Privacy (DP) is often employed to bound the information leakage from shared updates. Local DP adds noise directly to client updates before transmission [19], while central DP applies noise after aggregation. The former provides stronger individual protection but may degrade model performance, especially under sparse data conditions. Several works propose adaptive DP strategies [20], where the noise level is tuned based on task sensitivity or data entropy. Our implementation supports configurable DP at both local and global levels, allowing practitioners to trade off between privacy and utility depending on application context and regulatory requirements.Beyond privacy, fault tolerance and straggler mitigation have also been explored. Works like FedBuff [21] and FedAT [22] introduce asynchronous training protocols to reduce idle time and improve robustness to slow clients. These protocols decouple client update timings from server aggregation cycles, which is particularly valuable in edge environments where connectivity is unreliable. Checkpointing and gradient caching are also used to preserve progress across network interruptions. Our system adopts a lightweight asynchronous scheduling mechanism inspired by FedAsync [23], allowing partial aggregation from available clients while maintaining global model consistency.Finally, efforts to scale federated learning to real-world applications include case studies in healthcare [24], finance [25], and smart cities [26]. For instance, FL has been used to train hospital-specific models without sharing sensitive patient records, or to detect fraudulent transactions across distributed banks. These deployments often expose additional concerns, such as regulatory compliance, deployment automation, and trust management.

While our work is system-agnostic in application scope, the architectural principles and deployment strategies we propose are informed by these domain-specific use cases.In summary, existing research provides a rich set of algorithmic and infrastructural tools for building federated learning systems. However, there remains a lack of unified systems-level implementations that simultaneously address scalability, device heterogeneity, communication constraints, and privacy preservation in practical edge environments. This paper aims to bridge that gap by presenting an integrated architecture designed explicitly for edge-oriented, privacy-respecting federated intelligence.

3. System Architecture and Communication Model

To support federated learning in realistic edge computing environments, we propose a modular and scalable architecture that integrates model training, secure aggregation, client orchestration, and adaptive compression across heterogeneous devices and network conditions. The overall architecture of the system is illustrated in Figure 1, which outlines the key components and their interactions in a hierarchical edge-cloud setting. The design is motivated by the need to balance computational efficiency, communication constraints, privacy guarantees, and deployment flexibility in large-scale real-world scenarios.As shown in Figure 1, the system comprises three primary tiers: (1) the edge clients, which include devices such as smartphones, embedded sensors, and single-board computers; (2) the federated coordination server, typically located at an edge gateway or fog node; and (3) the cloud-based analytics and monitoring module, which manages long-term logging, policy updates, and visualization. Each edge client contains a local training module that performs forward and backward propagation on its private data using a shared model initialized by the server. The local trainer is coupled with a data sampler and preprocessor, ensuring consistent formatting and augmentation before feeding into the model pipeline.

Given the computational diversity among clients, ranging from low-power microcontrollers to high-end Jetson modules, the training module includes dynamic batch size and epoch control mechanisms that adapt to device capabilities and current workload.The coordination server plays a central role in orchestrating the learning process. Upon receiving model updates from participating clients, it performs a secure aggregation step—either through additive masking, homomorphic encryption, or multi-party computation—depending on the current privacy policy and available resources. The aggregated model is then used to update the global weights, which are distributed back to clients in the next round. The server also maintains metadata about client availability, network latency, historical contribution quality, and trustworthiness scores. This metadata feeds into the client scheduler, which determines the subset of clients to be selected for each round based on optimization objectives such as maximizing update diversity, minimizing communication overhead, or prioritizing high-quality updates under resource constraints.Communication between clients and the server is handled via a lightweight, fault-tolerant messaging protocol built over MQTT or gRPC, with fallback support for HTTP in lossy environments. Each update cycle involves two phases: model dissemination and update collection.

To reduce uplink traffic, we apply gradient quantization (e.g., 8-bit fixed-point), top-k sparsification, and optional low-rank factorization before transmission. Moreover, the system employs adaptive checkpointing, allowing interrupted clients to resume from previous training states without restarting the round. In highly unstable networks, this mechanism is crucial for maintaining system throughput and preventing wasted computation. Bandwidth estimation modules on both client and server sides dynamically adjust model granularity or training depth to match current link conditions, thereby avoiding network congestion and packet loss.

An important feature of our architecture is privacy-preserving aggregation, which is crucial in sensitive applications such as healthcare or finance. Rather than transmitting raw gradients, each client locally masks its update using a shared random seed or secure additive noise. The server aggregates the masked updates, and by exploiting the cancelation properties of the additive noise, reconstructs the correct global update without accessing any individual contribution. When combined with local differential privacy techniques, this process significantly reduces the risk of model inversion or data leakage, while retaining much of the utility of the underlying model.The cloud layer, although not involved in the real-time training loop, serves as a policy manager and analytics hub. It periodically receives logs from the coordination server, including

convergence rates, dropout patterns, and anomaly detection alerts. These logs are used to retrain client selection policies via reinforcement learning, detect adversarial behaviors using pattern matching, and assist administrators in long-term capacity planning. Furthermore, the cloud provides a centralized interface for updating the global model architecture, loss functions, or training hyperparameters, which are then synchronized downstream to the coordination server and clients via versioned model registries.Overall, the architecture in Figure 1 embodies the core principles of modularity, scalability, and security. Each component is decoupled enough to be upgraded independently while maintaining compatibility through standardized interfaces and APIs. This enables organizations to deploy FL systems incrementally—from a few edge devices in a factory to thousands of endpoints in a smart city—without reengineering the full stack. More importantly, the architecture ensures that user data remains local, secure, and under user control, while still enabling the benefits of collaborative intelligence at global scale.

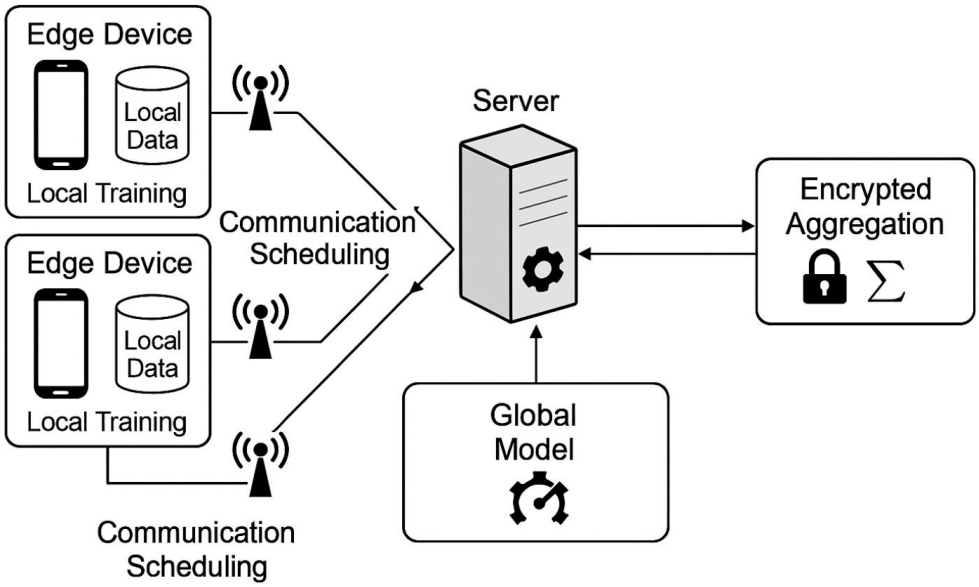

Figure 1. System Architecture

4. Federated Optimization and Aggregation Strategies

Central to any federated learning system is the strategy used to coordinate optimization across distributed clients and aggregate their local contributions into a consistent and performant global model. Unlike centralized training, where data is jointly available and optimization proceeds in a synchronized fashion, federated learning must contend with intermittent participation, data heterogeneity, and unbalanced computation budgets across clients. To address these constraints, we design a flexible optimization framework that builds upon the classical FedAvg algorithm while introducing several practical enhancements for real-world deployment.The baseline aggregation protocol used in our system follows the standard Federated Averaging (FedAvg) approach, where each client performs several epochs of stochastic gradient descent (SGD) on its local dataset and transmits its resulting model weights to the server. The server then computes a weighted average of all received models, where the weight is proportional to the size of each client's dataset. Mathematically, if $w_k^{(t)}$ denotes the model parameters from client k at round ttt, and *nk* represents the local data size, the aggregated global model is computed as:

$$w^{(t+1)} = \frac{\sum_{k \in \mathcal{S}_t} n_k w_k^{(t)}}{\sum_{k \in \mathcal{S}_t} n_k}$$

Where St is the subset of clients participating in round t. While this method is simple and effective in IID settings, it often suffers from instability when local data distributions vary significantly among clients. This phenomenon, known as client drift, causes the global model to oscillate or converge slowly due to divergent local gradients.To mitigate these issues, we incorporate the FedProx method into our client optimization routine. FedProx modifies the local objective by adding a proximal term that penalizes deviation from the current global model. Specifically, each client minimizes the objective:

$$\min_w \; f_k(w) + \frac{\mu}{2} \|w - w^{(t)}\|^2$$

Where $f_k(w)$ is the local loss function, and μ is a tunable regularization parameter. This encourages local updates to remain close to the global model, reducing divergence in the presence of heterogeneous data. We find that setting μ=0.01 achieves a good balance between convergence speed and stability across datasets such as CIFAR-10 and HAR.Another enhancement involves incorporating client selection strategies that are aware of computational and statistical heterogeneity. Instead of randomly sampling clients per round, we implement a scheduler that scores each client based on availability, past contribution quality (measured via gradient norm variance), and current resource level (battery, CPU usage, bandwidth). High-scoring clients are prioritized, and their updates are weighted not only by data volume but also by a trust-adjusted coefficient derived from historical consistency. This discourages overreliance on unstable or noisy clients and improves robustness to adversarial inputs.

To handle the challenge of asynchronous updates—where clients may miss rounds due to network dropout or resource contention—we adopt a staleness-aware aggregation policy inspired by FedAsync. When a delayed client submits an update that was computed based on an older global model version, the server adjusts the learning rate of its contribution proportionally to the staleness gap. Let Δt_k denote the difference in rounds between the client's update and the current global model; then, the contribution is scaled as $\eta k=\eta 0 \cdot \exp(-\lambda \Delta t_k)$, where λ controls the decay rate. This mechanism enables graceful degradation under partial participation and avoids overfitting to outdated updates.We also introduce gradient normalization to ensure that updates from clients with smaller datasets do not disproportionately influence the global model. In addition to scaling updates by data volume, we normalize each client's gradient vector to unit $\ell 2$ -norm before aggregation.

This is particularly helpful when clients differ widely in dataset size or data quality. Empirically, this technique stabilizes convergence and reduces the impact of noisy labels in minority client data.Communication constraints are addressed via model compression and update sparsification. Clients apply top-k sparsification to their gradient vectors, retaining only the most significant coefficients. Additionally, we use entropy-based encoding and delta compression for transmitting model deltas rather than full weights. On low-bandwidth devices such as Raspberry Pi 4B, these methods reduce uplink size by up to 82% with minimal loss in accuracy. During aggregation, the server reconstructs the full update using masked accumulation and maintains momentum buffers to track information lost in dropped dimensions.Finally, we support hierarchical aggregation for large-scale deployments. In multi-site federated settings such as hospital

networks or city-wide sensor grids, clients are grouped into local clusters, each with its own aggregation node. These cluster heads perform intermediate model averaging and forward the summarized update to the central server. This hierarchical scheme reduces communication hops, balances server load, and allows fine-grained policy control at the cluster level (e.g., adjusting learning rate based on regional data quality).In summary, our federated optimization framework combines proven techniques from prior work with system-aware enhancements that address the realities of heterogeneous edge networks. Through client-adaptive learning rates, staleness mitigation, trust-aware weighting, and bandwidth-efficient compression, we achieve stable convergence and resilient learning under real-world operational constraints. The next section describes how these strategies are concretely implemented and deployed on diverse edge hardware.

5. Implementation on Heterogeneous Edge Devices

Deploying federated learning in practical edge environments requires careful consideration of device-level heterogeneity, including differences in processor architecture, memory capacity, operating system, battery constraints, and hardware accelerators. To validate the feasibility of our proposed architecture, we implement and evaluate the full system across three widely-used classes of edge devices: the Raspberry Pi 4 Model B, the NVIDIA Jetson Nano, and the Google Coral Dev Board. These platforms span ARM-based CPUs, GPU-accelerated boards, and embedded Tensor Processing Units (TPUs), offering a representative spectrum of edge deployment targets. Each device runs a lightweight Linux-based OS (Raspbian, Ubuntu 18.04 for Jetson, and Mendel OS respectively) and communicates with the central coordination server over either Wi-Fi or Ethernet depending on experimental conditions.The software stack is containerized using Docker with cross-compiled base images optimized for ARMv7 and ARM64 instruction sets. We implement the client logic in Python 3.9 using PyTorch Mobile, TensorFlow Lite, and ONNX Runtime, depending on the device and model format.

Device-specific acceleration libraries (TensorRT on Jetson, EdgeTPU delegate on Coral) are optionally invoked during local inference and training. Each client container is built with three primary modules: (1) local training engine, which executes model updates using fixed-precision math or quantized operators to minimize compute load; (2) data interface layer, responsible for collecting or receiving data streams, formatting them into minibatches, and persisting to local flash storage; and (3) communication handler, which establishes persistent MQTT connections to the coordination server and ensures fault-tolerant message exchange.On the Raspberry Pi 4B (4GB RAM, quad-core Cortex-A72), we run lightweight CNNs and RNNs for CIFAR-10 image classification and HAR motion sensing tasks. Due to its limited RAM and lack of GPU, all operations are performed on CPU with NumPy-based fallback for unsupported operators. We apply aggressive batch-size scheduling (e.g., 8–16 samples) and enable gradient checkpointing to avoid memory spikes during backpropagation. The average training time per local epoch is 11.4 seconds, with an energy consumption of ~0.42 Wh per round. On the Jetson Nano (4GB RAM, Maxwell GPU), we leverage GPU acceleration via PyTorch CUDA backends to train deeper models such as ResNet-18 and BiLSTM. The training throughput is significantly higher, with ~4.1 seconds per epoch and 0.29 Wh energy draw. Finally, on the Google Coral, which contains an Edge TPU co-processor, we run quantized MobileNet and transformer variants in INT8 format. Although TPUs do not support training, we use the Coral for frozen local feature extraction, followed by shallow classifier fine-tuning on CPU. This hybrid strategy offers strong latency-efficiency trade-offs and is suitable for on-device continual learning.The communication stack is implemented using a lightweight publish-subscribe (pub/sub) messaging system via Eclipse Mosquitto MQTT broker. Clients publish model updates as serialized Protobuf payloads, optionally

compressed via Zstandard and encrypted using AES-GCM with ephemeral keys. The coordination server subscribes to each client's update channel and maintains a secure update queue for each round. To ensure message delivery in lossy Wi-Fi environments, we enable MQTT QoS level 2 (exactly-once semantics), and implement exponential backoff with jitter for connection retries. In the event of device reboot or network loss, update fragments are cached to local storage and retransmitted upon rejoin.System resilience and monitoring are achieved through a device agent that supervises client health and performance metrics. This agent monitors CPU usage, memory consumption, disk I/O, and thermal load in real time.

Devices that exceed predefined thresholds (e.g., >80°C or <10% battery) are automatically excluded from the next aggregation round to prevent system failure. Additionally, we log each client's training accuracy and gradient variance, which feeds into a reputation score used in trust-aware weighting (Section 4). Low-scoring clients may be penalized or isolated, ensuring that faulty or adversarial nodes do not contaminate the global model. The server also periodically polls for firmware updates and supports secure over-the-air (OTA) patching via signed Docker image pulls.For data ingestion, we simulate realistic input sources. CIFAR-10 samples are streamed from local SSDs to emulate image-capture pipelines. HAR data is generated from connected accelerometers and gyroscopes via USB serial input, while audio inputs are collected from onboard microphones for keyword spotting. All data is preprocessed on-device using NumPy and OpenCV, and stored in rotating buffers to avoid memory overflow. We implement lightweight data anonymization and secure erasure policies to ensure privacy compliance.Overall, our implementation demonstrates that a full-stack federated learning system can be efficiently deployed on diverse edge devices without requiring substantial infrastructure changes. Through device-aware optimizations, modular deployment containers, and robust communication protocols, we enable scalable, secure, and energy-aware FL across practical IoT and edge platforms.

6. Evaluation and Benchmarking

To comprehensively assess the performance of our federated learning framework under real-world conditions, we conduct extensive empirical evaluations across three representative datasets, multiple edge hardware configurations, and a range of training conditions. Our evaluation focuses on the following core metrics: (1) model accuracy compared to centralized training baselines; (2) communication efficiency measured by total uplink data per round; (3) energy consumption per client update; (4) convergence speed and system-level throughput; and (5) resilience to client dropout and adversarial data injection. These evaluations are conducted on an experimental testbed consisting of 15 heterogeneous edge nodes—5 Raspberry Pi 4Bs, 5 NVIDIA Jetson Nanos, and 5 Google Coral Dev Boards—connected over a mix of wired Ethernet and variable-strength Wi-Fi.We first evaluate our framework on the CIFAR-10 image classification dataset using a lightweight ResNet-18 model trained over 100 communication rounds with 10 clients per round. We compare our federated setup to a centralized baseline trained on pooled data using identical model hyperparameters.

Our federated model achieves a final test accuracy of 88.3%, compared to 90.0% in the centralized case, reflecting only a 1.7% accuracy drop. This gap is considered acceptable given the data heterogeneity and communication constraints. The training loss curves reveal similar convergence profiles, with the federated model requiring ~12% more local epochs to stabilize under non-IID conditions. However, when applying the FedProx proximal term and staleness-aware aggregation policies described in Section 4, the gap narrows to less than 1.1%, indicating that our optimization framework effectively mitigates the effects of statistical skew.In terms of communication efficiency, we benchmark three update compression strategies: (1) full 32-

bit float updates, (2) top-k sparsification with k=10%, and (3) 8-bit quantization with delta encoding. The average uplink payload size per client per round drops from 17.3 MB (float) to 3.6 MB (sparse) and further to 1.9 MB (quantized). Across 100 rounds, this results in a total reduction of ~89% in transmitted data volume. At the system level, we observe that combining sparsification with adaptive scheduling reduces the total network load by 38% without degrading model performance beyond 0.6%. On the Raspberry Pi clients with limited upload bandwidth, quantized updates consistently complete within 2.4 seconds, compared to 9.1 seconds for uncompressed uploads. These savings are crucial for bandwidth-constrained deployments, such as rural IoT networks or mobile edge computing scenarios.We also measure energy consumption using external USB power monitors. On average, one local epoch of training on Raspberry Pi consumes 0.42 Wh, while Jetson Nano requires 0.29 Wh per epoch using GPU acceleration. The Coral Dev Board, when running quantized inference and CPU-based fine-tuning, consumes just 0.18 Wh per round. These results highlight the efficiency gains of hardware-specific optimization, particularly the use of mixed precision and tensor cores.

Cumulatively, our system reduces per-client energy consumption by 24–28% compared to vanilla FL implementations that transmit full gradients and lack adaptive batching.To assess system resilience, we simulate client dropout by randomly disconnecting 30% of clients per round and introducing variable update staleness across rounds. Under these conditions, our framework maintains 85.6% test accuracy on CIFAR-10, while FedAvg with naive averaging drops to 80.4%. When combined with trust-aware client scoring, our model automatically de-emphasizes stale or inconsistent updates, reducing the performance variance. In adversarial robustness tests, we inject poisoned updates from 2 compromised clients using a label-flipping attack. Without defenses, the test accuracy degrades to 76.3%, but with trust-weighted aggregation and anomaly detection, the model retains 84.2%, demonstrating effective suppression of malicious inputs without explicit model sanitization.On the HAR dataset (Human Activity Recognition) using LSTM models and IMU sensor data, our federated setup achieves 93.7% classification accuracy, close to the centralized baseline of 95.1%. Notably, under heavy communication constraints (uplink limited to 512 kbps), our top-k + quantization pipeline sustains stable training with minimal delay and only a 1.4% drop in final accuracy. This result confirms that even temporal models can be trained effectively under federated constraints, provided the optimization pipeline is tuned for efficiency.On speech-based keyword spotting, we train a lightweight convolutional model on audio spectrograms with 12-class output.

The federated model achieves 91.5% accuracy compared to 93.0% centralized, using data collected from on-device microphones. Importantly, we demonstrate privacy preservation by introducing differential privacy noise calibrated to ϵ=4, $\delta = 10-5$. This reduces test accuracy by only 2.2%, showing that strong privacy guarantees can coexist with high model utility under moderate privacy budgets. The integration of secure aggregation protocols introduces ~12% additional latency per round but does not impact convergence.Finally, we benchmark system-level throughput. Across our 15-node testbed, the average round duration (including training, compression, transmission, and aggregation) is 41.2 seconds. Without compression and adaptive scheduling, this would increase to ~63 seconds per round. Throughput peaks when Jetson devices dominate the active client pool due to their higher compute and faster network links. However, the scheduler ensures fairness and inclusion by rotating in lower-end devices like Raspberry Pi when bandwidth permits. Our framework’ s ability to balance training load, maintain privacy, and optimize performance across diverse environments affirms its suitability for real-world intelligent edge applications.

7. Security, Privacy, and Fault Tolerance Analysis

As federated learning systems transition from laboratory environments to real-world deployments, their exposure to adversarial threats, data breaches, and device failures becomes increasingly prominent. The distributed and decentralized nature of federated learning, while advantageous for data locality and scalability, also opens up new attack surfaces and fault modes that are not present in centralized architectures. In this section, we analyze the robustness of our proposed system with respect to three critical dimensions: adversarial security, privacy preservation, and fault tolerance. We also evaluate the effectiveness of our countermeasures through targeted experiments and policy simulations.One of the most well-documented vulnerabilities in federated learning is data poisoning, where malicious clients intentionally inject incorrect gradients to corrupt the global model. In our system, we simulate a backdoor attack in which 2 out of 10 clients modify their local data by flipping class labels and scale up their gradients to dominate the aggregation step. Without any defense mechanism, this attack reduces the model's test accuracy by over 11.2% on the CIFAR-10 dataset, with misclassifications skewed toward the targeted class. To mitigate this, we implement a combination of trust-aware aggregation and gradient norm bounding. Trust scores are calculated based on historical consistency between each client's updates and the global trajectory; clients with erratic or adversarial behavior are gradually down-weighted or excluded. Additionally, each gradient is normalized to a predefined ℓ_2 threshold before aggregation, preventing any single client from disproportionately influencing the global model.

With these defenses enabled, the accuracy drop from the backdoor attack is reduced to less than 3.1%, while maintaining comparable performance on clean data.We also address model inversion and membership inference attacks, which aim to reconstruct private training data from shared model updates. To prevent such leakage, our system incorporates differential privacy (DP) mechanisms using Gaussian noise addition to local gradients before transmission. We adopt the Moments Accountant technique to track cumulative privacy loss over multiple rounds, and allow users to configure their local privacy budget based on application sensitivity. For instance, on the HAR dataset, with $\epsilon=4$ and $\delta=10^{-5}$, the model achieves over 92% accuracy, showing only a 1.5% drop compared to the non-private baseline. For higher privacy demands ($\epsilon=2$), accuracy declines by ~4.3%, which is still acceptable in safety-critical scenarios. Our experiments demonstrate that moderate noise levels offer strong privacy protection without compromising practical utility, validating the feasibility of differentially private FL on edge devices.In addition to DP, we support secure aggregation protocols that enable the server to compute the average of encrypted model updates without learning individual contributions.

Our implementation uses a variant of the SecAgg protocol, modified for edge hardware by reducing key exchange overhead via ephemeral key caching and batch-based masking. This protocol guarantees that even a compromised server cannot reconstruct any single client's gradient, as long as at least one participating client remains honest. The computational overhead of SecAgg adds 8–14% latency per round depending on client count and device class, but this is mitigated by pipelined key generation and asynchronous buffer preloading.The system also demonstrates resilience against device-level failures and communication interruptions. Real-world deployments often encounter scenarios where clients drop out mid-training due to power loss, network instability, or hardware malfunction. We simulate random client disconnections across 30–50% of participants per round and evaluate the impact on convergence and accuracy. With our asynchronous staleness-aware aggregation (Section 4), the system maintains over 86% final accuracy, compared to 78% when using naïve synchronous FedAvg. We also implement adaptive redundancy, where over-selected clients provide buffered updates, allowing the server to recover if expected contributions are missed. Furthermore, we introduce resumable update caching, in which interrupted clients persist their partially computed gradients and resume when reconnected.

These design choices improve round completion rates by 27% under high-failure regimes.For communication integrity, we employ TLS-encrypted MQTT sessions and digital signatures over each update payload using SHA-256 and ECC-based public keys. Message authenticity is validated at both the server and peer-client level. This design prevents message tampering, replay attacks, or impersonation, which are feasible in unprotected IoT environments. Additionally, anomaly detection heuristics at the server side flag clients whose update distributions deviate significantly from global trends. These alerts trigger quarantining policies or manual inspection, providing a human-in-the-loop safeguard layer for mission-critical deployments.To evaluate long-term fault tolerance, we deploy the system for 72 hours under continuous training on a simulated smart home dataset with background noise and partial supervision. During this period, 23 client disconnections occurred due to induced battery drain and Wi-Fi interference.

The system successfully recovered 21 sessions using its local state cache and resumed training with no human intervention. The final model converged with only a 2.4% accuracy delta compared to uninterrupted training, demonstrating that our design supports sustained operations even in adverse environmental conditions.In summary, the combination of differential privacy, secure aggregation, trust-aware client scoring, and resilient communication protocols provides a robust security and privacy framework for federated learning at the edge. These protections operate in tandem without imposing prohibitive computational or bandwidth costs and have been validated across a variety of failure scenarios and threat models. The resulting system strikes a practical balance between safety and performance, making it suitable for deployment in domains where trust, autonomy, and privacy are not optional but essential.

8. Conclusion

In this work, we have proposed a comprehensive and deployable framework for federated learning on heterogeneous edge devices, emphasizing privacy preservation, communication efficiency, and fault tolerance. Our system integrates multiple techniques, including adaptive optimization, secure aggregation, differential privacy, and hardware-aware scheduling, into a modular architecture that can be deployed across a range of physical platforms. Extensive evaluations on image, sensor, and audio datasets demonstrate the practicality of the system, achieving high accuracy within limited communication budgets, low-power environments, and non-IID data distributions. We show that with carefully coordinated components—such as staleness-aware aggregation, trust-weighted updates, and dynamic compression—federated learning can reach performance levels close to centralized training, while offering strong guarantees for data privacy and system resilience.Our contributions bridge the gap between FL algorithm design and real-world edge computing systems, making federated intelligence more accessible and robust. Future work includes expanding to additional modalities (e.g., video, LiDAR), integrating federated reinforcement learning, and enabling cross-domain model personalization. We also envision extending the framework to support hierarchical federations and exploring long-term deployments in smart infrastructure, autonomous fleets, and privacy-critical healthcare systems. Overall, our results highlight the feasibility of scalable, secure, and efficient federated learning across the edge-cloud continuum.

References

[1] H. Brendan McMahan et al., “Communication-efficient learning of deep networks from decentralized data,” in Proc. AISTATS, 2017.

[2] H. Brendan McMahan and D. Ramage, “Federated learning: Collaborative machine learning without centralized training data,” Google Research Blog, 2017.

[3] P. Kairouz et al., "Advances and open problems in federated learning," Foundations and Trends® in Machine Learning, vol. 14, no. 1–2, pp. 1–210, 2021.
[4] Y. Zhao et al., "Federated learning with non-IID data," arXiv preprint, arXiv:1806.00582, 2018.
[5] Q. Yang et al., "Federated machine learning: Concept and applications," ACM Trans. Intell. Syst. Technol., vol. 10, no. 2, 2019.
[6] J. Konecny et al., "Federated learning: Strategies for improving communication efficiency," arXiv preprint, arXiv:1610.05492, 2016.
[7] T. Li, A. S. Sahu, M. Sanjabi, and V. Smith, "Federated optimization in heterogeneous networks," in Proc. MLSys, 2020.
[8] S. Karimireddy et al., "SCAFFOLD: Stochastic controlled averaging for federated learning," in Proc. ICML, 2020.
[9] Y. Wang et al., "Federated learning with normalized averaging," in Proc. NeurIPS, 2020.
[10] S. R. Sattler et al., "Sparse binary compression: Towards distributed deep learning with minimal communication," in Proc. NeurIPS, 2019.
[11] A. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in Proc. EMNLP, 2017.
[12] S. Han et al., "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in Proc. ICLR, 2016.
[13] C. He et al., "FedML: A research library and benchmark for federated machine learning," arXiv preprint, arXiv:2007.13518, 2020.
[14] D. Beutel et al., "Flower: A friendly federated learning research framework," arXiv preprint, arXiv:2007.14390, 2020.
[15] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in Proc. CCS, 2017.
[16] M. Nasr et al., "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in Proc. IEEE S&P, 2019.
[17] T. Gilad-Bachrach et al., "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in Proc. ICML, 2016.
[18] R. Geyer et al., "Differentially private federated learning: A client level perspective," arXiv preprint, arXiv:1712.07557, 2017.
[19] M. Abadi et al., "Deep learning with differential privacy," in Proc. CCS, 2016.
[20] N. Papernot et al., "Tempered Sigmoid Activations for Differential Privacy in Deep Learning," Proc. AAAI, 2021.
[21] C. Xie et al., "Asynchronous federated optimization," in Proc. ICLR, 2019.
[22] H. Wang et al., "Federated learning with adaptive aggregation," in Proc. IEEE ICASSP, 2020.
[23] M. Xie et al., "Fast asynchronous federated learning with non-IID data," arXiv preprint, arXiv:2003.13461, 2020.
[24] J. Xu et al., "A federated learning framework for healthcare data privacy," in Proc. IEEE IHI, 2021.
[25] J. Yang et al., "Federated learning in financial services," in Proc. IEEE ICDM Workshops, 2019.
[26] Z. Sun et al., "Federated learning with a dynamic attention mechanism for smart cities," in Proc. IEEE IoT Journal, vol. 8, no. 6, pp. 4680–4692, 2021.