REVIEW ARTICLE

# Federated Learning for Artificial Intelligence in Embedded Systems

**Kanthavel Radhakrishnan**[1], **Dhaya Ramakrishnan**[1,*] **and R. Adline Freeda**[2]

[1] School of Electrical & Communications Engineering, PNG University of Technology, Lae, Papua New Guinea
[2] KCG College of Technology, Chennai, India

## Abstract

**Federated Learning (FL) has emerged as a principled paradigm for privacy-preserving decentralized machine learning, enabling model training across distributed embedded devices without centralizing sensitive data. This review examines FL as applied to resource-constrained embedded and edge AI systems, encompassing its architectural foundations, principal optimization algorithms, application domains, and security mechanisms. We analyze the interplay between FL's theoretical properties and the practical constraints imposed by heterogeneous embedded hardware, non-IID data distributions, bandwidth-limited IoT networks, and adversarial threat models. Application domains examined in depth include smart healthcare, autonomous vehicles, industrial IoT, precision agriculture, smart home automation, and urban infrastructure, with attention to how FL enables privacy-preserving AI deployment in each context. We further review the principal techniques that make embedded FL viable—including FedAvg, FedProx, model compression, gradient sparsification, differential privacy, secure multi-party computation, and blockchain-secured aggregation—and identify open research gaps in heterogeneous model aggregation and adaptive privacy protection. Future directions encompassing TinyML integration, federated reinforcement learning, and next-generation 5G/6G network infrastructure are discussed. The review aims to provide a technically grounded reference for researchers and practitioners seeking to deploy FL in real-world embedded environments where privacy, energy efficiency, and communication constraints are simultaneously binding.**

**Keywords**: federated learning, embedded systems, artificial intelligence, edge computing, privacy-preserving internet of things, machine learning at the edge, data privacy, decentralized machine learning.

## 1 Introduction

The proliferation of AI in embedded systems has fundamentally altered how intelligent devices operate, enabling real-time decision-making and a degree of device autonomy that was previously achievable only through continuous cloud connectivity. Embedded platforms—spanning Internet of Things (IoT) sensors, wearable health monitors, industrial controllers, and

**Submitted:** 21 March 2025
**Accepted:** 21 May 2025
**Published:** 27 June 2025

**Vol.** 2, **No.** 2, 2025.
10.62762/TETAI.2025.440076

***Corresponding author:**
✉ Dhaya Ramakrishnan
dhayavel2005@gmail.com

**Citation**
Radhakrishnan, K., Ramakrishnan, D., & Freeda, R. A. (2025). Federated Learning for Artificial Intelligence in Embedded Systems. *ICCK Transactions on Emerging Topics in Artificial Intelligence*, 2(2), 91–115.

smart home devices—have historically offloaded the computational burden of model training and inference to centralized cloud infrastructure. While effective, this architecture carries substantial costs: raw data must traverse potentially insecure networks, exposing sensitive information to interception and misuse; bandwidth availability in distributed field environments is often unreliable; and the round-trip latency between edge device and cloud server undermines the responsiveness demanded by time-critical applications in healthcare, automotive safety, and industrial control.

Federated Learning (FL) was developed precisely to address these tensions. Rather than aggregating raw data on a central server, FL trains models locally on participating devices and transmits only model parameter updates for server-side aggregation [1]. FL is particularly well suited to IoT environments, where the heterogeneity and resource constraints of distributed edge devices present fundamental challenges to conventional centralized learning pipelines [2]. By keeping sensitive data on the device that generated it, FL simultaneously reduces transmission overhead, narrows the attack surface for data breaches, and removes the practical barrier imposed by bandwidth scarcity in remote or infrastructure-limited deployments. The consequence for embedded AI is significant: devices can now engage in continuous on-device learning without exposing confidential data, enabling persistent improvement of models for applications as diverse as cardiac anomaly detection on wearables, object recognition in autonomous vehicles, occupancy inference in smart buildings, and predictive maintenance in industrial IoT [3]. That said, the heterogeneity inherent in large-scale embedded deployments—where devices differ substantially in processor architecture, memory capacity, and connectivity—poses a persistent challenge to achieving uniform model performance across the participant population [4].

The integration of FL into embedded systems introduces a cluster of interrelated technical challenges that distinguish it from FL deployments on more capable hardware. Device heterogeneity is perhaps the most pervasive: embedded platforms span several orders of magnitude in computational capability and storage, making it difficult to apply a single training configuration uniformly [5]. This heterogeneity is compounded at the data level, because devices in different physical contexts collect data with highly divergent statistical properties—the non-independent and identically distributed (non-IID) problem that makes naive gradient averaging a poor aggregation strategy and slows global model convergence [6]. Communication overhead presents a further structural constraint: the frequent exchange of model updates between edge nodes and aggregation servers can saturate low-bandwidth IoT network links, especially as deployment scale grows. Techniques such as model compression, weight pruning, and gradient sparsification have been proposed to reduce update volume, while differential privacy mechanisms have been introduced to provide formal privacy guarantees during aggregation. Security is an additional and equally pressing concern: because model updates flow across shared network infrastructure, FL systems are susceptible to adversarial manipulation including model poisoning and backdoor injection, demanding robust aggregation protocols and anomaly detection mechanisms for real-world deployments [7]. Addressing communication efficiency and security together, rather than in isolation, is essential to the practical scalability of FL across large embedded networks [8].

This paper presents a systematic review of FL as applied to embedded and edge AI systems, examining the interplay between its theoretical properties and the practical constraints of resource-limited hardware. We survey the foundational architecture of FL, analyze the principal algorithms—including FedAvg and FedProx—through the lens of embedded system constraints, and evaluate the privacy, security, and communication mechanisms that make FL viable at the edge. Real-world application domains are examined in depth, including smart healthcare, autonomous vehicles, industrial IoT, smart agriculture, and urban infrastructure, with attention to how FL enables privacy-preserving AI deployment in each context [9]. We further identify open research gaps in heterogeneous model aggregation and privacy protection under non-IID conditions, and we outline future directions encompassing TinyML integration, federated reinforcement learning, blockchain-secured aggregation, and the role of emerging 5G and 6G network infrastructure in enabling low-latency federated training at scale. The aim throughout is to provide a technically grounded reference for researchers and practitioners seeking to deploy FL in real-world embedded environments.

**Literature Search Strategy**

This review was conducted through systematic searches of IEEE Xplore, ACM Digital Library, Google Scholar, and arXiv, using the primary search terms *federated learning*, *embedded systems*, *edge AI*, *IoT*, *privacy-preserving machine learning*, and their combinations. The search covered publications from 2016 (the year FL was formally introduced) through early 2025. Inclusion criteria required that articles address FL in the context of resource-constrained hardware, privacy, security, or application domains relevant to embedded deployment. Review articles, conference papers, and journal publications were all considered; purely theoretical works without connection to embedded or edge deployment were excluded.

## 2 Foundations and Architecture of FL for Embedded Systems

The convergence of embedded AI and edge computing is reshaping how intelligent services are deployed across resource-constrained platforms, from IoT sensors and industrial controllers to wearables and mobile devices. FL sits at the center of this convergence, offering a principled mechanism for jointly training models across distributed embedded devices without ever centralizing the raw data those devices generate [3]. Understanding why FL is architecturally well matched to the embedded edge, what obstacles stand in the way of its deployment, and how recent hardware and algorithmic advances are lowering those obstacles is essential groundwork before examining specific applications and techniques.

### 2.1 Need for FL in Embedded and Edge Environments

The case for FL in embedded deployments is ultimately driven by the unsustainability of the centralized alternative. As the population of connected edge devices grows exponentially—from smart meters and agricultural sensors to factory floor actuators and personal health monitors—the volume of raw data they collectively generate far exceeds what can be economically or practically transferred to central servers [10]. Bandwidth on IoT network links is frequently constrained, latency requirements for safety-critical applications are strict, and the sensitivity of data generated in domains such as healthcare, personal communications, and security surveillance makes centralized aggregation a significant regulatory and ethical liability.

FL resolves these tensions by inverting the conventional data flow. Instead of routing raw measurements to a server, each device trains a local model on its own data and transmits only the resulting parameter update. The server aggregates these updates—using algorithms such as Federated Averaging—to refine a shared global model, which is then redistributed to participants. The raw data itself never leaves the originating device. This architecture yields benefits across several dimensions simultaneously. From a privacy standpoint, sensitive information such as medical sensor readings or security camera footage is processed entirely at the edge, without exposure to remote infrastructure [10]. From a communication standpoint, transmitting compact gradient vectors rather than raw data streams dramatically reduces the bandwidth burden, which is critical in networks where link capacity is scarce [11]. The approach also accommodates personalization naturally, since each device can fine-tune the global model on its own local data distribution without sharing that data with others. And because additional devices contribute to model improvement without placing further load on central servers, the architecture scales gracefully to populations of millions of participants—a property that centralized ML fundamentally cannot match.

### 2.2 Challenges of Implementing FL on Embedded and Edge Devices

Despite these structural advantages, deploying FL on embedded hardware introduces a set of challenges that do not arise in either conventional centralized ML or FL on more capable cloud or desktop platforms [12]. These challenges are not independent: they interact in ways that make naïve application of standard FL algorithms ineffective on real embedded deployments.

The most immediate constraint is hardware resource limitation. Embedded devices operate under tight bounds on RAM, non-volatile storage, processor clock speed, and battery capacity. These bounds restrict both the complexity of the local model that can be trained and the frequency with which updates can be computed and transmitted. A microcontroller with 256 KB of RAM cannot run the same local training loop as a smartphone, yet both may need to participate in the same federated round. Network instability compounds this problem: wireless links from edge devices are inherently variable, and synchronous FL protocols—in which the server waits for updates from all selected

participants before aggregating—are vulnerable to straggler effects when slow or intermittently connected devices delay the round. Hardware heterogeneity adds a further dimension of complexity, because the embedded device ecosystem spans processor architectures ranging from ARM Cortex-M to RISC-V to custom domain-specific ASICs, each with different numerical precision support, memory layout, and instruction throughput, making it difficult to deploy a single FL runtime with consistent performance across the full participant pool. Finally, the distributed and open nature of the FL update exchange creates a security surface that is qualitatively different from centralized training: because updates flow across shared network infrastructure from devices that may themselves be physically compromised, FL systems are exposed to model poisoning attacks in which a malicious participant injects manipulated gradients to degrade or backdoor the global model. Each of these challenges must be addressed—in combination, not in isolation—for FL to be viable at the embedded edge.

### 2.3 Recent Technological Trends

The embedded and edge computing landscape has evolved substantially in the years since FL was first proposed, and several technological developments are directly lowering the barriers identified above [8]. The TinyML movement has produced neural network architectures and training pipelines specifically designed for devices with kilobyte-scale memory and milliwatt-scale power budgets, making it feasible to run local training steps on hardware that would previously have been considered inference-only. Platforms such as Google's Coral Edge TPU and NVIDIA's Jetson Nano provide hardware acceleration for deep learning computations at power envelopes compatible with battery-operated deployment, substantially reducing the time and energy cost of local FL updates. At the algorithmic level, techniques such as post-training quantization, structured pruning, and low-rank weight decomposition allow models to be compressed to a fraction of their original size before deployment, reducing both the memory footprint required for local training and the communication cost of transmitting updates. Federated Reinforcement Learning (FRL) extends the FL paradigm to settings where agents must learn policies through interaction with an environment rather than from a fixed dataset, opening FL to cooperative embedded systems such as drone swarms, autonomous vehicle fleets, and multi-robot manufacturing cells. Taken together, these developments are steadily closing the gap between what FL requires and what embedded hardware can provide, making the combination of FL and edge AI an increasingly practical and commercially relevant proposition [13].

## 3 Key Applications and Case Studies of FL in Embedded Systems

The practical significance of FL for embedded systems lies not in its theoretical elegance but in how it resolves a set of constraints that have historically prevented the deployment of adaptive AI in resource-limited field environments. Embedded devices—IoT sensors, wearables, autonomous platforms, and industrial controllers—generate data that is simultaneously sensitive, voluminous, and geographically dispersed. Conventional cloud-based ML requires that data be transmitted to a central server for training, which is untenable when bandwidth is scarce, latency must be minimized, and data sensitivity precludes transmission altogether. FL resolves this impasse by inverting the data flow: models travel to data rather than data traveling to models.

### 3.1 Conceptual Foundations and Evolution

FL was formally introduced by McMahan et al. [1] as a method for training ML models across large populations of decentralized devices without requiring access to raw data. The central insight is that model weight updates—rather than the underlying training examples—can be aggregated at a server to produce an improved global model, leaving sensitive data resident on the originating device. This property is directly relevant to embedded deployments in healthcare [14], autonomous vehicles, and home automation, where the data generated is both irreplaceable for model improvement and legally or ethically off-limits for centralized collection. Efficient resource allocation during the update exchange is a further prerequisite for viability in wireless embedded environments [15].

Even so, FL's adoption in embedded systems has not been straightforward. The most persistent obstacle is device heterogeneity: smartphones, wearables, and industrial IoT sensors differ by orders of magnitude in processor speed, memory, and network connectivity, making uniform FL deployment technically challenging [5]. Compounding this, the statistical distribution of locally generated data varies dramatically across devices—a property known as non-IID data distribution—which causes naive gradient averaging to produce biased global models and slows convergence [5]. At scale, involving

thousands or millions of devices, even modest per-device communication costs accumulate into network-level bottlenecks [16]. To address the volume and cost of update transmission, researchers have proposed model pruning, gradient compression, and sparsification techniques [8]. Security presents an additional dimension of risk: model updates exchanged over shared networks are susceptible to poisoning attacks and backdoor injection, both of which can subtly corrupt the global model without detection [11].

The privacy protections inherent to FL—local data retention—are necessary but not sufficient. Model updates themselves can leak information about the underlying training data through gradient inversion attacks, motivating the development of complementary mechanisms. Differential privacy (DP) addresses this by adding calibrated noise to updates before transmission, formally bounding the information that any single participant's data can contribute to the aggregate [22]. Secure Multi-Party Computation (SMPC) allows multiple parties to jointly compute the aggregated update without any party observing individual contributions [23]. Trustworthy FL service frameworks layer auditing and integrity verification on top of these cryptographic protections [24]. Byzantine-resilient aggregation algorithms, such as Krum [36] and Trimmed Mean [38], further harden the system against the subset of participants who may deliberately submit malicious updates [7]. However, adaptive poisoning strategies can systematically circumvent these defenses by optimizing malicious updates to mimic the statistical profile of benign contributions [37]. Together, these mechanisms make FL deployments in embedded systems substantially more robust than the basic FedAvg formulation alone.

Recent years have seen several advances that expand the practical envelope of embedded FL. The TinyML movement has produced model architectures—MobileNet, SqueezeNet, and their derivatives—that fit within the kilobyte-scale memory budgets of microcontrollers, enabling local training steps on devices that were previously inference-only [17]. Personalized FL addresses the non-IID problem by allowing devices to maintain locally adapted variants of the global model, capturing individual data distributions while still contributing to collective learning. The distinction between cross-silo FL—involving a small number of trusted institutional participants such as hospitals or manufacturing plants—and cross-device FL at population scale on consumer devices has clarified the design space, with different aggregation strategies appropriate for each regime. Federated Transfer Learning (FTL) extends the paradigm further by allowing devices with small or idiosyncratic datasets to contribute meaningfully through transfer from a pre-trained global model. These advances collectively extend FL's reach into application domains where the original FedAvg formulation would have been insufficient. Table 1 summarizes how FL addresses the principal limitations of centralized ML for embedded systems, and Table 2 maps the specific strategies FL employs to the resource constraints of the embedded environment. Beyond the application domains examined in detail below, FL has also been successfully demonstrated in cooperative localization tasks across distributed sensor networks, illustrating its versatility in embedded environments where position estimation must be performed without centralizing raw ranging measurements [20].

### 3.2 Historical Development and Milestones

FL emerged from a practical need identified at Google: mobile keyboard prediction models could be substantially improved by learning from user typing behavior, but the sensitivity of that data made centralized collection unacceptable. The 2016 papers by McMahan et al. [1] and Konečný et al. [40] formalized the problem and introduced FedAvg and federated optimization as communication-efficient solutions for on-device intelligence, establishing the conceptual and algorithmic foundation that the entire field has since built upon. In 2017, the release of open-source frameworks including TensorFlow Federated lowered the barrier to experimentation and accelerated the growth of the research community. By 2018, FL had expanded beyond its mobile origins into healthcare and IoT, with demonstrations of privacy-preserving personalized health monitoring on wearable devices showing that the approach was viable in embedded contexts. The 2019 period brought significant advances in formal privacy protection, with hybrid approaches combining differential privacy and secure aggregation providing stronger theoretical guarantees against information leakage during the update exchange [26]. By 2020, attention had shifted to robustness under heterogeneous conditions: the FedProx algorithm addressed the non-IID data and device heterogeneity problems that had limited FedAvg's effectiveness in real-world embedded deployments, improving convergence stability across diverse IoT participant populations [5].

**Table 1.** Federated learning in embedded systems.

| Aspect | Challenges in Traditional ML | How FL Addresses It |
|---|---|---|
| Privacy | Centralized data collection exposes sensitive user data. | FL keeps data localized, only sharing model updates. |
| Bandwidth | Limited network bandwidth makes large data transfers costly. | FL reduces data transmission by only sharing model parameters. |
| Latency | Real-time applications require low-latency processing. | Local model training ensures faster decision-making. |
| Energy Efficiency | Embedded systems have limited power for processing and transmission. | FL optimizes computation and minimizes communication. |
| Scalability | Centralized ML struggles with a growing number of devices. | FL enables decentralized learning across multiple devices. |

**Table 2.** How FL fits into the resource-constrained embedded environment.

| Strategy | Description |
|---|---|
| Decentralized Learning | FL allows devices to train models locally and only send model updates (gradients or parameters), reducing data transfer needs. |
| Model Compression & Optimization | Techniques like pruning, quantization, and knowledge distillation help minimize model size and computational complexity, making FL feasible for embedded systems. |
| Adaptive Aggregation | FL uses methods like Federated Averaging (FedAvg) to efficiently collect model updates while reducing computational and communication costs. |
| Hierarchical FL Architecture | Instead of direct communication with a central server, hierarchical structures (e.g., edge-based aggregation) allow intermediate edge devices to collect updates before sending them to the cloud. |
| Energy-Efficient Training | Embedded systems can optimize power consumption using techniques such as selective update transfer and event-triggered learning. |
| Security & Privacy Mechanisms | FL integrates methods like differential privacy and secure aggregation to protect sensitive data from exposure and adversarial attacks. |

### 3.3 Basic Architecture for FL

The architecture of an FL system, illustrated in Figure 1, comprises three functional layers that interact in a coordinated cycle. At the base are the client devices—smartphones, IoT sensors, wearables, industrial controllers, or any embedded platform that holds local data and possesses sufficient compute to run a training step. Each device maintains a local copy of the model and computes parameter updates (gradients or weight deltas) based solely on its own data; no raw data is ever transferred off device. The central aggregation server occupies the second layer. It receives the parameter updates from selected clients, combines them using an aggregation rule—most commonly a weighted average proportional to each client's dataset size, as in FedAvg—and produces a refined global model that is broadcast back to participants. The server also manages the logistics of the training process: selecting which clients participate in each round, scheduling communication, and controlling synchronization to handle the straggler problem. The third layer is the communication network that connects clients to the server. Because only model parameters travel over this network, its bandwidth requirements are substantially lower than those of a data-centric pipeline; nonetheless, the periodic nature of the exchange means that communication is a recurring cost over potentially thousands of rounds [6]. Privacy-preserving data collection protocols in IoT contexts ensure that raw measurements never enter the network at any point in the process [27].

### 3.4 Basic Workflow in FL

The federated training loop, depicted in Figure 2, follows a repeating cycle of four phases. In the

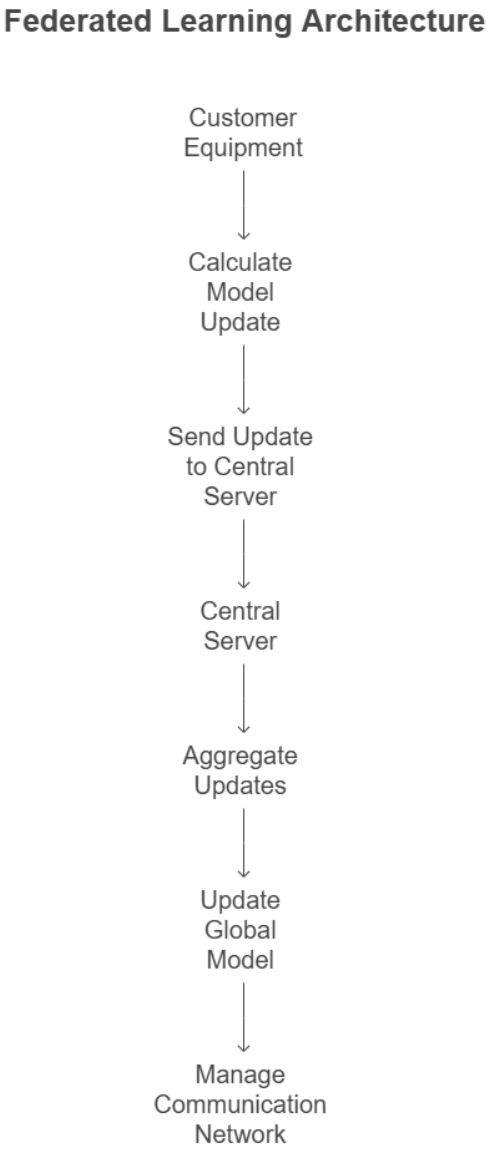


**Figure 1.** Basic FL architecture.

initialization phase, the server establishes a global model with random or pre-trained parameters and distributes it to the selected participant devices. Each device then enters the local training phase, running one or more epochs of gradient descent on its own data to produce a locally updated model. Once local training is complete, devices transmit their parameter updates to the server—not the raw data, and not the full updated model unless compression makes that economical—which aggregates the received updates to produce a new global model. This refined model is then redistributed to participants, completing one round. The cycle continues until the global model reaches a target performance threshold or a predefined round limit is reached. The simplicity of this loop belies the engineering complexity involved in making it work reliably across thousands of heterogeneous embedded devices under variable network conditions, which motivates the substantial body of work on aggregation algorithms, client selection strategies, and communication compression reviewed in subsequent sections.

### 3.5 Smart Agriculture

Precision agriculture represents a compelling embedded FL use case because it combines large volumes of environmentally sensitive sensor data with strong incentives to keep that data private and severe constraints on network connectivity [10]. A modern agricultural operation may deploy hundreds of IoT sensors monitoring soil moisture, ambient temperature, crop canopy reflectance, and pest pressure across geographically dispersed field plots, along with drone platforms performing aerial imaging and autonomous machinery executing site-specific interventions. Transmitting the raw outputs of this sensor network to a central server for model training is impractical on multiple grounds: the data volumes are large, rural network infrastructure is often limited to low-bandwidth cellular or LoRa links, and proprietary farm management data represents a competitive asset that operators are reluctant to share. FL addresses each of these constraints simultaneously. Sensors and drones train local models on their own measurements and transmit only compact gradient updates, reducing network load to a fraction of what raw data transfer would require. Each farm or field zone can fine-tune the shared global model to its specific soil type, microclimate, or crop variety, producing personalized decision support—optimal irrigation scheduling, targeted pest control recommendations—that a globally uniform model could not provide. As new sensors are deployed across additional sites, they join the federation and contribute to model improvement without overloading any central processing resource, making the system inherently scalable [16].

### 3.6 Edge-Based Medical Diagnostics

Healthcare represents perhaps the domain where the privacy argument for FL is strongest, because the data involved—patient health records, continuous physiological sensor streams, medical imaging studies—is subject to the most stringent regulatory protections, including HIPAA in the United States and GDPR in Europe, and the consequences of unauthorized disclosure are severe for both patients and institutions. FL enables hospitals, clinics, and individual patients' devices to participate in collaborative model improvement without any raw health data leaving the originating institution or device.

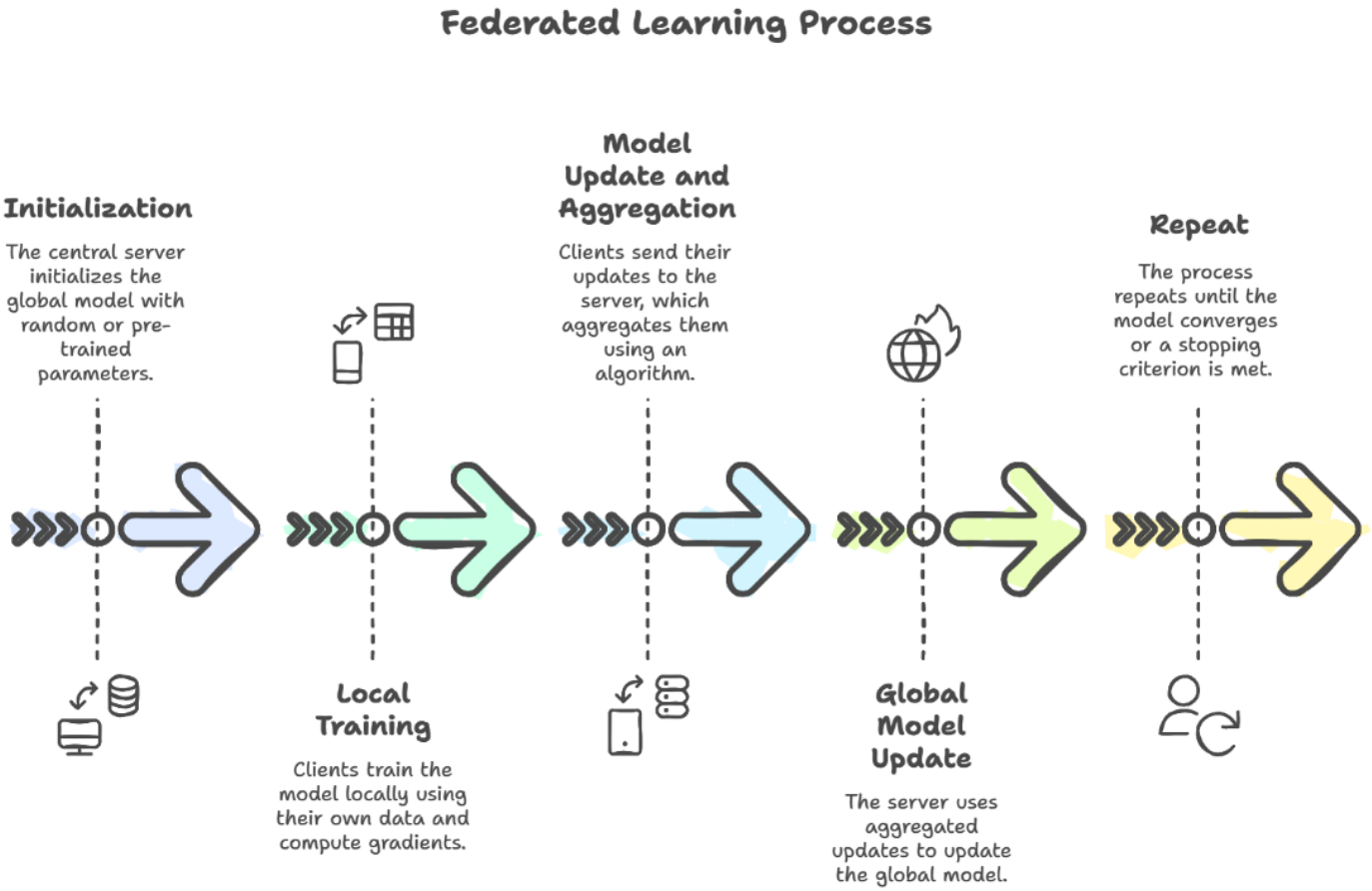


**Figure 2.** Workflow in FL.

A wearable ECG monitor can refine its arrhythmia detection model using the patient's own cardiac rhythm history; a hospital's radiology department can contribute to a shared chest X-ray classification model without exposing individual patient studies to external parties; a distributed network of continuous glucose monitors can collectively improve hypoglycemia prediction accuracy without pooling any patient's glycemic trajectory data. The result is a system in which model quality improves with population scale while individual privacy is preserved by design [14]. The real-time inference capability enabled by local model training is equally important: in clinical settings where a few seconds of latency can determine the appropriate response to a deteriorating patient, the ability to run inference locally—without a round trip to a cloud server—is a functional requirement rather than a merely desirable optimization.

### 3.7 Energy-Aware Edge Robotics

Autonomous mobile robots, logistics drones, and collaborative manufacturing systems present a distinct set of FL requirements shaped by strict energy budgets and the need for rapid environmental adaptation. A field inspection drone operating on a single battery charge cannot afford the energy cost of transmitting raw sensor data to a central server after each flight; yet it must continuously improve its obstacle detection and path planning models as it encounters new terrain and lighting conditions. FL allows the drone to update its local model in the field using onboard compute, transmitting only the resulting parameter delta at the end of the mission. In multi-robot settings—warehouse fleets, drone swarms, or cobotic manufacturing cells—FL enables collective learning without centralized data pooling: each robot contributes its local experience to a shared model while retaining the raw sensor logs that generated it. This combination of local adaptation and global knowledge sharing is particularly valuable when individual robots operate in heterogeneous sub-environments—different floor surfaces, lighting conditions, or task structures—that are each represented by only a single agent in the fleet. Security is a critical concern in these settings, as FL edge systems are vulnerable to sophisticated poisoning attacks that may exploit generative adversarial networks to craft malicious model updates [25]; robust aggregation mechanisms and anomaly detection at the server are therefore

essential components of any production edge robotics FL deployment.

### 3.8 Smart Home Automation

In smart home environments, FL enables a qualitatively different mode of personalization than cloud-based approaches: one in which the personalization model is derived entirely from the occupant's own behavioral data without that data ever leaving the home. Smart thermostats learn individual occupancy patterns and temperature preferences; lighting systems adapt to circadian preferences and activity-dependent illumination needs; security cameras refine their anomaly detection thresholds based on the specific visual characteristics of a given household's normal activity. Because all training data remains on the device, users are protected from the risks associated with behavioral profiling at cloud scale. Voice assistant platforms such as Google Home and Amazon Alexa represent a particularly high-stakes application of this principle: speech and natural language data are among the most sensitive categories of personal information, and FL provides a mechanism for improving wake-word detection and natural language understanding using real user utterances without routing those utterances to external servers. Smart meters and connected appliances can similarly apply FL to optimize energy consumption patterns—learning which loads to defer, when to pre-condition spaces, and how to coordinate with grid demand signals—while keeping household energy usage data private.

### 3.9 Autonomous Vehicles

Autonomous vehicle systems benefit from FL in two complementary ways: individual vehicles can improve their perception and decision models using locally generated driving data, and fleets of vehicles can collectively learn from the aggregate of their diverse driving experiences without centralizing any individual vehicle's GPS traces, camera footage, or behavioral logs. The non-IID nature of the data is pronounced in this domain—a vehicle operating primarily in urban stop-and-go traffic encounters a fundamentally different data distribution than one traveling rural highways—making personalized or heterogeneity-aware aggregation strategies particularly important. FL enables continuous improvement of safety-critical functions including object detection and classification, adaptive cruise control calibration, lane-keeping assistance, and hazard prediction, with model updates propagated across the fleet after each aggregation round. Traffic management infrastructure can participate in the same federation: roadside sensors and signal controllers contribute data on vehicle flow patterns and intersection behavior, enabling dynamic signal timing optimization that reduces congestion without requiring individual vehicles to report their location or routing history to a central authority.

### 3.10 Industrial IoT

In industrial manufacturing and logistics contexts, the data generated by embedded sensors—vibration spectra from rotating machinery, thermal signatures from electrical equipment, pressure traces from hydraulic systems—is both highly proprietary and extremely valuable for predictive maintenance and process optimization. FL enables competing manufacturers to collectively train more accurate anomaly detection and failure prediction models than any single operator could develop from their own data alone, without any party revealing the operational details of their specific processes. Embedded sensors on production equipment monitor early failure indicators—changes in vibration frequency, temperature gradients, current draw anomalies—and update local predictive maintenance models that flag maintenance requirements before equipment failure occurs, minimizing unplanned downtime [18]. The same federated architecture supports supply chain optimization across distributed logistics networks, where inventory management and demand forecasting models can be improved using real-time data from geographically dispersed warehouses and distribution centers without centralizing commercially sensitive stock and order data [19].

### 3.11 Intelligent Urban Infrastructure

Smart city deployments extend FL's embedded applications to urban-scale infrastructure, where the participating devices are traffic sensors, surveillance cameras, smart meters, and grid-connected energy systems distributed across an entire metropolitan area. Public safety applications represent a particularly sensitive use case: surveillance cameras and acoustic sensors can collaboratively train threat detection and anomaly recognition models without streaming raw video or audio to a central server, substantially reducing both the bandwidth requirements of the network and the privacy risks inherent in centralized video storage. In energy distribution, smart grid components—meters, inverters, controllable loads—can apply FL to develop localized demand

forecasting and load balancing models that adapt to neighborhood-level consumption patterns without aggregating individual household energy data at a utility's data center. Urban traffic management similarly benefits from federated intelligence: traffic signals, loop detectors, and connected vehicle infrastructure can coordinate through shared model updates to optimize signal timing and routing recommendations at the network level while preserving the location privacy of individual road users.

Table 3 provides a consolidated overview of FL applications across these embedded domains, highlighting the specific AI functions enabled and the privacy and efficiency properties that FL provides in each context.

Across all of these domains, the common thread is that FL enables the deployment of continuously improving AI models in environments where centralized data collection is either technically infeasible, economically prohibitive, or legally and ethically unacceptable. Figure 3 provides a visual overview of these representative application domains. The breadth of these applications reflects FL's fundamental suitability as an architectural choice for embedded AI: it is not a narrow optimization of a single system but a general paradigm shift in how learning is distributed across networks of resource-constrained devices.

## 4 Challenges, Research Gaps, and Future Directions

FL's fundamental value proposition—privacy-preserving decentralized learning across heterogeneous devices—is also the source of its most significant implementation challenges. The properties that make embedded FL attractive, namely that training occurs on diverse, resource-constrained hardware operating over variable networks with statistically heterogeneous local data, are precisely the properties that make it difficult to deploy reliably and efficiently. This section examines the principal challenge dimensions in depth, with attention to the specific ways in which the embedded context amplifies difficulties that are more tractable in conventional FL settings.

### 4.1 Device and Hardware Heterogeneity

The embedded device ecosystem is radically heterogeneous in ways that have no counterpart in data center FL deployments. At the hardware level, participating devices may range from microcontrollers with 256 KB of RAM and no floating-point unit, through mid-range application processors, to edge AI accelerators with dedicated tensor cores—all potentially participating in the same federated round. This span of computational capability means that a local training configuration appropriate for one class of device may be entirely infeasible for another. Memory constraints limit the batch size and model size that can be used during local training; processor throughput determines how many local epochs can be completed within a given time budget; and battery capacity determines how many such rounds a device can sustain before requiring recharge. The practical consequence is that FL frameworks designed for the embedded edge must either adapt the training workload dynamically to each device's capability, or accept that participation will be uneven and that some devices will consistently contribute lower-quality updates than others.

Hardware heterogeneity extends beyond raw resource availability to architectural incompatibility. ARM Cortex-M, RISC-V, and custom domain-specific ASICs each have different instruction sets, numerical precision support, memory layout conventions, and compiler toolchains. A model compiled and quantized for one architecture may not run correctly on another without explicit adaptation. Software heterogeneity compounds this further: devices in a real-world embedded FL deployment may run different real-time operating systems, different versions of the same framework, or different hardware abstraction layers, making it difficult to guarantee that local training produces numerically consistent results across the participant pool. These compatibility challenges are not merely engineering inconveniences; they affect the mathematical properties of the aggregation step, since updates produced by differently configured local trainers may not combine cleanly under standard weighted averaging.

### 4.2 Data Privacy, Security, and the Limits of Local Retention

A common misconception about FL is that local data retention fully solves the privacy problem. In practice, the model updates that FL transmits in place of raw data carry significant information about the underlying training examples, and this information can be extracted by a determined adversary. Gradient inversion attacks—in which an attacker who observes a model update reconstructs a close approximation of the training batch that produced it—have been

**Table 3.** Applications of FL in embedded systems.

| Application | Description |
|---|---|
| Healthcare | Wearable health devices (e.g., smartwatches, fitness trackers) train models to detect irregular heart rates, blood pressure deviations, and other health conditions while preserving patient data privacy. Smart hospitals collaborate on disease surveillance models without sharing sensitive patient records across institutions. |
| Smart Home | Home automation systems learn occupant preferences for lighting, temperature, and security locally without transmitting personal behavioral data to cloud servers. Voice assistants refine speech recognition models on-device, and smart meters optimize household energy consumption using FL-trained load forecasting models. |
| Autonomous Vehicles | Connected vehicles share navigation pattern updates, road hazard detections, and object recognition improvements without exposing raw driving data including GPS traces or camera footage. FL improves safety functions such as adaptive cruise control and lane-keeping, and enables traffic management systems to optimize routing while preserving individual location privacy. |
| Industrial IoT (IIoT) | Predictive maintenance models are trained on embedded sensors to detect early equipment failure signatures from vibration, temperature, and pressure data. Smart factories collaboratively improve production efficiency models without exposing proprietary manufacturing process data to external parties. |
| Agriculture & Environmental Monitoring | FL enables precision agriculture by training crop monitoring models using distributed field sensors without centralizing farm operational data. Environmental monitoring networks improve pollution dispersion and weather forecasting models with minimal bandwidth usage, and FL supports sustainable farming through federated optimization of irrigation scheduling and resource allocation. |
| Smart Cities & Infrastructure | Public surveillance systems improve security threat detection models collaboratively without streaming raw video to cloud servers. FL enables decentralized energy distribution optimization in smart grids by training localized demand models on regional consumption data. Traffic lights and road sensors collaborate through FL to improve urban mobility and reduce congestion while preserving driver privacy. |

demonstrated to recover recognizable images from gradients of vision models and to reconstruct text from gradients of language models. In embedded healthcare and personal monitoring contexts, where local datasets may consist of a single person's physiological time series or behavioral patterns, the risk of re-identification from gradient inspection is a serious concern that local data retention alone does not address.

Security threats originate not only from external observers but from within the federation itself. Compromised or maliciously configured devices can submit manipulated updates designed to degrade the global model's performance on a targeted input class (model poisoning) or to cause specific inputs to be systematically misclassified (backdoor attacks). Because the server in a standard FL setup cannot inspect local training data to verify the integrity of submitted updates, detecting poisoning is significantly harder than in a centralized setting. This problem is aggravated in embedded deployments where devices may be physically accessible to adversaries—a concern that does not arise in data center environments. The cryptographic defenses available to address these threats, including homomorphic encryption and differential privacy noise addition, are themselves computationally expensive in ways that may be prohibitive for the most constrained embedded

hardware, creating a fundamental tension between security and deployability that has not yet been resolved satisfactorily.

### 4.3 Non-IID Data Distributions and Aggregation Quality

The statistical heterogeneity of data across embedded devices is arguably the most technically consequential challenge for FL algorithm design. Standard FedAvg was derived under the assumption that local datasets are independent and identically distributed samples from the global data distribution; in real embedded deployments, this assumption is almost universally violated. A wearable health monitor trained on a single patient's cardiac rhythm, an agricultural sensor trained on one farm's soil moisture dynamics, or an autonomous vehicle trained in a specific geographic region each produces a local model that reflects a narrow and potentially unrepresentative slice of the problem distribution. When these locally biased models are averaged, the result may not converge to the same optimum as a model trained on the pooled global dataset, and in extreme cases the averaging process causes oscillation or divergence rather than improvement.

The non-IID problem interacts with partial participation in ways that further complicate aggregation. In large-scale embedded deployments, only a subset of devices participates in any given round due to connectivity constraints, battery limitations, or scheduling conflicts. If the selected subset systematically overrepresents certain data distributions—for instance, because devices in good network coverage areas are more likely to be selected—the aggregated update will be biased toward those distributions, and the global model will perform poorly on the underrepresented device population. Addressing this requires either more sophisticated client selection strategies that actively seek representational balance, or aggregation algorithms that explicitly weight updates to correct for distributional skew. Neither approach is straightforward to implement robustly across the heterogeneous hardware and connectivity conditions of an embedded deployment.

A related gap concerns the mismatch between a single global model and the diversity of local inference requirements. In many embedded applications, what is needed is not a model that performs adequately on average across all devices, but a model that performs well on each individual device's specific data distribution. Personalized FL addresses this by allowing devices to maintain locally adapted model variants, but the interaction between personalization and global aggregation—how much local adaptation to allow before updates cease to be combinable—remains an open research question, particularly for devices with very small local datasets.

### 4.4 Communication Overhead and Energy Constraints

Communication is the primary operational cost of FL, and in embedded networks its management is critical to system viability. IoT deployments commonly rely on low-power wide-area network (LPWAN) protocols such as LoRa or NB-IoT, which offer range and energy efficiency at the price of very limited throughput—often measured in kilobits per second. Transmitting the full gradient vector of even a modestly sized neural network over such a link requires many seconds and consumes a significant fraction of a battery-powered device's daily energy budget. At scale, the aggregate bandwidth consumed by many devices transmitting updates simultaneously can saturate network infrastructure even when individual update sizes are individually manageable. These constraints motivate a range of communication reduction techniques—gradient sparsification, quantized update transmission, local accumulation of updates across multiple rounds before transmission—but each introduces its own tradeoff between communication cost and convergence quality that must be carefully characterized for specific embedded deployment scenarios.

Energy consumption is not separable from communication overhead in battery-powered embedded devices. Local training steps consume processor cycles and memory bandwidth; update transmission consumes radio frequency energy; and both compete with the primary sensing and actuation functions the device was designed to perform. Devices that spend too large a fraction of their energy budget on FL participation may fail their primary mission, which is unacceptable in safety-critical applications such as industrial monitoring or autonomous navigation. Event-triggered learning—in which local training is performed only when the local data distribution shifts detectably—and selective update transmission—in which only the most informative gradient components are communicated—represent promising directions for reducing the energy cost of participation, but their behavior under the full range

of embedded deployment conditions is not yet well understood.

Asynchronous participation presents a further systems challenge. Real-world embedded devices are not reliably available: they go offline when batteries deplete, when they move out of network coverage, or when their primary task demands exclusive use of computational resources. Synchronous FL protocols, which require all selected participants to complete their local training and return updates before the server aggregates, are vulnerable to these dropouts, either waiting indefinitely for straggler updates or discarding them and accepting a biased aggregate. Asynchronous protocols that allow the server to aggregate updates as they arrive avoid this waiting cost but introduce new challenges around staleness—updates computed on an old version of the global model may actually decrease model quality when applied to a newer version—and around fairness, since fast devices with reliable connectivity will contribute disproportionately to model evolution.

### 4.5 Ultra-Low-Latency and Real-Time Learning Requirements

A class of embedded applications—autonomous vehicles, industrial process control, robotic manipulation—requires not just that inference be low-latency but that the model itself be updated in near real-time as operating conditions change. These requirements push against the grain of standard FL, in which training is conducted in rounds with inter-round gaps measured in minutes to hours. A vehicle that encounters an unusual road condition needs its perception model updated on a timescale of milliseconds, not the duration of a federated training round. Achieving this requires a fundamental rethinking of the FL training loop for real-time embedded contexts: local training must be compressed to the minimum number of gradient steps consistent with useful model improvement, communication must be minimized to the point where it does not introduce perceptible latency, and the server must be capable of aggregating and redistributing updates faster than the environment evolves. Meeting these requirements simultaneously on embedded hardware is an open engineering and algorithmic challenge that no existing FL framework addresses comprehensively. Effective FL deployment for real-world embedded systems will require frameworks that jointly optimize model architecture, aggregation algorithm, communication protocol, and hardware configuration as an integrated system rather than addressing each dimension independently.

## 5 Techniques and Frameworks Enabling FL in Embedded Systems

The challenges identified in the previous section—device heterogeneity, non-IID data distributions, communication overhead, and security vulnerabilities—have motivated a substantial body of algorithmic and systems research aimed at making FL viable on resource-constrained embedded hardware. This section reviews the principal technical contributions across four dimensions: federated optimization algorithms, model compression, communication-efficient gradient transmission, and secure aggregation. These are not independent developments; in practice, effective embedded FL deployment requires their coordinated application.

### 5.1 Federated Optimization Algorithms: FedAvg and FedProx

Federated Averaging (FedAvg), introduced by McMahan et al. [1], remains the foundational algorithm for FL. Its mechanics are straightforward: each participating device $k$, holding a local dataset of $n_k$ examples, trains a local model independently and transmits the resulting weight vector $w_k$ to the aggregation server. The server computes a weighted average proportional to each device's dataset size to produce an updated global model:

$$w \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_k \quad \text{where} \quad n = \sum_{k=1}^{K} n_k \tag{1}$$

The appeal of FedAvg in embedded contexts is its communication efficiency: rather than exchanging raw data or gradients at every optimization step, devices perform multiple local epochs before transmitting a single update, substantially reducing the number of communication rounds required to reach a target accuracy. Google's Gboard keyboard application illustrates this concretely [29]: millions of smartphones with heterogeneous processing capacities and network conditions collaboratively train a next-word prediction model without any user's typing history leaving their device. Each phone fine-tunes the shared language model on local interaction logs and returns an encrypted weight delta to the aggregation server, achieving model improvement at population scale with per-device communication costs that remain within the practical

limits of mobile data connections. In healthcare wearables, FedAvg has been applied to cardiac monitoring across distributed heart rate sensors, each of which trains a local anomaly detection model on its wearer's physiological data; the aggregated global model achieves population-level predictive accuracy that no individual device's local dataset could support, while raw health records remain on-device throughout.

Despite its effectiveness when devices are relatively homogeneous, FedAvg's performance degrades substantially under the conditions most characteristic of real embedded deployments: hardware heterogeneity and non-IID data distributions. When devices differ significantly in computational capability, the local training trajectories diverge at rates proportional to the number of local epochs completed, and when local data distributions are dissimilar, this divergence may not cancel upon aggregation—a phenomenon known as client drift that causes the global model to converge to a suboptimal point or fail to converge at all [28]. FedProx addresses this by augmenting each device's local objective with a proximal regularization term that penalizes deviation from the current global model:

$$\min_{w} \; f_k(w) + \frac{\mu}{2}\|w - w_{\text{global}}\|^2 \tag{2}$$

The coefficient $\mu$ controls the strength of this regularization. A higher value of $\mu$ constrains local updates to remain close to the global model, reducing drift at the cost of slower adaptation to local data; a lower value permits more local specialization. This proximal term has two important practical effects in heterogeneous embedded settings. First, it makes the algorithm tolerant of variable local computation: devices that complete fewer local steps due to resource limitations produce updates that are closer to the global model by virtue of the regularization, rather than being arbitrarily less trained. Second, it stabilizes aggregation under non-IID distributions by preventing any device's local model from drifting so far that its update degrades the global average.

The practical significance of this distinction has been demonstrated across multiple embedded domains. In Industrial IoT deployments, manufacturing devices equipped with embedded vibration and thermal sensors vary considerably in data generation rates and signal-to-noise characteristics; FedProx enabled predictive maintenance models to be trained across these heterogeneous systems without high-noise devices corrupting the global model, resulting in earlier failure detection and reduced unplanned downtime. In autonomous vehicle fleets, where urban and rural driving environments produce fundamentally different data distributions, FedProx allowed each vehicle to adapt locally to its operating environment while the proximal term maintained sufficient alignment with the fleet-wide model to preserve safety-critical generalization properties [28]. The contrast between FedAvg and FedProx is therefore not merely algorithmic: it reflects a substantive design choice about the tradeoff between global model quality and robustness to the heterogeneity that is inherent in real embedded deployments.

### 5.2 Model Compression and Optimization

The feasibility of local training on embedded hardware depends critically on whether the model being trained fits within the device's memory budget and can complete a training step within its energy and time constraints. For the majority of embedded platforms, standard deep learning model sizes—typically tens to hundreds of megabytes for vision or language models—are incompatible with available resources. Model compression techniques reduce this gap by trading a controlled degree of representational capacity for substantially reduced computational and memory requirements.

Weight pruning removes parameters whose contribution to model output is below a threshold, producing a sparse network that requires less memory to store and fewer multiply-accumulate operations to evaluate. Structured pruning, which removes entire filters or attention heads rather than individual weights, is particularly suited to embedded hardware because it preserves the dense matrix structure that most processor architectures are optimized to execute efficiently. Quantization reduces the numerical precision of stored weights and activations from the 32-bit floating-point representation used during training to lower-precision fixed-point formats—commonly 8-bit integers, and in some cases 4-bit or binary representations—which reduces memory footprint by a factor of four to eight and replaces expensive floating-point operations with integer arithmetic that executes faster and at lower energy cost on most embedded processors. Knowledge distillation produces compact models through a training procedure in which a small "student" network is trained to replicate the output

distribution of a larger, more accurate "teacher" network, rather than being trained directly on ground-truth labels; this allows the student to achieve accuracy closer to the teacher's than direct training on the same data would permit, at a fraction of the inference cost. Low-rank decomposition factorizes large weight matrices into products of smaller matrices whose total parameter count is substantially lower, reducing both storage requirements and the computational cost of the forward pass. In embedded FL practice, these techniques are applied in combination: a model is first designed or pre-trained at full precision, then pruned, quantized, and potentially distilled before deployment, with the resulting compressed model used as the starting point for federated local training on embedded devices.

### 5.3 Gradient Sparsification and Communication Efficiency

Even with compressed models, the communication cost of federated training remains a binding constraint in bandwidth-limited embedded networks. A focused treatment of communication-efficient federated learning tailored to IoT deployment constraints, covering gradient compression and update sparsification strategies, is provided in [42]. The fundamental insight motivating sparsification is that gradient vectors are empirically sparse in a meaningful sense: a small fraction of entries account for the majority of the information content, and transmitting only these entries recovers most of the convergence benefit of transmitting the full gradient at a fraction of the communication cost.

Top-$K$ sparsification selects the $K$ gradient entries with the largest absolute values for transmission, discarding the remainder. The selection threshold adapts naturally to the gradient magnitude distribution, ensuring that the transmitted subset captures the most informative directional information regardless of the specific model or training stage. Error feedback mechanisms accumulate the discarded gradient components locally and add them to future updates, preventing the systematic bias that would otherwise result from consistently omitting small-but-non-zero entries. Random dropout selects a uniformly random subset of gradient entries for transmission, offering lower computational overhead than magnitude-based selection at the cost of higher variance in the transmitted signal; this tradeoff is favorable in severely bandwidth-constrained environments where the overhead of sorting gradient magnitudes is itself prohibitive. Both approaches can be combined with quantization of the selected entries, further reducing the per-entry transmission cost. The joint design of sparsification and quantization—determining the optimal tradeoff between the number of entries transmitted and the precision with which each is represented, given a fixed communication budget—remains an active area of research with direct relevance to embedded FL deployment.

### 5.4 Secure Aggregation Mechanisms

A systematic survey of privacy-preserving aggregation methods in federated learning, covering the theoretical guarantees and practical tradeoffs of differential privacy, SMPC, and homomorphic encryption, is provided in [21]. The security requirements of embedded FL necessitate protection at two levels: protection of the information content of individual model updates during transmission and aggregation, and protection of the global model against manipulation by malicious participants. Several complementary mechanisms have been developed to address these requirements, each with a different profile of computational cost and security guarantee that must be matched to the capabilities of the target embedded hardware.

Differential privacy (DP) provides a formal information-theoretic guarantee: by adding carefully calibrated Gaussian or Laplacian noise to model updates before transmission, DP bounds the amount of information that any single participant's data can contribute to the aggregate, preventing adversaries from reconstructing individual training examples from observed updates [33]. The privacy guarantee is parameterized by $\varepsilon$, the privacy budget, which quantifies the maximum allowable information leakage; smaller values of $\varepsilon$ provide stronger privacy at the cost of higher noise magnitude and consequently slower convergence. For embedded systems, the computational overhead of DP noise generation is modest, but the accuracy cost of strong privacy guarantees can be significant when local datasets are small, as is common in personal device deployments.

Secure Multi-Party Computation (SMPC) takes a different approach, allowing the aggregation server to compute the sum of participant updates without observing any individual update [23]. In secret sharing protocols, each device splits its update into multiple shares distributed among other participants; only the sum of shares—which

equals the sum of updates—can be reconstructed by the server, while individual shares reveal nothing about the underlying update. Garbled circuit protocols generalize this to arbitrary functions. The primary limitation of SMPC for embedded systems is computational: the cryptographic operations required are substantially more expensive than the gradient computation itself, which may be prohibitive on the most constrained hardware. Homomorphic encryption provides an alternative by allowing arithmetic operations—specifically, the weighted summation required for FedAvg aggregation—to be performed directly on encrypted ciphertext, so that the server aggregates without ever accessing plaintext gradients. Current homomorphic encryption schemes impose ciphertext expansion factors and computational overheads that remain challenging for embedded deployment, though hardware-accelerated implementations are an active research direction.

Blockchain-based aggregation addresses a different threat: the integrity and auditability of the aggregation process itself. By recording model updates and aggregated results on an immutable distributed ledger, blockchain architectures provide tamper-evident provenance for the training history and eliminate the single point of trust represented by a centralized aggregation server [34]. Smart contracts can enforce contribution quality requirements—rejecting updates that fall outside statistical bounds or rewarding high-quality contributions through token-based incentive mechanisms. The combination of blockchain with SMPC provides both integrity and confidentiality: updates are cryptographically protected during aggregation, and the aggregation process itself is publicly verifiable [23]. Blockchain-based ledger systems further support auditable data provenance and privacy-preserving data sharing across organizational boundaries, providing infrastructure upon which cross-institutional FL deployments can be built [35].

### 5.5 TinyML and Low-Power Embedded FL

The TinyML research program addresses the challenge of deploying ML inference—and increasingly, ML training—on microcontrollers and ultra-low-power sensors that operate with kilobyte-scale memory budgets and milliwatt-scale power envelopes [30]. The architectures developed within this program, including MobileNet, SqueezeNet, and their successors optimized for specific hardware targets such as the Google Edge TPU and Arm Cortex-M series, achieve competitive accuracy on classification and detection benchmarks while fitting within the resource envelope of embedded hardware that was previously considered insufficient for neural network deployment. When combined with FL, TinyML enables a qualitatively new capability: on-device continual learning in which models adapt to changing local conditions without requiring cloud connectivity or server-side retraining. A wearable sensor can refine its activity recognition model as the wearer's behavior patterns evolve; an agricultural IoT node can update its crop health classifier as seasonal conditions change; an industrial vibration sensor can recalibrate its anomaly detection threshold as equipment ages. This combination of local adaptability and global knowledge sharing through federated aggregation represents the most practically significant near-term application of FL in resource-constrained embedded environments.

### 5.6 Federated Reinforcement Learning

Federated Reinforcement Learning (FRL) extends the FL paradigm to sequential decision-making problems, in which agents must learn policies through interaction with an environment rather than from a fixed labelled dataset. The application of reinforcement learning (RL) to optimize FL itself—for instance, using RL-based adaptive client selection to improve convergence efficiency on non-IID data across heterogeneous embedded devices—represents one direction of this research [31]. The more general FRL formulation enables populations of embedded agents—autonomous robots, drone swarms, vehicle fleets, smart grid actuators—to learn shared or complementary policies through federated aggregation of locally computed policy gradients, without any agent sharing its raw observation or reward history. This is particularly valuable in multi-robot and multi-agent embedded systems, where the diversity of local experience across agents is a resource to be exploited rather than a heterogeneity problem to be managed. Research challenges in FRL include the design of reward structures that remain meaningful after federated aggregation, the management of non-stationarity in environments where multiple agents are simultaneously adapting their policies, and the development of privacy-preserving mechanisms appropriate to the continuous action and observation spaces characteristic of physical embedded systems.

### 5.7 Edge Computing Integration and Blockchain Security

The integration of FL with edge computing infrastructure introduces a hierarchical aggregation architecture that partially decouples embedded devices from the central aggregation server [32]. In Hierarchical FL (HFL), geographically co-located devices aggregate their updates at a nearby edge node—a base station, gateway, or on-premises server—before the edge-level aggregates are forwarded to a global server for final combination. This two-tier structure reduces the communication distance and latency for the majority of update traffic, which flows only to the local edge node, while preserving the global learning benefit of cross-region aggregation. Edge nodes can also perform local anomaly detection to filter potentially poisoned updates before they propagate to the global model, adding a security layer that reduces the burden on the central server. Hardware advances in edge AI accelerators—including the Google Edge TPU, Intel Movidius Neural Compute Stick, and successive generations of edge-optimized SoCs—are lowering the cost and power consumption of edge aggregation nodes, making the HFL architecture increasingly practical for large-scale embedded deployments [33].

The convergence of TinyML, FRL, hierarchical edge computing, and blockchain-secured aggregation represents the current frontier of embedded FL research. Each of these developments addresses a specific dimension of the embedded FL challenge—resource constraints, sequential decision problems, communication efficiency, and security integrity, respectively—and their combination opens the possibility of FL systems that are simultaneously more capable, more efficient, and more trustworthy than any single-technique approach can achieve. The practical realization of this convergence in production embedded deployments remains an active engineering and research challenge, but the trajectory of development across all four dimensions is clearly towards greater feasibility at the resource margins of the embedded hardware ecosystem.

## 6 Future Directions and Emerging Solutions

### 6.1 Scalability and Adaptability

As FL deployments scale from tens to thousands or millions of embedded devices, the systems challenges of managing the training process grow qualitatively rather than merely quantitatively [13]. At large participant counts, the logistics of coordinating model update collection, detecting and handling stragglers, and maintaining consistent global model versions across a dynamically changing device population become the dominant engineering constraints. Devices enter and leave the federation continuously—due to battery depletion, network outages, or changes in operational status—and training protocols designed for static, fully participating populations perform poorly under these conditions. Federated hyperparameter optimization presents a further difficulty: learning rate schedules, local epoch counts, and update aggregation frequencies that are appropriate for one subset of devices may be inefficient or destabilizing for another, and no single configuration generalizes across the full range of embedded hardware capabilities.

Hierarchical FL (HFL) addresses the scalability problem by introducing intermediate aggregation nodes—edge servers, base stations, or gateway devices—that collect and consolidate updates from nearby embedded clients before forwarding regional aggregates to the global server. This two-tier structure reduces the communication load on the global server, shortens the effective round-trip distance for most update traffic, and enables the global server to operate on a manageable number of high-quality regional aggregates rather than thousands of raw device updates. Self-adaptive FL algorithms that dynamically adjust training parameters based on real-time estimates of device capability and network conditions represent a complementary direction, allowing the training configuration to track the heterogeneity of the participant population rather than assuming a fixed operating point. For deployments where the communication cost or trust assumptions of centralized aggregation are prohibitive, blockchain-based peer-to-peer architectures provide a path to fully decentralized FL in which no single server holds a privileged aggregation role, at the cost of additional protocol complexity and consensus overhead.

### 6.2 Handling Device Heterogeneity

The heterogeneity of embedded device populations—spanning differences in processor architecture, memory capacity, network connectivity, and local data distribution—is not a transient engineering problem but a permanent structural feature of the environments where embedded FL must operate. Devices with high-speed connections can participate in every round with full updates;

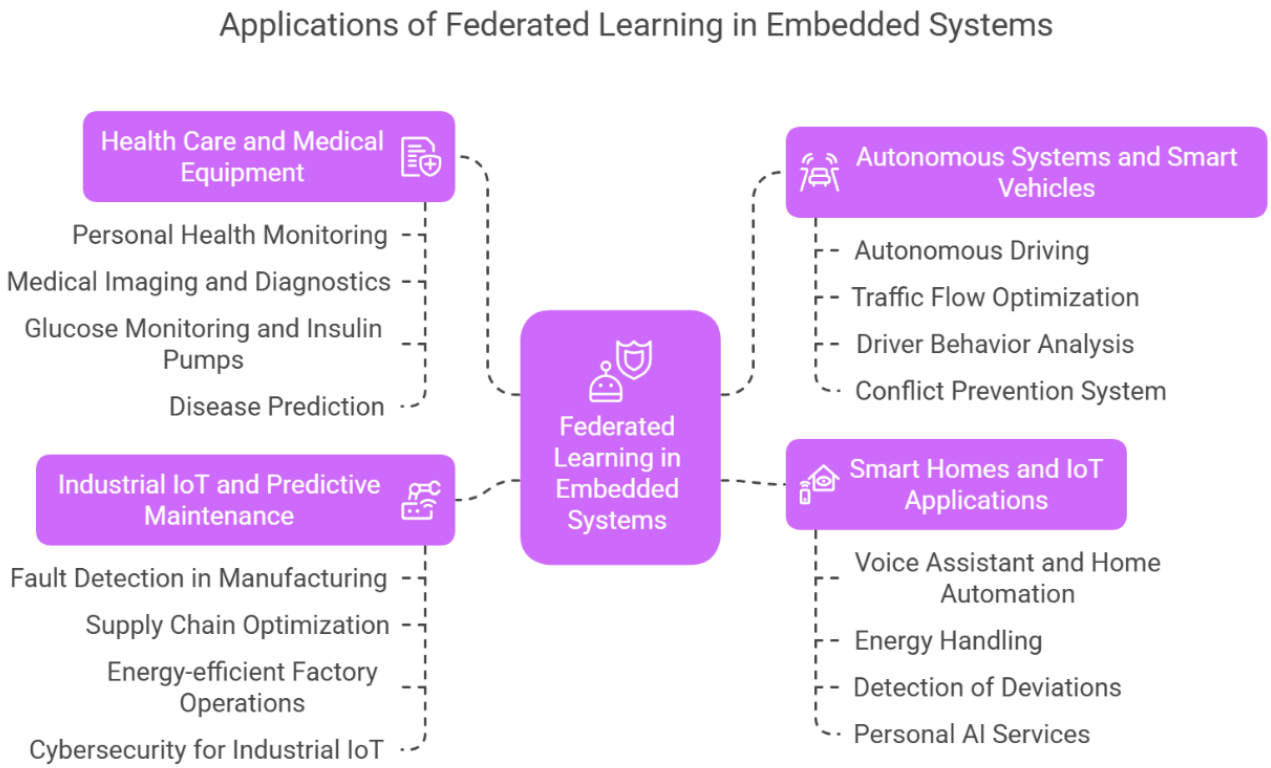


**Figure 3.** Representative application domains of FL in embedded systems, spanning healthcare wearables, autonomous vehicles, industrial IoT, smart home devices, precision agriculture, and urban infrastructure.

devices on LPWAN links may be able to contribute only compressed partial updates at infrequent intervals. Devices with abundant local data contribute well-conditioned updates; devices that have just been deployed or whose operating conditions have changed may contribute updates that are individually accurate but statistically distant from the global distribution. The consequence is that standard synchronous FL protocols, in which all selected devices are expected to complete equivalent local training and transmit equivalent updates within a fixed time window, systematically disadvantage resource-constrained devices and produce aggregates biased toward the most capable participants.

Personalized FL (PFL) reframes the objective from learning a single global model to learning a family of related models, each adapted to the specific data distribution and capability profile of an individual device or device class. By treating the global model as a shared initialization or regularization anchor rather than the final deployed artifact, PFL accommodates distributional heterogeneity without requiring all devices to converge to the same parameter values. Asynchronous FL relaxes the synchronization requirement entirely, allowing devices to submit updates at their own pace and the server to aggregate as updates arrive; this eliminates the straggler problem at the cost of introducing staleness, since updates computed on an older version of the global model may be less beneficial when applied to a newer one. Federated Transfer Learning (FTL) addresses the related problem of devices with small or idiosyncratic local datasets by leveraging pre-trained representations to accelerate local adaptation, allowing devices whose data alone would be insufficient to train a useful model to contribute meaningfully to the federation through transfer.

### 6.3 Communication and Computation Efficiency

Communication cost is the primary operational constraint on the frequency and scale of FL training in embedded networks. Frequent transmission of full model updates over low-bandwidth IoT links consumes a disproportionate share of the energy budget of battery-powered devices and can saturate shared network infrastructure at deployment scale. The latency requirements of real-time embedded applications—autonomous navigation, industrial process control, emergency response systems—add a further dimension: not only must total communication volume be minimized, but individual update cycles must complete within time bounds that are incompatible with the round-trip latency of current FL protocols over constrained links.

Gradient compression and sparsification reduce per-round communication volume by transmitting only the most informative subset of gradient entries, as discussed in the techniques section. Adaptive client selection complements this by choosing, for each round, the subset of available devices whose updates are most likely to improve the global model given their current data distribution and connectivity, rather than selecting randomly or by availability alone. Federated distillation replaces direct parameter exchange with knowledge distillation: rather than sharing weight vectors, devices share the output distributions of their local models on a shared unlabelled dataset, and the server distills these into a global model update; this approach reduces the communication payload to the size of the shared dataset's label distribution rather than the model parameter vector, which can be orders of magnitude smaller for large models. Edge-cloud collaboration structures the computation so that embedded devices perform local inference and lightweight adaptation, while periodic offloading to edge servers handles the heavier retraining steps that exceed on-device resource budgets, balancing latency, energy, and model quality across the two tiers.

### 6.4 Integration with 5G and Next-Generation Networks

The deployment of FL in embedded systems has been significantly enhanced by the emergence of 5G network infrastructure and its accompanying edge computing ecosystem [39]. The ultra-reliable low-latency communication (URLLC) mode of 5G provides round-trip latencies of under one millisecond for short messages, enabling model update exchanges that are fast enough to support real-time federated adaptation in autonomous vehicle and industrial control applications. The massive machine-type communication (mMTC) mode supports simultaneous connectivity for large numbers of low-power IoT devices, enabling FL participation at deployment scales that were previously infeasible due to network access contention. In autonomous vehicle deployments specifically, 5G connectivity allows vehicles to exchange model updates based on local driving observations without transmitting raw sensor data, continuously refining fleet-wide perception models from the aggregate of diverse driving environments [28]. Edge computing nodes co-located with 5G base stations introduce localized aggregation and model validation capability, reducing the communication distance for most update traffic and enabling hierarchical FL architectures that process updates closer to the devices that generated them [18].

Practical 5G-enabled FL deployments span multiple application domains. Smartwatch-class wearables leverage on-device federated optimization to minimize the communication overhead of model updates over 5G connections, enabling continuous health monitoring refinement within tight battery constraints [15]. Intelligent traffic management systems rely on FL's robustness to non-IID vehicular sensor data—which varies substantially across locations and times of day—to maintain accurate traffic flow models across geographically dispersed intersections [41]. Precision agriculture applications use edge-cloud collaboration over 5G to allow distributed field sensors to contribute to soil condition and weather forecasting models while raw sensor streams remain on-site.

Looking further ahead, the transition from 5G to 6G networks—projected to provide ten-fold reductions in latency and order-of-magnitude increases in device density relative to 5G—will further expand the operational envelope of embedded FL. AI-driven network management in 6G architectures is expected to enable self-adaptive FL protocols in which the network itself optimizes update scheduling, compression levels, and aggregation timing in response to real-time traffic and device conditions, removing the manual configuration burden that currently limits FL deployment in dynamic embedded environments. Quantum communication channels, if realized at the access network level, would provide information-theoretically secure update transmission that eliminates the computational overhead of current cryptographic security mechanisms. Network slicing in both 5G and 6G allows dedicated virtual network segments to be allocated to FL traffic, providing quality-of-service guarantees that isolate federated training communication from competing traffic and ensure that round-trip times remain within the bounds required by latency-sensitive embedded applications.

Table 4 consolidates the principal challenges across all four dimensions and the corresponding solution directions discussed in this section.

### 6.5 Research Gaps in Model Aggregation under Heterogeneous Conditions

Despite the progress represented by algorithms such as FedProx and MOON (Model-Contrastive Federated Learning), fundamental gaps remain in the

**Table 4.** Challenges and future solutions of FL in embedded AI systems.

| Category | Challenges | Future Solutions |
|---|---|---|
| Scalability & Adaptability | Managing model updates and aggregation becomes increasingly complex as participant counts grow. Devices frequently join and leave the FL network, requiring adaptive training mechanisms. Adapting learning rates, batch sizes, and update frequencies across heterogeneous devices is difficult without a self-tuning protocol. | Hierarchical FL (HFL) uses intermediate edge nodes to collect and consolidate updates before forwarding to the central server, reducing load and improving scalability. Self-adaptive FL algorithms dynamically adjust training parameters based on real-time device capability and network conditions. Blockchain-based peer-to-peer architectures eliminate centralized aggregation, improving resilience and reducing trust requirements. |
| Handling Device Heterogeneity | Devices differ significantly in memory, processing power, battery life, and network connectivity. Some participants operate on high-speed connections while others rely on low-bandwidth IoT links. Non-IID local data distributions cause biased aggregates and slow or unstable global model convergence. | Personalized FL (PFL) develops client-adapted models rather than enforcing a single global model, accommodating distributional and capability heterogeneity. Asynchronous FL allows devices to submit updates at their own pace, eliminating straggler-induced delays. Federated Transfer Learning (FTL) enables devices with small local datasets to contribute meaningfully through pre-trained knowledge transfer. |
| Reducing Communication & Computation Costs | Frequent full model update transmission causes network congestion and rapid battery depletion on resource-constrained devices. Real-time applications such as autonomous vehicles require update latencies that current FL round times cannot meet. | Gradient compression and top-$K$ sparsification reduce per-round communication volume. Adaptive client selection chooses participants whose updates are most likely to improve the global model given current network and data conditions. Federated distillation replaces weight vector exchange with knowledge transfer over a shared dataset, substantially reducing payload size. Edge-cloud collaboration offloads computation-intensive steps to edge servers while keeping lightweight adaptation on-device. |
| Integration with 5G & Beyond | Ultra-low latency is critical for real-time AI applications such as autonomous systems and industrial control. High device density in large-scale IoT deployments strains network access infrastructure. Network slicing and quality-of-service guarantees for FL traffic are not yet standardized. | 5G URLLC and mMTC modes enable real-time model update synchronization across large heterogeneous device populations. AI-driven 6G network management will enable self-adaptive FL scheduling that responds to real-time traffic conditions. Quantum communication channels will provide information-theoretically secure update transmission without cryptographic computational overhead. |

aggregation of model updates from heterogeneous embedded populations [5]. Standard aggregation methods, including FedAvg and its proximal variants, were designed under the assumption that participating devices share identical model architectures, comparable computational resources, and approximately balanced data distributions. In real embedded deployments, none of these assumptions holds reliably. Devices vary in the model architectures they can accommodate—a microcontroller may support only a lightweight convolutional network while an edge server in the same federation runs a substantially deeper model—and their data distributions may be so dissimilar that weighted averaging of their updates produces a global model that performs adequately on no individual device's distribution.

Partial participation compounds the aggregation problem in ways that go beyond mere sample size reduction. When the subset of devices that participates in a given round is determined by connectivity and battery availability rather than

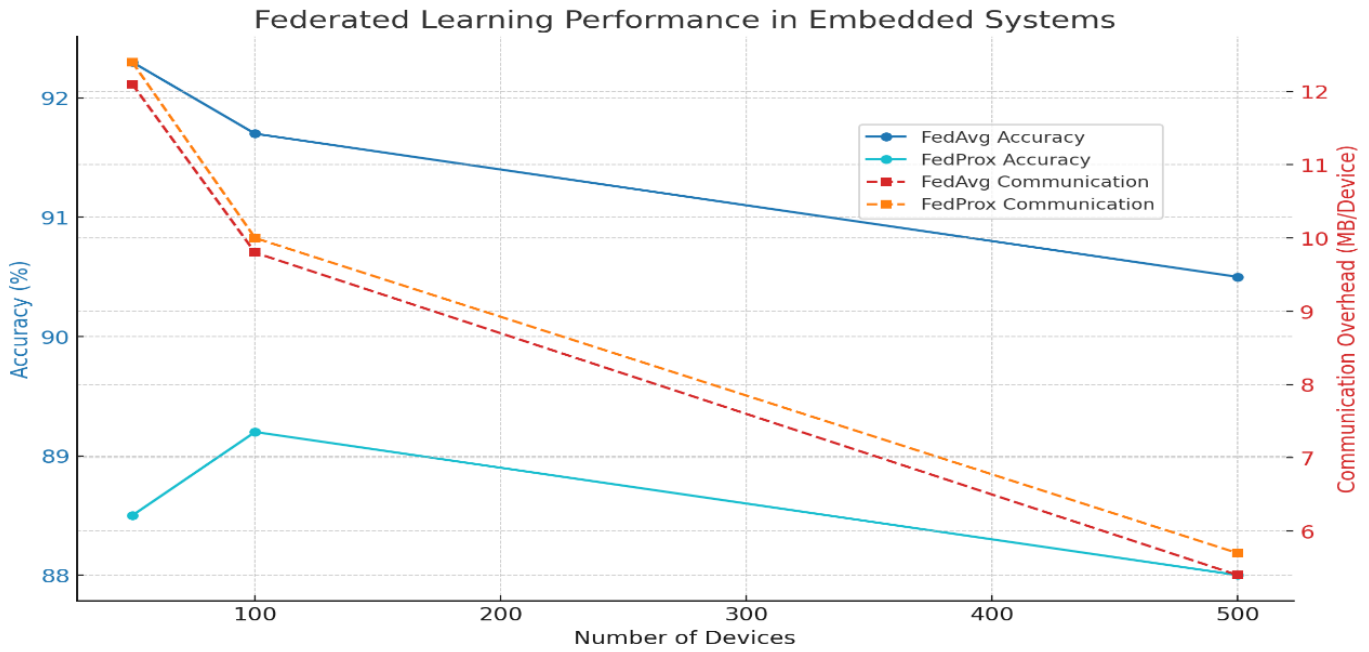


**Figure 4.** Synthetic simulation results showing model accuracy and per-device communication overhead for FedAvg and FedProx across participant populations of 50, 100, and 500 embedded devices under IID and non-IID data distribution conditions.

**Table 5.** Synthetic simulation results: FL scalability across embedded device populations under varying data distributions and aggregation algorithms.

| Devices | Data Distribution | Aggregation Algorithm | Accuracy (%) | Training Time (h) | Comm. Overhead (MB/device) |
|---|---|---|---|---|---|
| 50 | IID | FedAvg | 92.3 | 2.4 | 12.1 |
| 50 | Non-IID | FedAvg | 88.5 | 2.8 | 12.4 |
| 100 | IID | FedProx | 91.7 | 4.1 | 9.8 |
| 100 | Non-IID | FedProx | 89.2 | 4.6 | 10.0 |
| 500 | IID | FedAvg | 90.5 | 10.2 | 5.4 |
| 500 | Non-IID | FedProx | 88.0 | 11.7 | 5.7 |

by design, the resulting aggregate systematically overrepresents devices with reliable power and network access and underrepresents those operating at the resource margins. Over successive rounds, this selection bias accumulates, producing a global model that is progressively less representative of the underrepresented device population. Addressing this requires either active correction of participation bias through weighted aggregation or explicit client selection strategies that compensate for the correlation between device capability and participation probability. Current algorithms do not provide satisfactory solutions to this problem under the full range of conditions encountered in embedded deployments, and it represents an important direction for future research.

### 6.6 Research Gaps in Privacy Protection under Heterogeneous Conditions

The application of differential privacy and secure aggregation protocols to heterogeneous embedded populations raises challenges that extend beyond the resource cost of cryptographic computation. In non-IID settings, model updates are more personalized and less generalizable than in IID settings: a device whose local data is highly distinctive produces an update that carries more information about that data than a device whose local data closely matches the global distribution. Standard DP mechanisms calibrated to a uniform privacy budget across all devices therefore provide weaker privacy guarantees for the most distinctive—and typically most vulnerable—device populations than the nominal

$\varepsilon$ value suggests. Device-specific privacy budgets that account for local data sensitivity and update informativeness would provide more meaningful guarantees, but their implementation requires the aggregation server to have some knowledge of local data characteristics, which may itself constitute a privacy violation.

The threat model facing embedded FL deployments is also evolving in ways that outpace current defense mechanisms. Adaptive adversaries that observe the privacy protection mechanisms in use and craft attacks specifically to exploit their weaknesses—for instance, targeting devices with weaker DP noise levels or exploiting the known structure of SMPC protocols—are more dangerous than the static adversaries assumed in most formal privacy analyses. Membership inference attacks have been shown to achieve higher success rates against updates from non-IID devices, precisely because the personalized structure of those updates makes them more distinguishable [16]. Emerging defense directions include contextual differential privacy frameworks that adapt the noise magnitude to the estimated sensitivity of each device's current update, and hardware-based security primitives such as Trusted Execution Environments (TEEs) that provide isolated computation zones on the device itself, enabling privacy-preserving local training without exposing gradient information even to the device's own operating system. The integration of TEEs into the FL training loop for embedded systems remains largely unexplored beyond isolated proof-of-concept demonstrations and represents a high-impact direction for future work.

### 6.7 Synthetic Simulation Results

To characterize how FL scales across varying device populations and data distribution conditions, a synthetic simulation was conducted in which the number of participating devices, the data distribution type (IID versus non-IID), and the aggregation algorithm (FedAvg versus FedProx) were varied systematically. The results, summarized in Table 5 and Figure 4, reveal several consistent patterns. Model accuracy remains relatively stable as device count increases from 50 to 500 under IID conditions, confirming that FL's aggregation mechanism scales gracefully when data distributions are balanced. Under non-IID conditions, accuracy degrades modestly but consistently for both algorithms, with FedAvg showing greater sensitivity to distributional heterogeneity than FedProx across all device counts. FedProx achieves higher accuracy than FedAvg under non-IID conditions at the cost of a modest increase in training time, reflecting the additional computational overhead of the proximal regularization term. Communication overhead per device decreases monotonically with increasing participant count under both algorithms, a consequence of the hierarchical aggregation structure that distributes the per-round communication load across a larger participant pool—a result with direct practical significance for bandwidth-limited embedded deployments where per-device communication cost is the primary constraint on participation scale.

## 7 Conclusion

This review has examined the integration of federated learning into embedded and edge AI systems, covering its architectural foundations, principal algorithms, application domains, security considerations, and emerging research directions. The analysis demonstrates that FL offers a principled and practically viable solution to the privacy, bandwidth, and latency constraints that make centralized ML unsuitable for many embedded deployments, and that its adoption across healthcare, autonomous systems, industrial IoT, smart homes, and urban infrastructure is already generating measurable real-world value.

At the same time, the review makes clear that reliable embedded FL deployment requires coordinated progress across multiple dimensions simultaneously. Device and data heterogeneity, communication overhead, privacy inference risks, and the computational cost of cryptographic defenses each impose constraints that no single technique resolves in isolation. Advances in TinyML, federated optimization algorithms such as FedProx, gradient sparsification, differential privacy, and blockchain-secured aggregation are individually promising, but their practical impact depends on co-design with the hardware and network constraints of target deployment environments. Personalized FL, hierarchical aggregation architectures, and the low-latency communication infrastructure provided by 5G and forthcoming 6G networks represent the directions most likely to extend FL's reach to the resource margins of the embedded ecosystem. Addressing these challenges systematically will determine whether FL fulfills its potential as the foundational learning paradigm for privacy-preserving intelligence

at the network edge.

## Data Availability Statement

Not applicable.

## Funding

This work was supported without any funding.

## Conflicts of Interest

The authors declare no conflicts of interest.

## Ethical Approval and Consent to Participate

Not applicable.

## References

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017, April). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273-1282). PMLR. [Crossref]

[2] Imteaj, A., Thakker, U., Wang, S., Li, J., & Amini, M. H. (2021). A survey on federated learning for resource-constrained IoT devices. *IEEE Internet of Things Journal, 9*(1), 1-24. [Crossref]

[3] Hazra, A., Adhikari, M., Nandy, S., Doulani, K., & Menon, V. G. (2022). Federated-learning-aided next-generation edge networks for intelligent services. *IEEE Network, 36*(3), 56-64. [Crossref]

[4] Abdelmoniem, A. M., Ho, C. Y., Papageorgiou, P., & Canini, M. (2023). A comprehensive empirical study of heterogeneity in federated learning. *IEEE Internet of Things Journal, 10*(16), 14071-14083. [Crossref]

[5] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems, 2*, 429-450.

[6] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*. [Crossref]

[7] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... & Zhao, S. (2021). Advances and open problems in federated learning. *Foundations and trends® in machine learning, 14*(1-2), 1-210. [Crossref]

[8] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine, 37*(3), 50-60. [Crossref]

[9] Aledhari, M., Razzak, R., Parizi, R. M., & Saeed, F. (2020). Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access, 8*, 140699-140725. [Crossref]

[10] Nguyen, D. C., Ding, M., Pathirana, P. N., Seneviratne, A., Li, J., & Poor, H. V. (2021). Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials, 23*(3), 1622-1658. [Crossref]

[11] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., & Shmatikov, V. (2020, June). How to backdoor federated learning. In *International conference on artificial intelligence and statistics* (pp. 2938-2948). PMLR.

[12] Shejwalkar, V., & Houmansadr, A. (2021, January). Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*. [Crossref]

[13] Mahlool, D. H., & Abed, M. H. (2022). A comprehensive survey on federated learning: Concept and applications. *Mobile Computing and Sustainable Informatics: Proceedings of ICMCSI 2022*, 539-553. [Crossref]

[14] Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H. R., Albarqouni, S., ... & Cardoso, M. J. (2020). The future of digital health with federated learning. *NPJ digital medicine, 3*(1), 119. [Crossref]

[15] Nguyen, V. D., Sharma, S. K., Vu, T. X., Chatzinotas, S., & Ottersten, B. (2020). Efficient federated learning algorithm for resource allocation in wireless IoT networks. *IEEE Internet of Things Journal, 8*(5), 3394-3409. [Crossref]

[16] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., ... & Roselander, J. (2019). Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1, 374-388.

[17] Dutta, L., & Bharali, S. (2021). Tinyml meets iot: A comprehensive survey. *Internet of Things, 16*, 100461. [Crossref]

[18] Zhang, W., Lu, Q., Yu, Q., Li, Z., Liu, Y., Lo, S. K., ... & Zhu, L. (2020). Blockchain-based federated learning for device failure detection in industrial IoT. *IEEE Internet of Things Journal, 8*(7), 5926-5937. [Crossref]

[19] Qi, Q., & Chen, X. (2022). Robust design of federated learning for edge-intelligent networks. *IEEE Transactions on Communications, 70*(7), 4469-4481. [Crossref]

[20] Yin, F., Lin, Z., Kong, Q., Xu, Y., Li, D., Theodoridis, S., & Cui, S. R. (2020). FedLoc: Federated learning framework for data-driven cooperative localization and location data processing. *IEEE Open Journal of Signal Processing, 1*, 187-215. [Crossref]

[21] Liu, Z., Guo, J., Yang, W., Fan, J., Lam, K. Y., & Zhao, J. (2022). Privacy-preserving aggregation in federated learning: A survey. *IEEE Transactions on Big Data*. [Crossref]

[22] El Ouadrhiri, A., & Abdelhadi, A. (2022). Differential

privacy for deep and federated learning: A survey. *IEEE access, 10*, 22359-22380. [Crossref]

[23] Kanagavelu, R., Li, Z., Samsudin, J., Yang, Y., Yang, F., Goh, R. S. M., ... & Wang, S. (2020, May). Two-phase multi-party computation enabled privacy-preserving federated learning. In *2020 20th IEEE/ACM international symposium on cluster, cloud and internet computing (CCGRID)* (pp. 410-419). IEEE. [Crossref]

[24] Mazzocca, C., Romandini, N., Mendula, M., Montanari, R., & Bellavista, P. (2023). TruFLaaS: Trustworthy federated learning as a service. *IEEE Internet of Things Journal, 10*(24), 21266-21281. [Crossref]

[25] Zhang, J., Chen, J., Wu, D., Chen, B., & Yu, S. (2019, August). Poisoning attack in federated learning using generative adversarial nets. In *2019 18th IEEE international conference on trust, security and privacy in computing and communications/13th IEEE international conference on big data science and engineering (TrustCom/BigDataSE)* (pp. 374-380). IEEE. [Crossref]

[26] Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., & Zhou, Y. (2019, November). A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security* (pp. 1-11). [Crossref]

[27] Liu, Y. N., Wang, Y. P., Wang, X. F., Xia, Z., & Xu, J. F. (2019). Privacy-preserving raw data collection without a trusted authority for IoT. *Computer Networks, 148*, 340-348. [Crossref]

[28] Li, X., Huang, K., Yang, W., Wang, S., & Zhang, Z. (2019). On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*. [Crossref]

[29] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., ... & Ramage, D. (2018). Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*. [Crossref]

[30] Fusco, P., Rimoli, G. P., & Ficco, M. (2025). TinyML and federated learning for resource-constrained medical devices. In *Artificial Intelligence Techniques for Analysing Sensitive Data in Medical Cyber-Physical Systems: System Protection and Data Analysis* (pp. 113-126). Cham: Springer Nature Switzerland. [Crossref]

[31] Wang, H., Kaplan, Z., Niu, D., & Li, B. (2020, July). Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on computer communications* (pp. 1698-1707). IEEE. [Crossref]

[32] Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W. H., Leung, K. K., & Tassiulas, L. (2022). Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems, 34*(12), 10374-10386. [Crossref]

[33] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016, October). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 308-318). [Crossref]

[34] Short, A. R., Leligou, H. C., & Theocharis, E. (2021, January). Execution of a Federated Learning process within a smart contract. In *2021 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 1-4). IEEE. [Crossref]

[35] Klaine, P. V., Xu, H., Zhang, L., Imran, M., & Zhu, Z. (2023). A privacy-preserving blockchain platform for a data marketplace. *Distributed Ledger Technologies: Research and Practice, 2*(1), 1-16. [Crossref]

[36] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30.

[37] Fang, M., Cao, X., Jia, J., & Gong, N. (2020). Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX security symposium (USENIX Security 20)* (pp. 1605-1622). https://www.usenix.org/conference/usenixsecurity20/presentation/fang

[38] Chen, Y., Su, L., & Xu, J. (2017). Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems, 1*(2), 1-25. [Crossref]

[39] Niknam, S., Dhillon, H. S., & Reed, J. H. (2020). Federated learning for wireless communications: Motivation, opportunities, and challenges. *IEEE Communications Magazine, 58*(6), 46-51. [Crossref]

[40] Konečný, J., McMahan, H. B., Ramage, D., & Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*. [Crossref]

[41] Sattler, F., Wiedemann, S., Müller, K. R., & Samek, W. (2019). Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems, 31*(9), 3400-3413. [Crossref]

[42] Kishor, K. (2022). Communication-efficient federated learning. In *Federated Learning for IoT Applications* (pp. 135-156). Cham: Springer International Publishing. [Crossref]

**Prof. Dr. Kanthavel Radhakrishnan** is having more than 25 years of academic, research and administrative experience in Asia, Middle East, and Oceania. Currently he is working as Professor of Computer Engineering, School of Electrical and Communications Engineering, PNG University of Technology (Public University & Engineers Australia Accredited) , Lae, Papua New Guinea. He received his Post-Doctoral Fellowship from University of Louisiana, USA, and Ph.D Degree from Anna University, India. He has published more than 20 books, 200 articles in international/national journals/conferences and 05 patents. He is the series editor of a number of book series

and serves in various editorial capacities of several international journals. He delivered Key Note addresses in various International Conferences in abroad. His research interests focus on Cloud computing, Machine-deep learning and Intelligent IoT and Wireless Sensor Networks. (Email: kanthavel2005@gmail.com)

**Dr. Dhaya Ramakrishnan** has more than 19 years of academic, research, and administrative experience in Asia, the Middle East, and Oceania. She is currently working as Senior Faculty of Computer Engineering, School of Electrical and Communications Engineering, PNG University of Technology (Public University & Engineers Australia Accredited), Lae, Papua New Guinea. She received her Post-Doctoral Fellowship from the University of Louisiana, USA, and her Ph.D. degree from Manonmaniam Sundaranar University, India. She has published more than 20 books and 150 articles in international and national journals and conferences, and holds 04 patents. She serves as series editor of several book series and in various editorial capacities for international journals. Her research interests focus on cloud computing, artificial intelligence, embedded systems, machine learning, deep learning, wireless sensor networks, and network security. She received the Young Engineer Award from the Institution of Engineers (IEI), Kolkata, India. (Email: dhayavel2005@gmail.com)

**Dr. R. Adline Freeda** has 25 years of teaching and research experience and she has received the M.E in Computer Science and engineering and Ph.D. in Computer science and engineering from Hindustan Institute of Technology & Science, Tamil Nadu, India. Her current research interests include Machine learning, Software Engineering and Cloud Computing. She has published more articles in international/national journals/conferences and patents. (Email: adlinefreeda@gmail.com)