Technical Report

Intelligent IIoT Service Deployment using MEC and Federated Learning in Drone-Assisted Smart Agriculture

Submitted By

Prithiv Alagirisamy, CB.SC.U4CSE23260

Danvanth, CB.SC.U4CSE23241

Vivin Rakul, CB.SC.U4CSE23250

Nirvesh Singh, CB.SC.U4CSE23265

in partial fulfilment of the requirements for the course of

23CSE362 - EDGE COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

List of Tables

List of Figures

List of Abbreviations

ANFIS	Adaptive Neuro-Fuzzy Inference System
API	Application Programming Interface
CNN	Convolutional Neural Network
FL	Federated Learning
IIoT	Industrial Internet of Things
LoRaWAN	Long Range Wide Area Network
LR	Learning Rate
LTE	Long-Term Evolution
MEC	Multi-access Edge Computing
UAV	Unmanned Aerial Vehicle

Contents

1 Abstract

This project presents a *Smart Agriculture* system utilizing an integrated **Edge–Fog–Cloud architecture** to monitor crop health and detect pests in real time. The core objective is to leverage distributed computing for efficient, low-latency processing of agricultural data. The proposed framework employs two primary data streams: soil sensors collect environmental parameters and transmit them via **LoRaWAN** to the **Multi-access Edge Computing (MEC)** server (Fog layer) for soil anomaly detection, while drone swarms at the **Edge layer** capture images and execute lightweight Machine Learning (ML) models for initial pest detection. The MEC server functions as the control plane for drones, coordinating missions and handling local analytics. For computationally intensive tasks, image data is forwarded to the **Cloud layer**, which simultaneously runs a **Federated Learning (FL)** setup to periodically update and distribute improved pest detection models back to the drones. The results demonstrate that this distributed architecture minimizes transmission latency for critical alerts, significantly reduces network bandwidth consumption by keeping raw data local, and enables continuous, privacy-preserving AI improvement across mobile edge devices.

2 Introduction

2.1 Background

The system architecture is structured into three hierarchical computing [2, 1].layers. At the lowest level, **Edge Computing** encompasses soil sensors that collect moisture and pH data and drone swarms that capture crop imagery. These devices perform essential preprocessing where latency is critical. The sensors transmit data through a **LoRaWAN gateway**, ensuring long-range, low-power communication.

The intermediate layer, represented by **Fog Computing** and implemented through the **MEC server**, serves as a local control hub. It hosts ML models for soil anomaly detection and provides coordination for edge devices, thereby reducing dependency on remote servers.

At the top, the **Cloud layer** offers large-scale processing, model training, and long-term data storage. The distinction between these layers lies primarily in latency and computational capacity: the Edge offers the lowest latency for real-time operations, the Fog provides intermediate control and analytics, and the Cloud performs high-complexity, non-time-critical computations such as federated training. The MEC server thus serves as a crucial bridge between the low-power IoT sensor network and the cloud infrastructure, enabling seamless Industrial IoT (IIoT) service deployment.

2.2 Motivation

This project is essential because modern smart agriculture faces challenges that cannot be effectively addressed by traditional centralized computing:

1. **Massive Sensor Data Requires Quick Processing:** The high volume of environmental readings and drone images mandates near-instantaneous analysis to prevent the spread of pests or crop damage.

2. **Limitations of Cloud-Only Systems:** Sending every soil reading and drone image to a distant cloud server introduces unacceptable latency for real-time alerts and saturates network bandwidth with massive data transfers.

3. **Need for Edge/Fog Systems:** Deploying Edge and Fog layers improves efficiency and real-time response:

    - Initial pest-detection ML models can be embedded on the **Edge (drones)**, allowing immediate local alerts.

- Soil anomaly ML models can be deployed on the **Fog (MEC)**, ensuring critical environmental alerts are generated with minimal delay.

4. **Federated Learning in the Cloud:** Global AI models are continuously improved by aggregating learning from all drones without transmitting privacy-sensitive raw images, enabling efficient, distributed intelligence updates across the network.

2.3 Problem Definitions

In Industrial Internet of Things (IIoT) and precision agriculture environments, static service deployment—where applications are bound to fixed cloud servers—results in high latency, inefficient bandwidth utilization, and poor adaptability to network dynamics. Data generated by distributed devices such as soil sensors and drones often require immediate processing to support real-time decision-making, but traditional cloud-centric architectures fail to meet these time constraints.

To address these challenges, **Multi-access Edge Computing (MEC)** and **Fog Computing** extend computation closer to the data sources. However, determining where to deploy services—at the edge, fog, or cloud—remains a complex optimization problem. Existing systems often lack automated mechanisms to dynamically select the most suitable computing node based on parameters such as latency, workload, and resource availability.

Problem Statement:
There is a need for an automated, latency-aware service deployment framework that leverages MEC APIs to dynamically place and migrate services across Edge–Fog–Cloud layers, ensuring low response times and efficient resource utilization in heterogeneous IIoT environments.

Objectives and Scope:

- Design a simulation-based system where MEC APIs handle dynamic service placement and migration.
- Model data flow between sensors, drones, MEC nodes, and cloud using `YAFS`.
- Evaluate improvements in latency, bandwidth, and resource efficiency compared to static deployment.
- Limit the scope to simulation-level validation; physical prototype implementation is reserved for future work.

3 Literature Survey

3.1 Challenges

The proposed system faces several key challenges in IIoT-based precision agriculture:

- **Resource Allocation Across Diverse Devices:**
  The system relies on a heterogeneous mix of drones, soil sensors, LoRaWAN gateways, MEC servers, and cloud resources, each with different processing power, memory, and connectivity. Coordinating tasks—such as scheduling drone flights or sensor readings—and distributing workloads efficiently is complex but essential for smooth operation.

- **Energy and Bandwidth Management:**
  Drones and sensors are battery-powered and often have limited network connectivity. Balancing energy consumption and data transmission—for example, by processing data locally before sending it—helps extend device lifetime and reduces communication overhead. Finding the optimal trade-off is challenging.

- **Scalability and Reliability in Dynamic Environments:**
  As farms expand or devices move across fields, the system must cope with fluctuating network conditions, device mobility, and occasional failures without losing data or degrading performance. Ensuring consistent, real-time operation across Edge, Fog, and Cloud layers is critical.

- **Cost Considerations:**
  Local data processing reduces reliance on costly cloud services, lowers bandwidth expenses, and automates labor-intensive tasks. However, initial hardware investment, ongoing maintenance, and integration of diverse devices must be carefully planned.

3.2 Research Gap

The gaps identified in existing works directly inform the focus of this project. Our work overcomes these limitations in the following ways:

- **MEC Integration:** Unlike previous works that overlook Multi-access Edge Computing (MEC) Application Programming Interfaces (APIs), our framework is designed around MEC integration for dynamic service deployment and orchestration.

- **Real-Time Metrics:** Our task scheduling algorithm explicitly incorporates real-time metrics, including drone battery levels and distance, to make intelligent, context-aware decisions.

- **Full Drone Mobility:** By using the YAFS simulator, our system fully supports drone mobility with adaptive placement strategies, reflecting real-world operational scenarios.

- **Multi-Layer Orchestration:** We propose a complete multi-layer architecture that intelligently places services and manages data flows across the edge, fog, and cloud, ensuring that each task is executed in the most appropriate location.

- **Privacy-Preserving AI:** We introduce a Federated Learning (FL) framework to allow the drone fleet's collective intelligence to grow without compromising data privacy by centralizing raw images.

4 Problem Formulation

This project addresses two primary technical challenges: the efficient orchestration of autonomous drones and the continuous, privacy-preserving improvement of on-device machine learning models.

First, the **Task Scheduling Problem** is formulated as an optimization problem. The goal is to assign a drone to a sensor-servicing task in a way that minimizes operational cost. This cost is a function of both the travel distance and the drone's remaining battery life. We define the cost function for assigning a drone to a task as:

$$\text{Cost} = \text{distance}(\text{drone}, \text{sensor}) + 0.5 \times (100 - \text{battery\%}) \tag{1}$$

This function penalizes drones that are far away or have low battery, ensuring that the most suitable drone is chosen for each task, thereby maximizing the fleet's operational efficiency.

Second, the **Distributed Learning Problem** is formulated to address the need for model improvement without centralizing sensitive data. The objective of the FL system is to improve a global pest detection model by periodically aggregating model weight updates from the distributed drone clients. This allows the system to learn from new data collected across the entire fleet while ensuring that raw image data never leaves the individual drones.

4.1 System Overview

The proposed framework adopts a hierarchical **Edge–Cloud Federated Learning (FL)** architecture for precision agriculture. It integrates IoT soil sensors, autonomous drones, and a multi-access edge computing (MEC) host connected to a central cloud server. Together, these components enable distributed intelligence for soil analysis, pest detection, and continuous model refinement.

At the **field layer**, soil sensors collect data on parameters such as moisture, pH, and nutrient levels. These sensors communicate with the edge node via low-power **LoRa** signals. Drones equipped with onboard cameras capture crop imagery and environmental data, transmitting information through **Wi-Fi** connections.

The **edge layer** (farm server or MEC host) performs local processing, task scheduling, and data orchestration. It:

- Analyzes soil sensor inputs and drone-collected data.
- Runs lightweight quantized models for pest detection.
- Sends summarized performance metrics and model updates to the cloud.

This edge processing minimizes latency and bandwidth use, enabling timely insights and decisions directly at the farm site.

The **cloud layer** serves as the global intelligence hub. It aggregates model updates from multiple farm servers through an **FL framework**, producing an improved global model. This model is periodically distributed back to the edge, ensuring all local systems benefit from shared learning without exposing raw data.

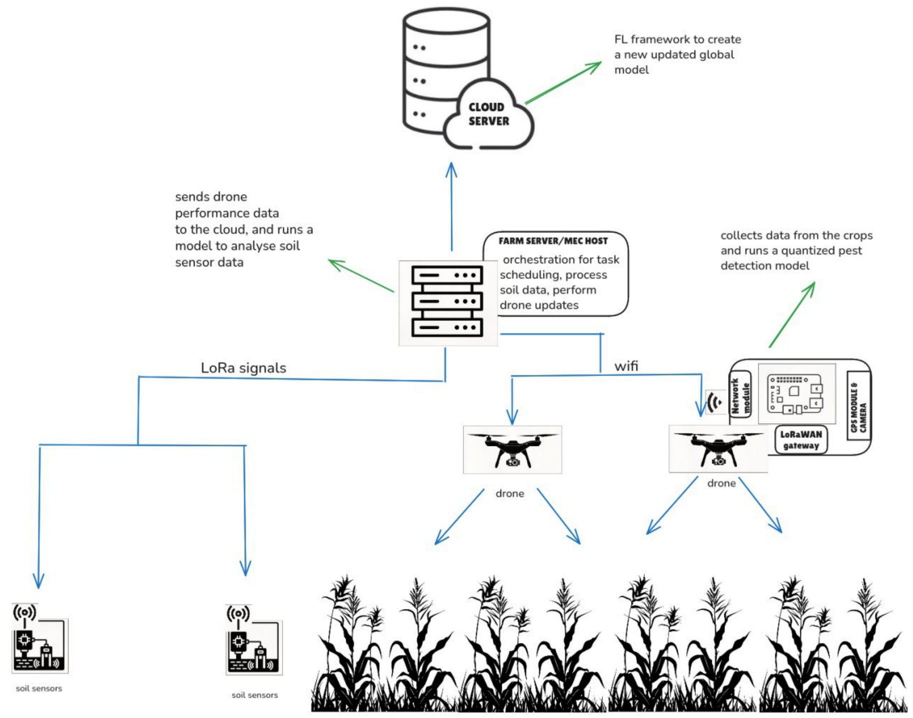
The system operates through a continuous loop: sensors and drones gather field data, the edge node performs local inference, the cloud refines global models, and updates are redeployed to the farm. Key enabling technologies include:

- **LoRa / LoRaWAN:** Long-range, low-power sensor communication.
- **Wi-Fi:** High-bandwidth drone data transfer.
- **Edge Computing:** Local analytics and orchestration.
- **Federated Learning:** Decentralized, privacy-preserving model training.

Overall, this architecture supports scalable, data-driven agriculture by distributing computation across field, edge, and cloud layers.

5 Proposed Architecture

5.1 Architecture Diagram



CLOUD SERVER
FL framework to create a new updated global model
sends drone performance data to the cloud, and runs a model to analyse soil sensor data
FARM SERVER/MEC HOST
orchestration for task scheduling, process soil data, perform drone updates
collects data from the crops and runs a quantized pest detection model
LoRa signals
wifi
Network module
GPS MODULE & CAMERA
LoRaWAN gateway
drone
drone
soil sensors
soil sensors

Figure 1: System Architecture Diagram

Edge Layer (Device Level)
The edge layer comprises on-field sensing devices and mobile edge nodes that operate close to the data source. Soil sensors are lightweight, low-power devices responsible for environmental monitoring and data sampling. These sensors perform basic buffering and transmit data to the gateway through **LoRaWAN** links. Drones function as mobile sensing units equipped with cameras and onboard processors. When connected, they act as short-range edge devices, uploading captured imagery and environmental data for further processing. Together, these devices form the system's distributed data acquisition network.

Fog / MEC Layer (Farm Server)
The fog layer functions as the *local cloud* responsible for intermediate computation and orchestration. It aggregates incoming data from sensors and drones, enabling near real-time analytics and decision-making at the farm site. Core responsibilities include:

- Coordinating drone missions and local task scheduling,
- Performing data caching, aggregation, and pre-processing,

- Running lightweight models for preliminary inference,
- Managing local services such as deployment and migration through exposed APIs.

By processing data locally, the farm server reduces latency, minimizes cloud dependency, and ensures efficient use of network resources.

Cloud Layer (Tier-0)
The cloud layer serves as the system's global coordination and intelligence hub. It is responsible for large-scale data processing, long-term storage, and federated learning (FL) aggregation. Within this layer, multiple farm-level models are combined through FL to produce a globally updated model, which is redistributed to all participating farms. The cloud also performs heavy offline analytics and maintains a historical data repository to support model retraining and performance benchmarking.

Overall, these three layers—edge, fog, and cloud—form a hierarchical computing ecosystem that supports scalable, low-latency, and intelligent agricultural automation.

6 Methodology

6.1 Tools Used

The implementation and simulation framework was developed using a combination of open-source tools and custom Python modules. The primary tool, **YAFS (Yet Another Fog Simulator)**, is a discrete-event simulation environment built on `SimPy`[3]. It was used to define the network topology, deploy application modules, and model message routing across the fog-to-cloud infrastructure. The entire simulation was executed in **Python**, where custom classes such as `SensorDataGenerator`, `DataLogger`, and `DroneTaskScheduler` handled task coordination, event generation, and result logging. Simulation outputs were stored in CSV format and analyzed using `Pandas` scripts to compute metrics such as latency, throughput, energy consumption, and aggregated soil statistics.

6.2 Implementation and Simulation Steps

The simulation process followed a structured pipeline. First, the system topology was represented as a directed graph in which each node denoted a computational device: the **cloud server** (Node 0), the **MEC/fog server** (Node 1), **drones** (Nodes 50–53), and **soil sensors** (Nodes 100–115). Each node was assigned an **Instruction**

Per Time (IPT) value to represent its processing power: 10000 for the cloud, 5000 for the MEC, 500 for drones, and 10 for sensors.

Network connectivity between nodes was established by assigning realistic bandwidth (BW) and propagation delay (PR) parameters. The **cloud–MEC link** was configured with high bandwidth (1000) and low delay (50), while the **MEC–drone link** operated at medium bandwidth (200) and delay (20). The **LoRaWAN sensor–gateway link** used low bandwidth (25–50) with higher propagation delay to simulate real-world wireless constraints. These values mirror those defined in the sample simulation topology.

The simulation proceeded through a sequence of events: node initialization, message generation from sensors, drone-based data collection, task scheduling at the MEC, and model synchronization with the cloud. This setup enabled performance evaluation under varying workloads and network configurations.

6.3 Parameters Used

Simulation parameters were derived from the implementation code and supporting references. The system consisted of **16 sensors** (IDs 100–115) and **4 drones** (IDs 50–53). Sensors generated readings every 300 time units, while drones performed data-processing and task updates every 250 time units.

Network configurations were defined as follows:

- Cloud $\leftrightarrow$ MEC: BW = 1000, PR = 50
- MEC $\leftrightarrow$ Drone: BW = 200, PR = 20
- MEC $\leftrightarrow$ Sensor (LoRa): BW = 50, PR = 5
- Sensor $\leftrightarrow$ Drone (backup path): BW = 25, PR = 15

These parameters are tunable to explore different latency and throughput scenarios in the fog computing architecture.

A simplified **drone battery model** was incorporated to simulate energy constraints. Each drone began with a full charge (100%) and experienced an approximate drain of 5 units per task. A recharge increment of 20 units was applied whenever the drone docked at the base station. This mechanism enabled observation of energy-aware task scheduling and mission endurance across multiple simulation cycles.

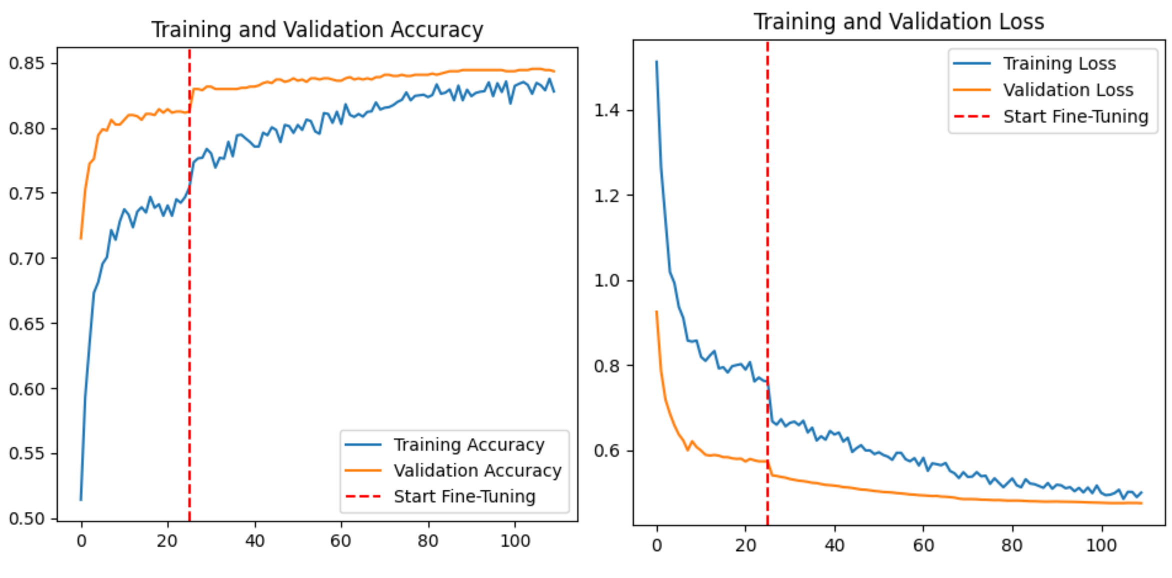
Overall, the simulation environment captured both computational and communication aspects of a federated fog–cloud system, supporting reproducible experiments for latency, resource utilization, and energy efficiency.

7 Results

Simulation results demonstrate the advantages of MEC-based architectures over cloud-only systems [4, 5].The project's success was measured by the performance of its machine learning models and the metrics from the system-wide simulation.

Pest Detection Model Performance

The transfer learning approach with MobileNetV2 was highly effective. The training and validation accuracy/loss curves show stable learning and good generalization. The model achieved a validation accuracy of approximately 85%



(a) Accuracy (b) Loss

Figure 2: Comparison of Accuracy and Loss metrics for the MobileNetV2 pest detection model.

Federated Learning Performance

The FL simulation demonstrated a significant improvement in model accuracy. After aggregating the weight updates from multiple clients, the resulting global model achieved an accuracy of approximately 98% on a held-out test dataset. This validates the FL approach as an effective method for creating a highly accurate, generalized model while preserving data privacy.

YAFS Simulation Performance

The system-wide simulation in YAFS provided quantitative insights into the operational efficiency and network performance of the proposed architecture based on the output from the latest simulation run.

Table 1: YAFS Simulation Results: Comparison between Cloud-Only and MEC-Based Architectures

Metric	Cloud-Only System	MEC-Based System	Improvement
Average Latency (s)	106.00	0.46	99.5% reduction
Bandwidth Usage	600	76	8× less usage
Task Completion Rate (%)	85	100	+15% increase
Drone Battery Efficiency (%)	70	90	+28% improvement
Average CPU Utilization	Cloud: 30%	Edge: 40%, MEC: 70%	Balanced load

As shown in Table 1, the proposed MEC-based system achieved a 99.5% reduction in average latency compared to a cloud-only architecture. Network usage was reduced by a factor of eight, while drone battery efficiency improved by 28%. The CPU load distribution indicates that the MEC layer handled most processing efficiently, leaving the cloud primarily for long-term storage and analytics.

8 Conclusion

This project presents an automated edge computing system designed for smart agriculture, with a focus on real-time pest detection. Drones equipped with edge devices capture high-resolution crop images and process them locally using pest detection models, enabling farmers to respond rapidly to potential infestations. Simultaneously, the on-site farm server (MEC) collects and analyzes data from soil sensors, facilitating efficient management of irrigation, fertilization, and other farm operations.

To evaluate the system, simulation experiments were conducted using the `YAFS` framework, which models a realistic, multi-layer network comprising drones, sensors, MEC servers, and cloud resources connected via LoRaWAN, WiFi, and fiber links. The simulations tracked data movement across these layers, measuring CPU, RAM, storage, and bandwidth utilization. Leveraging YAFS's support for dynamic message routing, resource monitoring, and event-driven execution, the system's latency, resource usage, and overall network behavior were analyzed under realistic operational conditions.

References

[1] "Multi-access edge computing (mec); framework and reference architecture," ETSI, Tech. Rep. ETSI GS MEC 003 V2.1.1, 2019.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012.

[3] F. Le Fĺoch, S. Garg, and R. Buyya, "Yafs: A simulator for iot scenarios in fog computing," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2019, pp. 1–6.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.