Minor in AI

TinyML

Edge AI Bootcamp

June 06, 2025

1 Introduction

In the evolving field of embedded intelligence, TinyML has emerged as a groundbreaking technology that brings machine learning capabilities to low-power, resource-constrained edge devices. One compelling application of this paradigm is the use of mobile phones as sensor hubs to develop real-time, intelligent systems. This lecture focused on a practical deployment of TinyML using smartphones and the Edge Impulse platform to build a fall detection system tailored for elderly care. By leveraging mobile sensors and cloud-assisted model training, developers can build cost-effective and deployable solutions without specialized hardware.

TinyML aims to democratize machine learning by enabling inference at the edge, close to where data is generated. The increasing availability of open-source tools and platforms has catalyzed the adoption of TinyML in both research and industry. This practical exercise bridges the gap between theory and implementation by allowing students to collect sensor data directly from mobile phones and observe its transformation into deployable ML models.

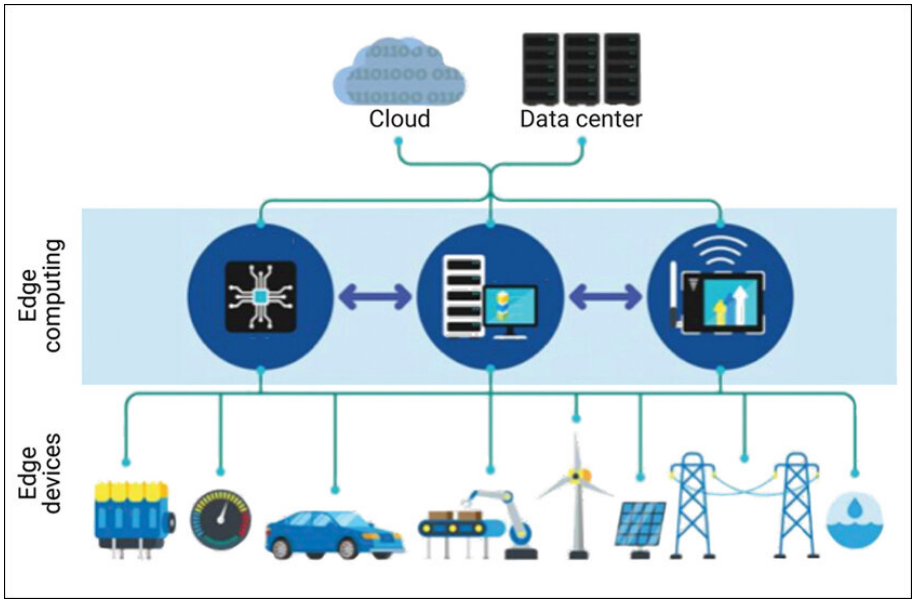


Figure 1: EdgeAI & TinyML

2 Using Mobile Phone Sensors for TinyML

Modern smartphones are equipped with an array of sensors such as:

- **IMU (Inertial Measurement Unit)**: combines accelerometer and gyroscope
- **Magnetometer, Proximity, Light, Barometer, Heart Rate Sensors**
- **Camera and GPS-based Positional Sensors**

These sensors provide an economical and accessible way to collect high-quality data for model training. Instead of relying on separate hardware, developers can leverage the sensors already present in smartphones.

Smartphones eliminate the need for purchasing and integrating external sensors. This makes TinyML prototyping more accessible and faster, especially for beginners. Additionally, smartphone APIs often allow sampling data at various frequencies, which is essential for tasks like human activity recognition or gesture detection.

3 Edge Impulse: An End-to-End Platform

Edge Impulse is an intuitive platform tailored for edge AI applications, offering:

- No-code/low-code workflows for collecting and labeling data.
- Real-time feature extraction, model training, and testing.
- Seamless deployment to devices like Arduino, ESP32, or Jetson Nano.

One of the major strengths of Edge Impulse is its device compatibility and smooth user experience. Students or developers can get started in minutes by creating an account and linking their phones through QR codes. From there, the pipeline includes data acquisition, signal processing, training using pre-built or custom models, and evaluating performance.

Edge Impulse supports a wide range of input formats — including CSV, JSON, or even streaming from third-party APIs. The models generated can be exported as C++ libraries, TensorFlow Lite models, or even WebAssembly for deployment in browsers.

4 Case Study: Fall Detection System

4.1 Objective

Detect and classify motion events as either "safe" (normal use) or "fall" (abrupt motion typical of falling).

Fall detection systems are particularly important in elderly care, where unnoticed falls can lead to serious injuries. By using embedded ML, such systems can be deployed locally on wearable devices, ensuring both privacy and responsiveness.

4.2 Data Collection Process

- Users connect their phones to Edge Impulse Studio by scanning a QR code.
- Activities are manually labeled as "safe" or "fall". Each label needs at least 20 high-quality samples.
- Data is sampled in 10-second windows, but this is configurable.
- Positional sensor data (pitch, roll, gyroscope) provides refined motion features.

Effective fall simulation was achieved by safely mimicking fall scenarios — such as dropping phones onto mattresses or enacting realistic movement patterns using volunteers.

4.3 Data Quality Considerations

- Inconsistent or noisy data is removed.
- Users are advised to stick to one sensor type (e.g., position sensor) to maintain signal consistency.
- Sampling frequency and window duration are tuned based on application demands.

Labeling quality is paramount. Mislabelled samples introduce noise and may severely degrade model performance. Therefore, care was taken to review and clean datasets before training.

5 Model Training and Evaluation

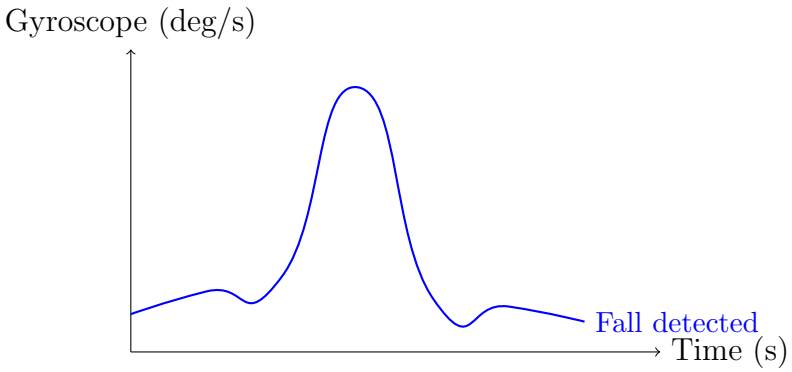
Once data is collected:

- A standard 80-20 train-test split is applied.
- Manual reassignment may be done to balance the split.
- Data is windowed and segmented based on sampling rate and sensor type.
- Model performance is evaluated using metrics like accuracy, recall, and confusion matrix.

Window size and sampling frequency directly influence model responsiveness and memory usage. A smaller window can capture transient signals but increases noise. Choosing a balance is essential.

The model was exported as a TensorFlow Lite format and tested on a mobile device and ESP32 board. Offline inference capability was verified successfully — demonstrating the potential for real-world deployment in environments lacking internet access.

6 Visualization: Sensor Readings During a Fall



7 Troubleshooting and Tips

Common issues faced by users included:

- Sensor failures or non-responsiveness: switch browsers or use another device.
- Unstable wireless connections: reconnect or refresh Edge Impulse Studio.
- Inaccurate fall simulation: safely simulate using mattresses or human-like gestures.
- Mislabelled data: double-check labels for all samples and remove ambiguous ones.

Participants were advised to start with fewer samples and iterate progressively. As models improved, more varied and complex samples were introduced to improve generalization. Frequent validation after each training iteration helped track performance shifts and catch issues early.

Key Takeaways

1. Smartphones are powerful tools for TinyML prototyping due to their integrated sensors.
2. Edge Impulse simplifies the end-to-end workflow from data collection to model deployment.
3. Accurate data labeling and consistent sensor use are crucial for successful models.
4. Fall detection is a practical and socially impactful application of TinyML.
5. Offline inference makes these solutions suitable for real-world IoT deployments.
6. Hands-on tools like Edge Impulse reduce the barrier to entry for ML on the edge.
7. Design choices in window size, sampling frequency, and feature type greatly affect model outcome.