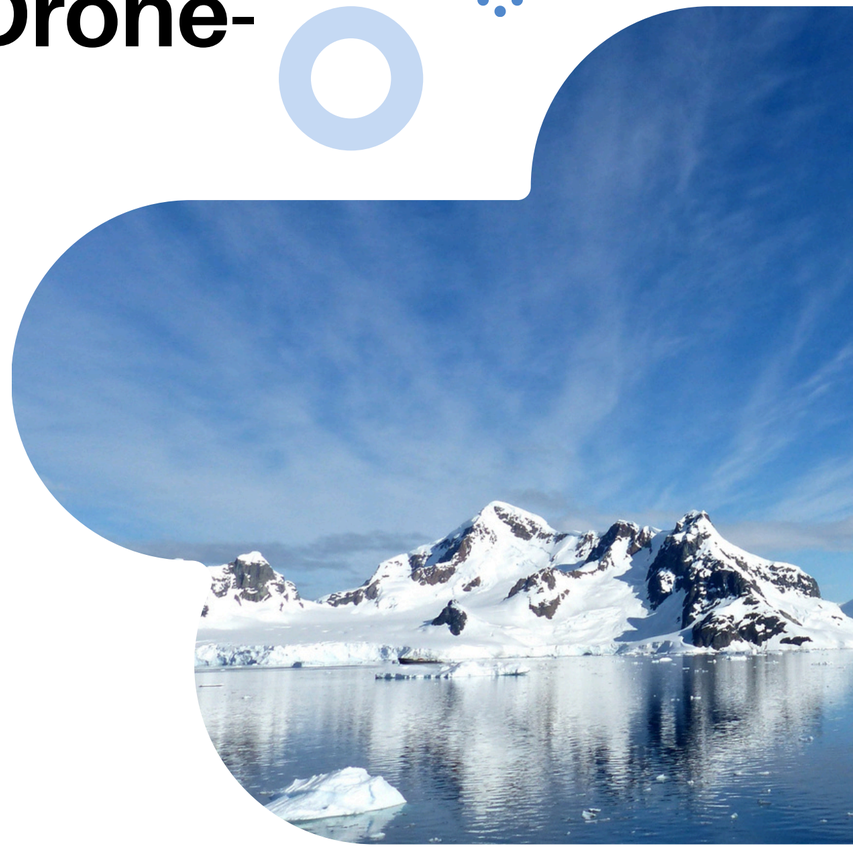
Intelligent IIoT Service Deployment using MEC and Federated Learning in Drone-Assisted Smart Agriculture



Team No. 16

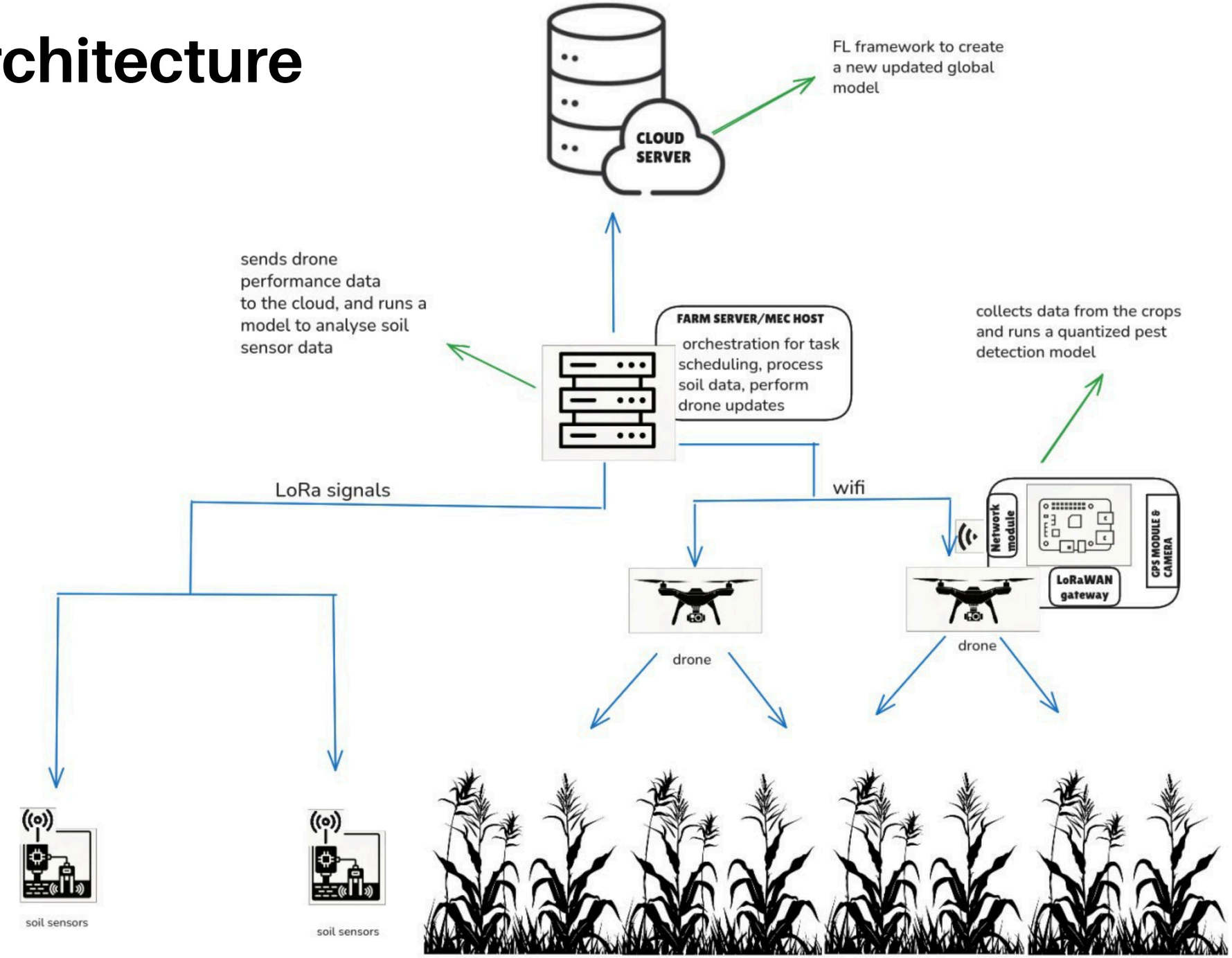
EDGLERS

PROBLEM STATEMENT

Traditional cloud-based farming systems suffer from high latency, unreliable connectivity, and limited scalability, making real-time decision-making difficult in large or remote fields.

An edge computing framework can overcome these challenges by processing data locally, enabling faster, smarter, and more responsive farm automation and pest detection.

Edge Architecture



Task Scheduling:

The greedy algorithm assigns each sensor task to the nearest available drone, aiming for quick, locally optimal assignments without global optimization cycles.

Reasoning: Used to find a globally optimal solution for assigning drones to collect data from sensors. It minimizes the total travel distance and energy consumption for the entire drone fleet.

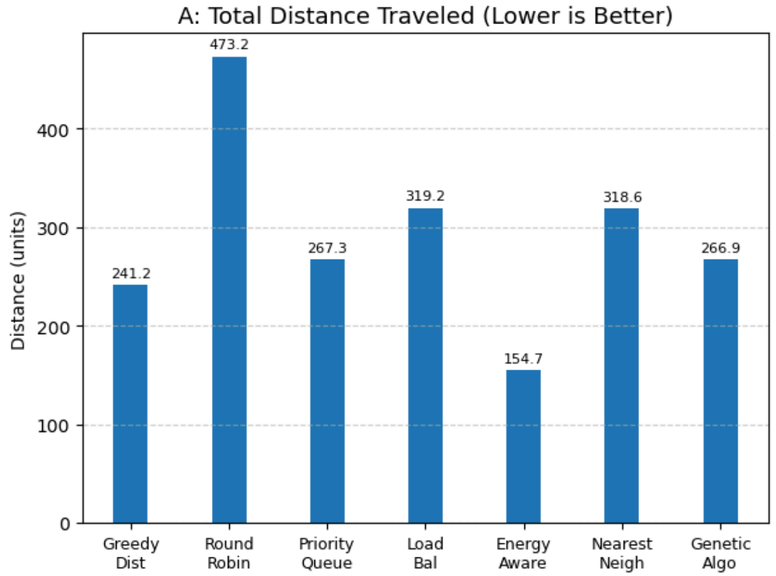
Performance Metric: YAFS simulation shows effective task distribution. For 16 tasks, the algorithm assigned 8 tasks to Drone 52 (123.07 units distance) and 8 tasks to Drone 53 (148.76 units distance), efficiently utilizing available assets.

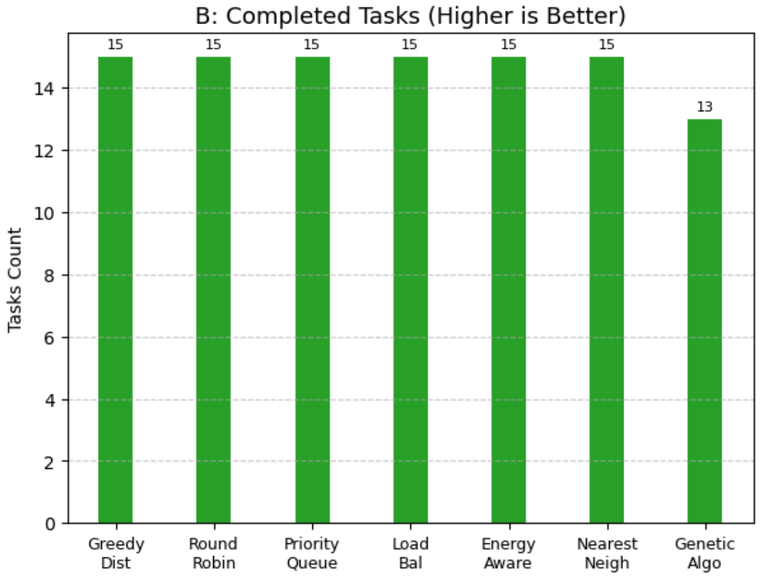
Protocols:

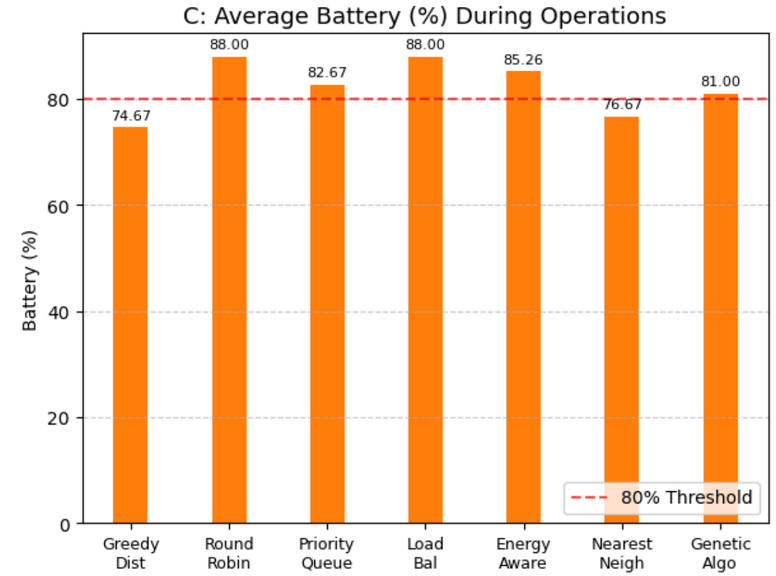
LoRaWAN Gateway Protocol: Used for connecting soil sensors to the MEC server, leveraging low-power, long-range wireless capability for efficient field data gathering.
Reasoning: Selected for its long-range and extremely low power consumption, which is essential for battery-operated soil sensors distributed across a large field. Its low bandwidth is sufficient for transmitting small sensor data packets.
Performance Metric: Simulation shows an average throughput of 1.27 Mbps, appropriate for IoT sensor data.

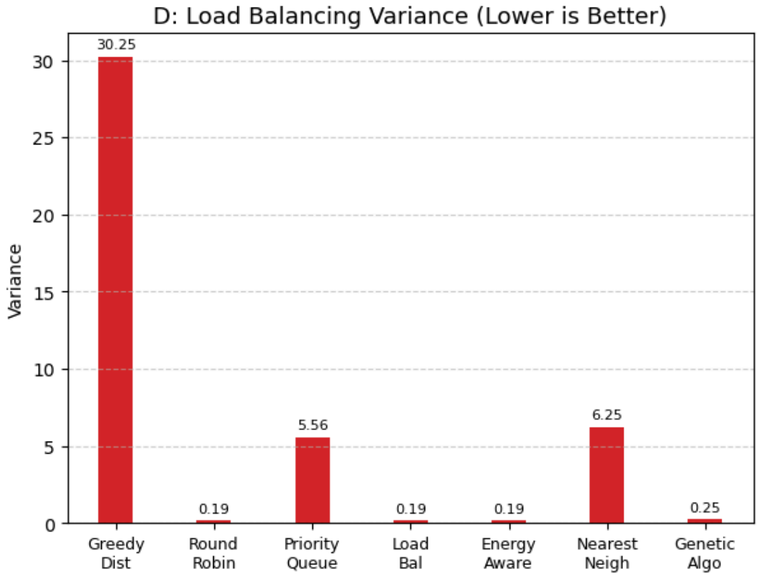
Local LTE Network: Used for drone-to-MEC server links, supporting higher data rates (e.g., image transfers) and low latency for mobile edge nodes (drones).
Reasoning: Chosen to provide the high bandwidth and low latency required for transmitting high-resolution images from drones and receiving real-time control commands from the MEC server.
Performance Metric: Simulation shows an average throughput of 34.58 Mbps for the Drone-to-Fog link, capable of handling image data streams.

Performance Comparison of Drone Task Scheduling Algorithms









Changes from review 2

1.Changes the LoRaWAN receiver from independent drones to the MEC server

2.Added an FL framework that updates the drone models every month

3.Complete model and architecutre integration

Edge AI Applications in Smart Agriculture

1. Federated Learning with a Drone Fleet
Concept: We created a system where a fleet of drones learns to detect pests collaboratively without sending private image data to a central server. This is a key edge computing pattern for privacy and network efficiency
Process: Each drone improves its local model. Only the small model updates are sent to a server, which aggregates them to create a much better global model.
Result: Achieved ~98% accuracy, demonstrating a high-performance, decentralized learning system on edge devices.

2. On-Device Models for Real-Time Analysis

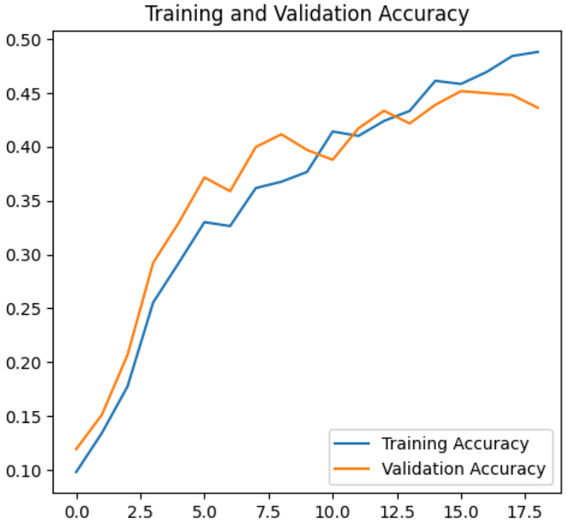
On-Drone Pest Detection:
Developed an efficient CNN (MobileNetV2) to run directly on each drone for immediate pest identification.This on-device inference model achieved ~85% accuracy
On-Microcontroller Soil Analysis:
Built a lightweight Fuzzy Logic model to classify soil pH in real-time on a low-power microcontroller.This demonstrates running AI on resource-constrained edge devices, achieving ~97% accuracy.

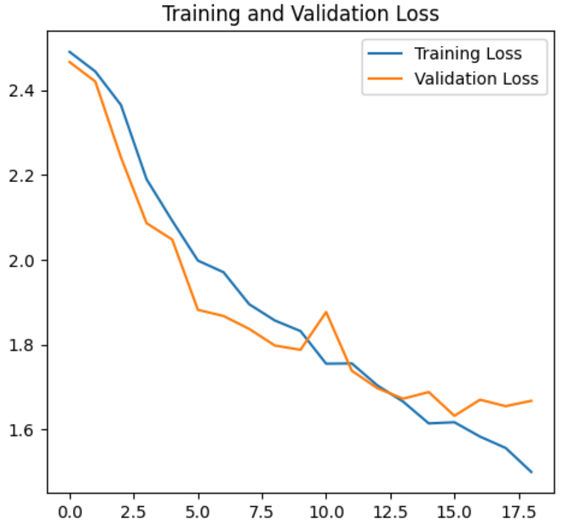
RESULT:

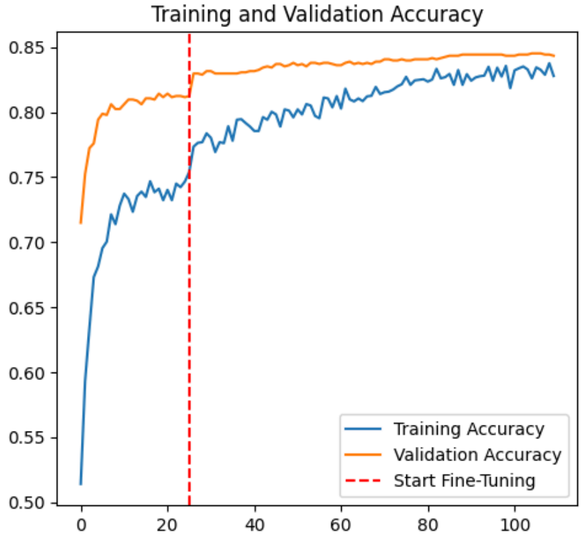
```
Total network transmissions: 76

App DroneProcessing Statistics:
  - Messages processed: 11876
  - Avg transmission time: -0.13
  - Avg service time: 0.46
  - Deployed on nodes: [50 51 52 53  1]

App SensorDataCollection Statistics:
  - Messages processed: 752
  - Avg transmission time: 0.00
  - Avg service time: 100.00
  - Deployed on nodes: [100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115]

=== SOIL SENSOR DATA ANALYSIS ===
Total soil sensor readings: 16
Sensors that generated data: 11
Average soil conditions:
  - pH Level: 6.58 (range: 5.9-7.5)
  - Moisture: 36.4% (range: 14.9-51.6%)
  - Temperature: 21.6°C (range: 12.0-30.5°C)
  - Nitrogen: 21.9 ppm
  - Phosphorus: 14.5 ppm
  - Potassium: 30.7 ppm

=== DRONE DATA COLLECTION ANALYSIS ===
Total data collections by drones: 16

Drone 52 Collections:
  - Data packets collected: 8
  - Sensors visited: 6
  - Average battery during collections: 82.5%
  - Sensors covered: [np.int64(100), np.int64(101), np.int64(106), np.int64(107), np.int64(112), np.int64(113)]

Drone 53 Collections:
  - Data packets collected: 8
  - Sensors visited: 7
  - Average battery during collections: 82.5%
  - Sensors covered: [np.int64(103), np.int64(104), np.int64(106), np.int64(107), np.int64(109), np.int64(111), np.int
4(115)]
```

```
=== MEC SERVER PROCESSING ANALYSIS ===

=== DRONE OPERATIONS ===
Total data collections: 16

Drone 52:
  - Collections: 8
  - Sensors visited: 6
  - Avg battery: 77.50%

Drone 53:
  - Collections: 8
  - Sensors visited: 7
  - Avg battery: 77.50%

=== TASK SCHEDULING ===
Total scheduled tasks: 16
Average distance to sensor: 16.99

Drone 52 schedule:
  - Tasks assigned: 8
  - Total distance: 123.07
  - Sensors covered: [113 106 101 112 100 107]

Drone 53 schedule:
  - Tasks assigned: 8
  - Total distance: 148.76
  - Sensors covered: [107 106 104 115 111 109 103]
```

Results








```
[starkiller@starkiller-linux fl_model]$ python drone_client.py
2025-09-18 11:02:41.603617: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn t
, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-09-18 11:02:41.669858: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
AttributeError: 'MessageFactory' object has no attribute 'GetPrototype'
AttributeError: 'MessageFactory' object has no attribute 'GetPrototype'
AttributeError: 'MessageFactory' object has no attribute 'GetPrototype'
AttributeError: 'MessageFactory' object has no attribute 'GetPrototype'
AttributeError: 'MessageFactory' object has no attribute 'GetPrototype'

--- Starting Local Training on drone_2744 ---
/usr/lib/python3.13/site-packages/keras/src/saving/saving_lib.py:797: UserWarning: Skipping variable loading for optimizer 'adam', because it has 2 variables whereas the saved optimizer has 74 variables.
  saveable.load_own_variables(weights_store.get(inner_path))
Successfully loaded global weights from global_model_initial.weights.h5
Simulating collection of new local data...
Found 30 files belonging to 3 classes.
Starting local fine-tuning...
Epoch 1/5
4/4 ━━━━━━━━━━━━━━━━━━━━ 13s 129ms/step - accuracy: 0.1667 - loss: 3.5472
Epoch 2/5
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 118ms/step - accuracy: 0.1333 - loss: 3.3055
Epoch 3/5
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 121ms/step - accuracy: 0.1667 - loss: 2.8302
Epoch 4/5
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step - accuracy: 0.2000 - loss: 3.0405
Epoch 5/5
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 118ms/step - accuracy: 0.0667 - loss: 3.0590
Local fine-tuning complete.
Saved updated weights to drone_2744_update_1758173583.weights.h5
```