Federated Spellcasting

Yaliang Cai, Hugo Sjödin, Vahid Faraji, Haobo Zu

Dec. 19, 2025

EITP40 - IOT

Final Project

Project Overview

Motivation

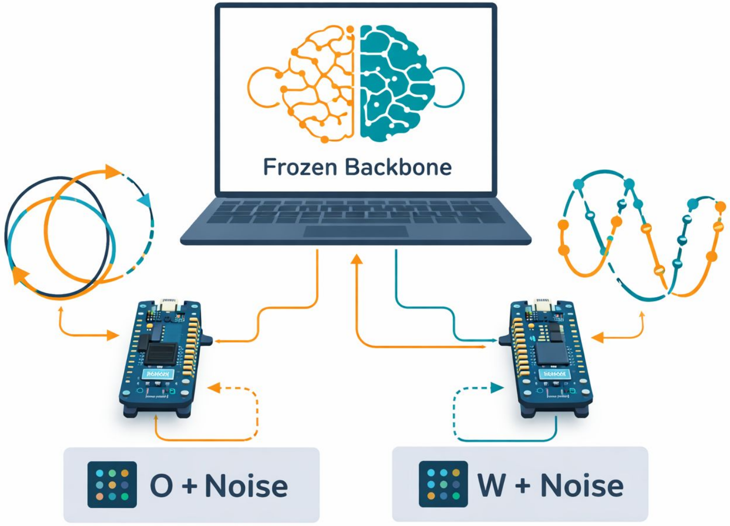
Running ML models directly on low-resource IoT devices is challenging due to limited memory and computation power

Project Goal

To recognize motion-based gestures (W and O) using IMU sensors and demonstrate distributed machine learning between a PC and Arduino devices.

System Overview

The system uses two IMU-equipped Arduino Nano 33 BLE Sense boards and a PC for training, deployment, and inference.

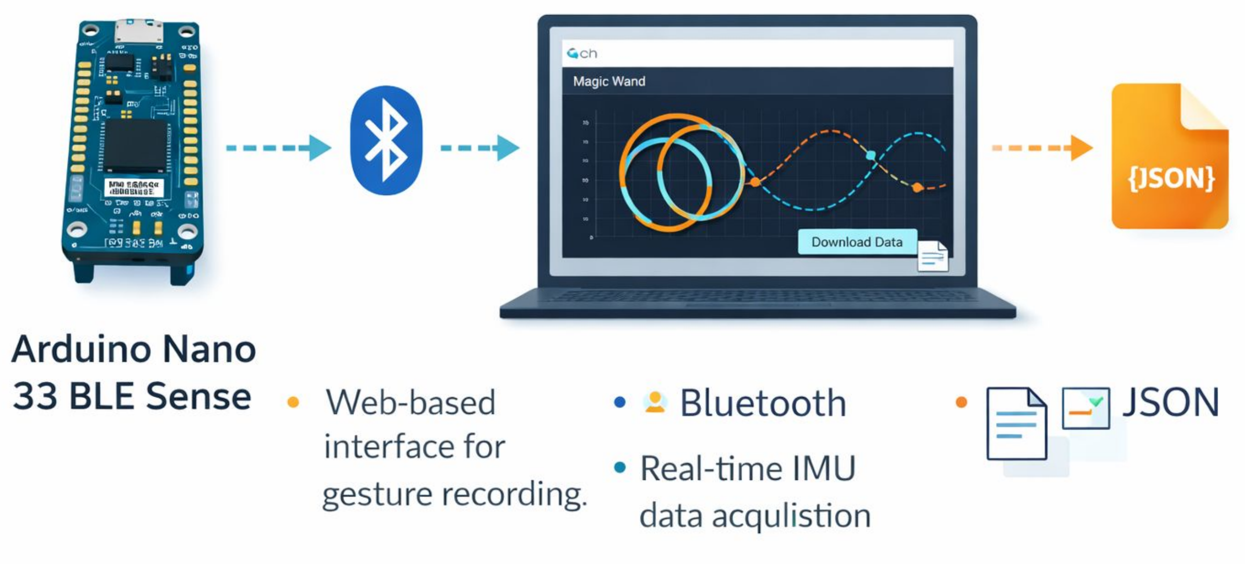


Why This Matters?

The project demonstrates a practical TinyML solution for real-world IoT applications with limited resources.

Data collection

- Web-based interface
- Real-time IMU data
- Bluetooth connection
- Visualization, labeling, and JSON export



Visualization Matters

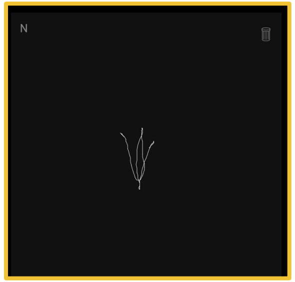
Challenges

Gesture ambiguity
Similar gestures can look different across recordings

Labeling uncertainty
Even with visualization, some samples are hard to label

User variability
Speed and motion style vary between executions

Noise and incomplete gestures
Near-miss movements are difficult to classify



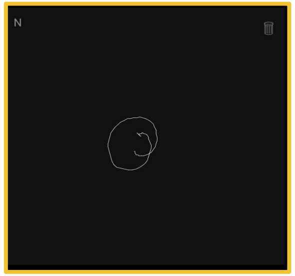
Benefits

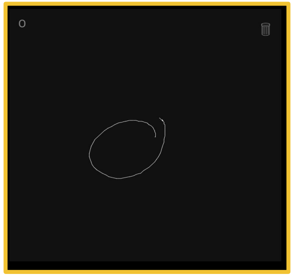
Real-time visualization
Immediate feedback during recording

Improved labeling accuracy
Visual inspection helps reduce obvious errors

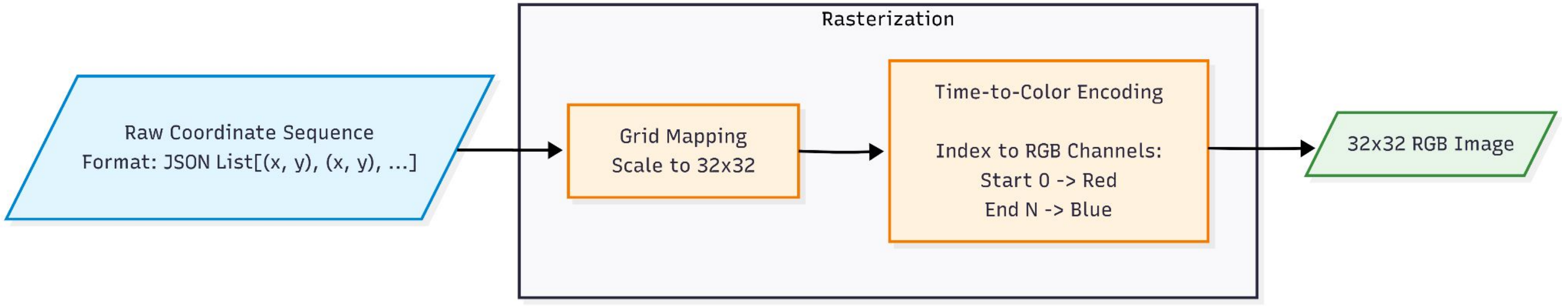
Easy data management
One-click export in JSON format

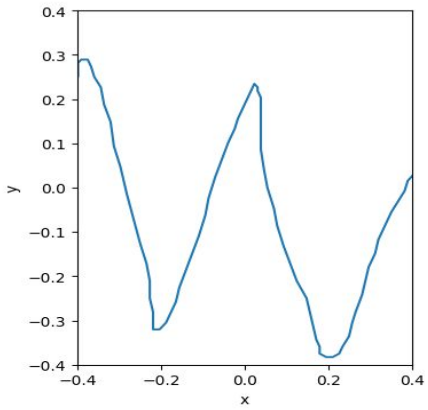
Rapid iteration
Bad samples can be removed instantly

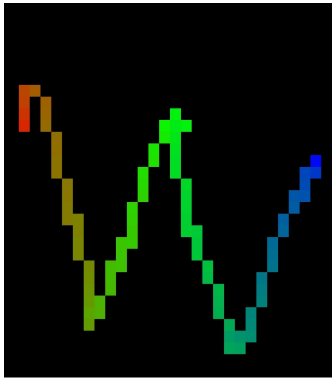


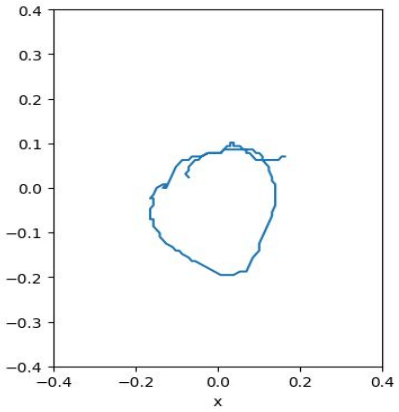


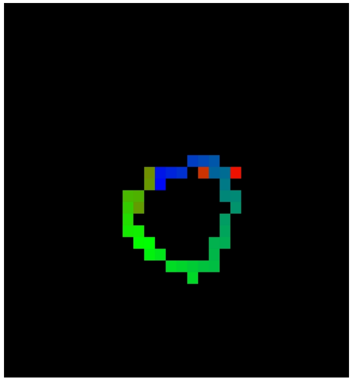
Simplifying the Problem



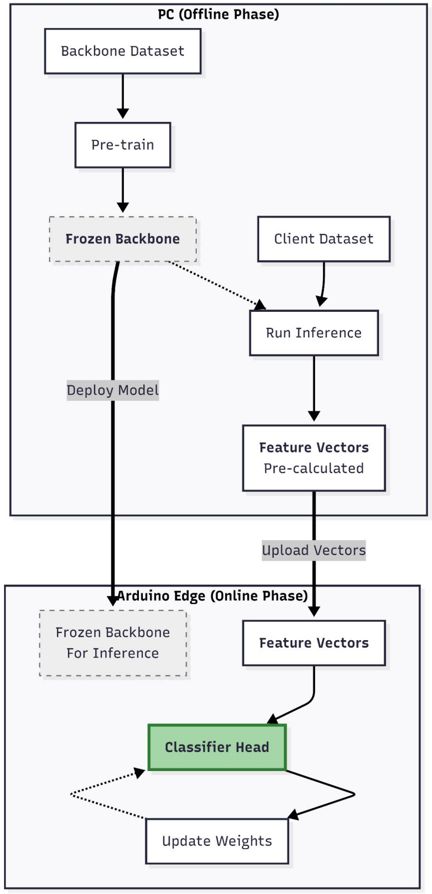








System Pipeline: Hybrid Split-Computing



Offline Phase: PC

- Backbone Pre-training
- Feature Caching

Online Phase: Arduino Edge

- The Training Loop

Benefit: Eliminates the need to run forward passes on the CNN layers during local training.

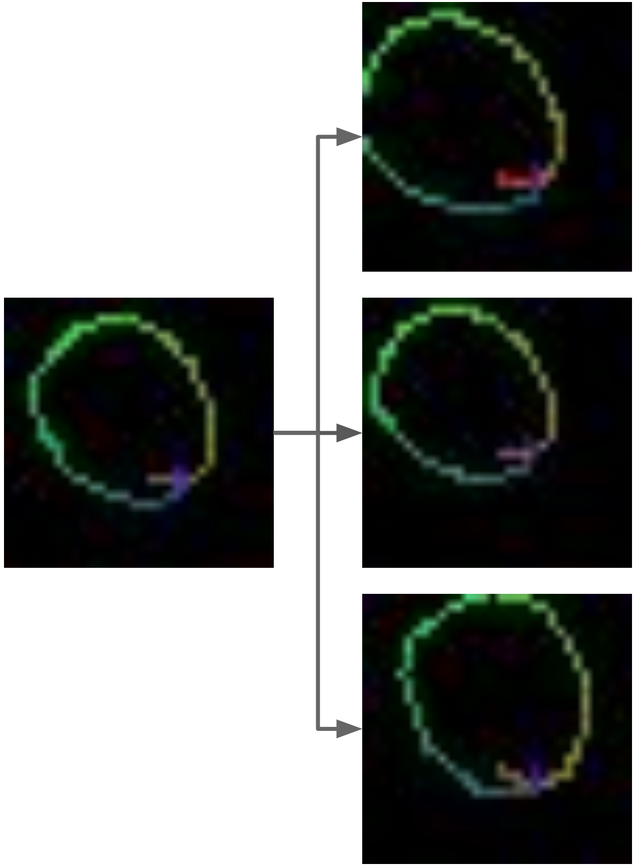
Non-IID Data Distribution

- Non-IID split across Client A (O+Noise) and Client B (W+Noise)
- 50% backbone pre-training on PC (no data leakage)
- Frozen backbone → feature vectors for Arduino
- Efficient TinyML-friendly design

Dataset Name	Quantity	Composition (O / W / Noise)	Storage Location	Data Format	Purpose
Backbone Set	**400** (50%)	100 / 100 / 200	**PC**	Raw Data (Images)	Train the **Backbone**. (CNN feature extractor).
Client Set A	**160** (20%)	80 / 0 / 80	**Arduino A**	**Feature Vectors**	Train **Head A**. (Biased data: knows 'O' only).
Client Set B	**160** (20%)	0 / 80 / 80	**Arduino B**	**Feature Vectors**	Train **Head B**. (Biased data: knows 'W' only).
Test Set	**80** (10%)	20 / 20 / 40	**Arduino A & B**	**Feature Vectors**	**Validate accuracy.** (To verify performance).

The key idea is that heavy representation learning happens once on the PC, while edge devices only handle lightweight, biased local learning using feature vectors.

Data Augmentation



The Challenge: Data Scarcity

- The PC dataset is small (400 samples total).
- May lead to **Overfitting**.

Random transformations

- **Shift**
- **Scaling**
- **Rotation**

Benefit

- **Robustness:** The Backbone learns to recognize the "essence" of the shape (W or O) regardless of size or angle.
- **Generalization:** Ensures the Feature Extractor works well for new, unseen users (Node A and Node B).

ML Algorithm ....

ML Algorithm and Implementation on IoT Sensor

Dividing computational load

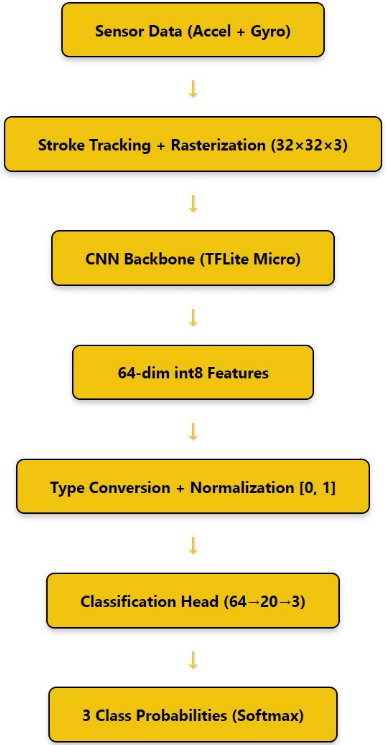
- Training on computer
- vs
- Training on Arduino

Workflow

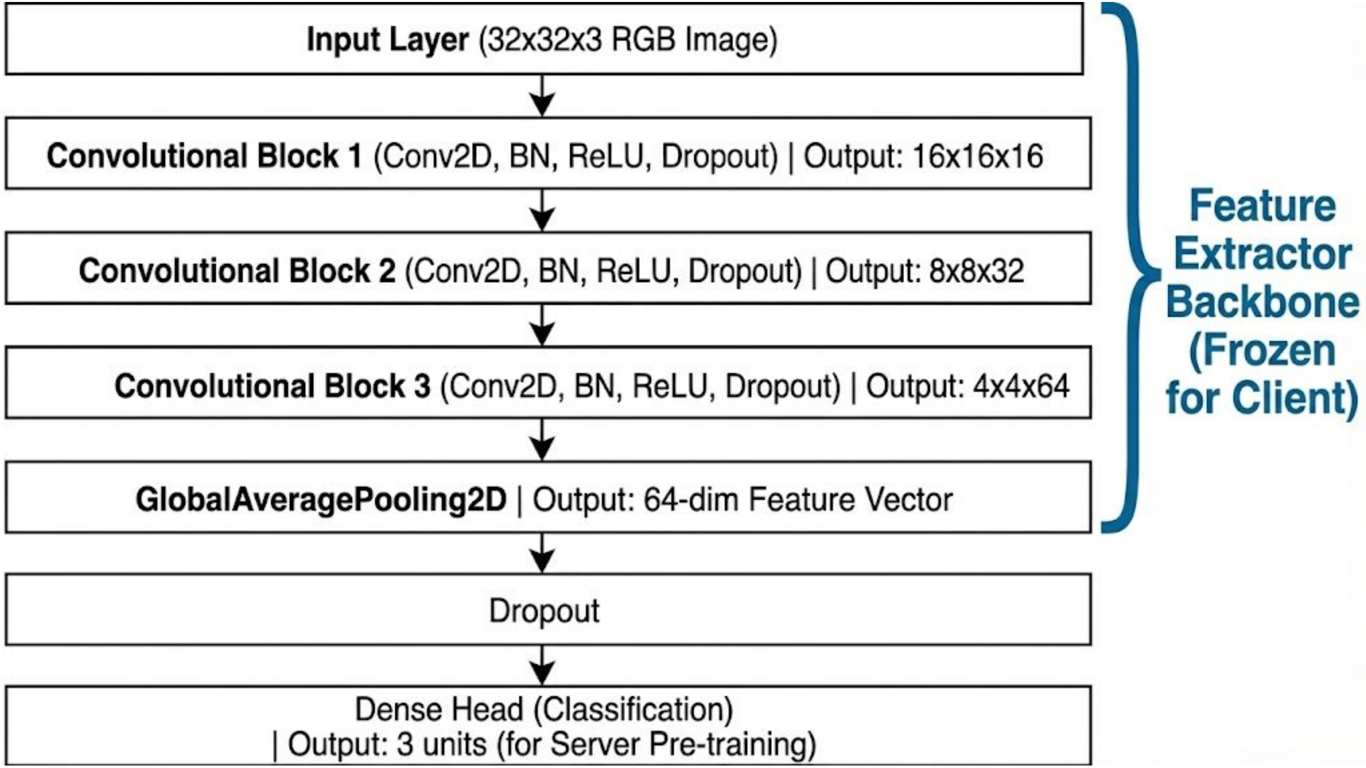
- Dividing simplifies workflow and building of models

Problem

- How do we combine the two?



CNN Architecture



Federated Learning

Decentralized Collaborative Training on Edge

Enables two independent edge nodes to collaboratively train a shared neural network using the **FedAvg** algorithm.

Model weights are fused via Bluetooth Low Energy (BLE) without ever exposing raw local training datasets to the peer node or cloud.

Node A: Central (Aggregator)

- ✓ **BLE Role:** Client (Scanner/Initiator)
- ✓ **FL Role:** Aggregator & Timekeeper
- ✓ **Dataset:** { 'O' Gestures, Noise }
- ✓ **Task:** Downloads weights, computes Global Average, redistributes model.

Node B: Peripheral (Worker)

- ✓ **BLE Role:** Server (Advertiser)
- ✓ **FL Role:** Local Trainer
- ✓ **Dataset:** { 'W' Gestures, Noise }
- ✓ **Task:** Performs local training loops, serves weights via BLE characteristics.

Phase I: On-Device Local Training

Backpropagation on local private data (2 Epochs)

↓

Phase II: Serialization & Handshake

Weights → Linear Buffer | Node B advertises "READY"

↓

Phase III: Aggregation (FedAvg)

Node A fetches weights in chunks (MTU Limit) & Averages

↓

Synchronization

Updated Global Model sent back to Node B

FL Verification

FL Evaluation & Result

Convergence Analysis

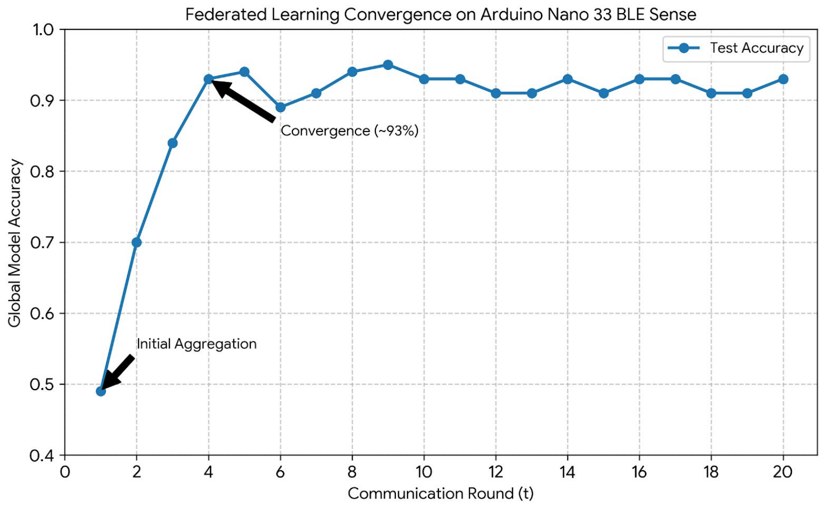
Start: Random initialization (49% Accuracy)

Fast Surge: Reached ~93% within 4 Global Rounds

Stability: Maintained ~93% plateau after convergence

Robustness

Implemented connection supervision timeout. If BLE disconnects (Packet Loss), system auto-heals and retries without hanging.



Demo

Thank You