EDGE AI & TINYML

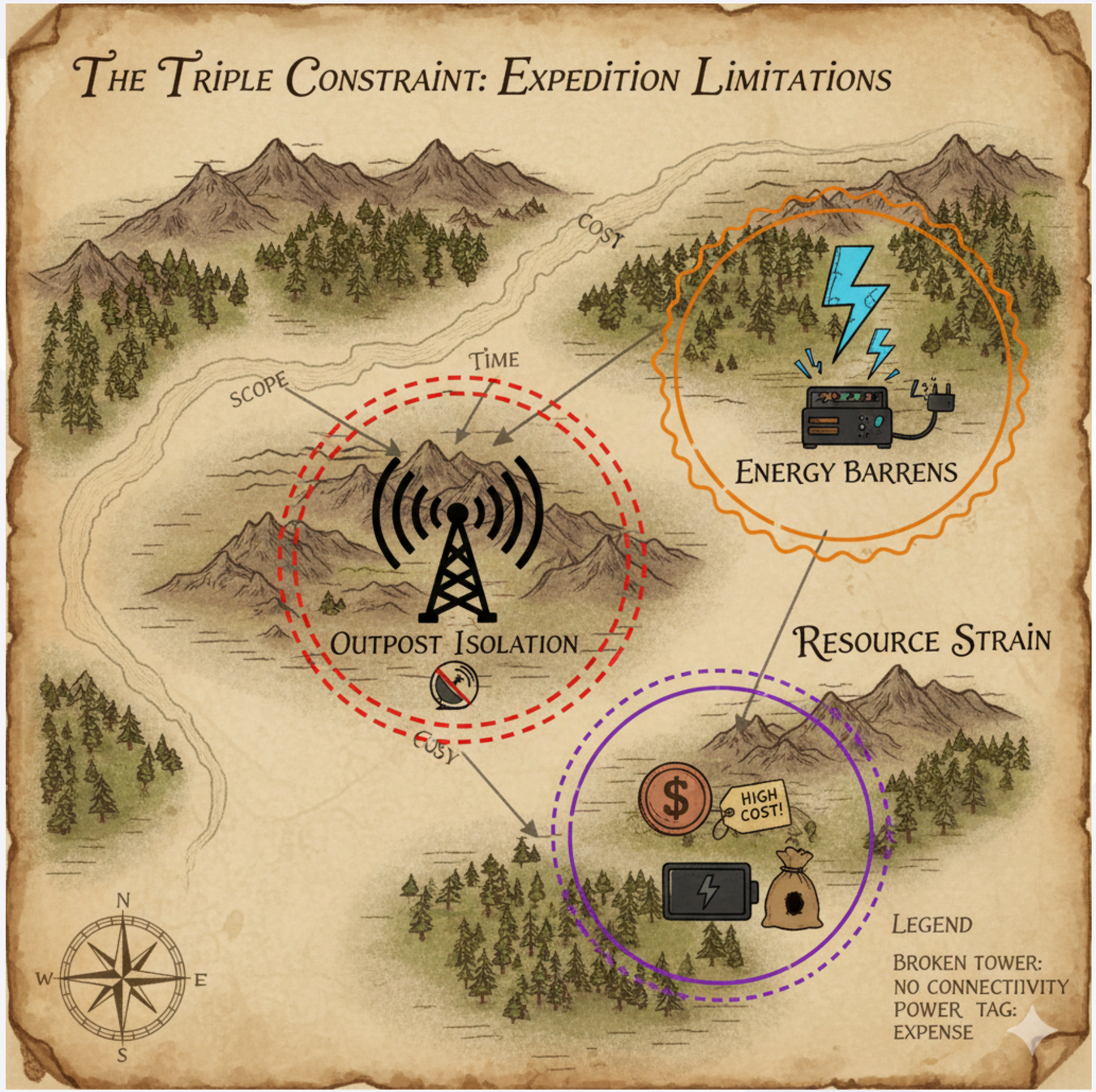
Unlocking Potential in Resource-Constrained Indian Settings

THE "RESOURCE-CONSTRAINED" REALITY

Understanding India's unique challenges in connectivity, power & cost

***Key Challenges in the Indian Setting*:**

- Intermittent or Poor Connectivity (The Latency Challenge):
  - Only about 50% of the population has access to the internet, and coverage is often patchy and slow in rural/remote areas.
  - Reliance on cloud processing (sending data far away) leads to unacceptable delays (high latency) for critical applications like healthcare or agriculture.
- Unreliable Power Supply (The Energy Challenge):
  - Many villages and small towns face frequent power cuts or rely on off-grid solutions.
  - High-performance cloud devices or powerful edge gateways are impractical due to their heavy power demands.



- High Cost of Deployment
  - (The Affordability Challenge)
  - The target demographic often has limited disposable income, making expensive sensor networks, high-end compute hardware, or large data plans unaffordable.
  - Solutions must utilize low cost, widely available hardware (microcontrollers <$5)

- Data Privacy and Security:
  - Sending sensitive user data to foreign cloud servers raises significant trust and security concerns.

THE PROBLEM WITH CLOUD-ONLY AI

Why sending all data to the cloud fails in the Indian context ??

Component	Description	Requirement
Data Collection	Sensors/devices collect raw data (images, audio, temperature).	Requires initial device hardware.
Transmission	All raw data is immediately sent to a remote cloud server.	Requires constant, high-speed internet.
Processing	Powerful servers run complex AI models (training and inference).	Requires massive, reliable power and expensive infrastructure.
Response	The processed result is sent back to the device.	Requires low-latency network connection.

Why the Cloud-Only Model Fails in India ??

⚡ **High Latency for Decisions:**

- The round-trip delay (device → cloud → device) is too slow for time-sensitive applications
- *Result*: Critical failure to act quickly.

📶 **Reliance on Flawless Connectivity:**

- The model breaks entirely when internet connection is intermittent or unavailable in areas.
- *Result*: Zero functionality during network outages.

💰 **Prohibitive Cost of Data:**

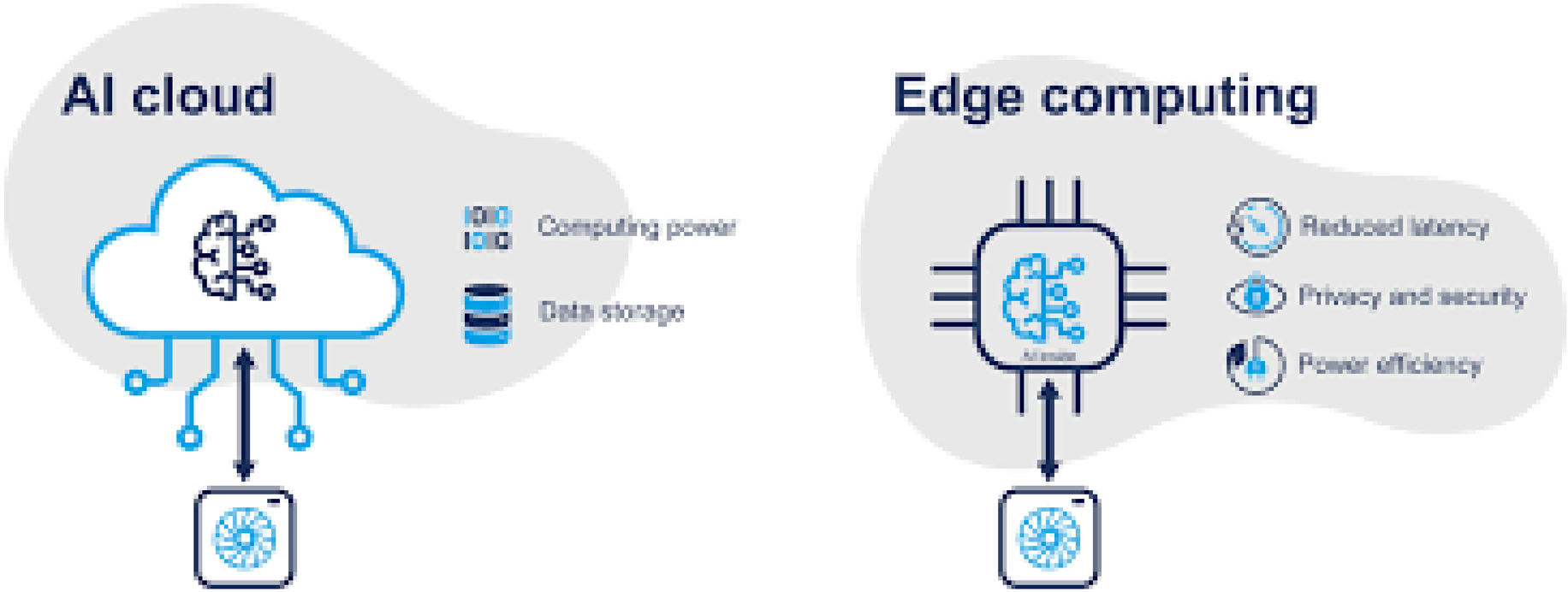
- Sending large volumes of raw data (especially video/images) over mobile networks is expensive for users and implementers.
- *Result*: Unsustainable operating cost for low-budget projects.

🔒 **Security and Privacy Risk:**

- The constant transfer and storage of personal data on remote servers increases the surface area for attack and raises data sovereignty concerns.
- *Result*: Reduced user trust and regulatory challenges.

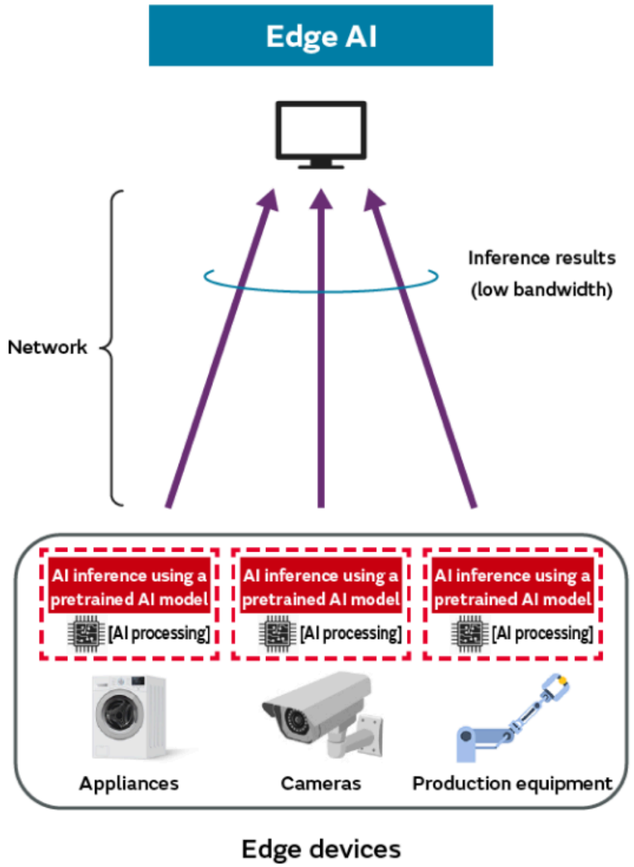
DEFINING EDGE AI

- Edge AI refers to the execution of AI algorithms specifically the inference stage (making predictions or classifications) — directly on a local device, such as a sensor, gateway, or microcontroller.
- It is about processing data where it is generated (at the "edge" of the network), instead of sending it to a central cloud server.



How Edge AI Transforms the Workflow ??

1. ***Local Processing:*** Data is captured by the sensor and processed by the AI model immediately on the device itself.
2. ***Smart Filtering:*** Only the results or necessary alerts are sent upstream to the cloud, not the raw data.
   (e.g., "Pest detected," "Water pump is failing")
3. ***Cloud Retains Training:*** The training of the large, complex AI model still occurs in the powerful cloud environment.
4. ***Model Deployment:*** The optimized, small AI model ("brain") is then pushed down to the low-power edge device.



The Key Edge AI Benefits for India

🏃 **Zero Latency:** Decisions are made in milliseconds on the device, critical for real-time safety and operational efficiency.

📶 **Functionality Offline:** The device continues to operate and make intelligent decisions even when there is zero network connectivity.

💰 **Reduced Bandwidth Costs:** By sending only compressed results, data usage and associated costs are drastically lowered.

🔒 **Enhanced Privacy:** Sensitive data (raw video, audio) never leaves the local device, addressing privacy concerns directly.

PROCESS OF CLASSICAL ML

01.	Data Input:	AI systems learn from large datasets.
02.	Pattern Recognition:	Identifies trends and behaviors.
03.	Decision Making	Applies learning to make predictions.
04.	Self Improvement	Gets better with more data.

TINYML (TINY MACHINE LEARNING)

- TinyML is a subset of Machine Learning focused on deploying highly efficient, compressed AI models (specifically Deep Learning) onto extremely low-power, resource-constrained microcontrollers (MCUs).
- These MCUs typically have less than 1MB of Flash memory and can run on milliwatts (mW) of power the kind of power drawn by a small LED light.
- ***TinyML achieves this miniaturization through several advanced techniques:***
  - *Model Quantization:* Reducing the precision of the numbers used in the model (from 32-bit floating point to 8-bit integer) to shrink model size and accelerate computation.
  - *Model Pruning*: Removing redundant or less important connections (weights) within the neural network to reduce computational load.
  - *Highly Optimized Frameworks:* Using dedicated, minimalist software libraries (e.g., TensorFlow Lite Micro, Edge Impulse) that are specifically designed to run on bare-metal MCUs.

WHY TINYML IS CRUCIAL FOR THE INDIAN CONTEXT ?

Feature	TinyML Advantage	Impact for India
Power Consumption	Operates on mW (battery life measured in months/years).	Essential for off-grid and solar-powered deployments.
Cost	Runs on inexpensive, ubiquitous MCUs (<$5 Chips)	Makes large-scale affordable deployment feasible for farmers, small businesses, and rural clinics.
Form Factor	Small physical size allows deployment in wearable devices and existing infrastructure.	Enables AI integration into low-cost devices where space is restricted (e.g., water meters, crop sensors).
Computational Needs	Focuses on simple inference tasks (classification, anomaly detection).	Perfectly suited for specific, localized problems common in Indian industry and agriculture.

HOW EDGE + TINYML IS A PERFECT FIT FOR INDIA

The "4-Lows" Advantage: Low Power, Low Latency, Low Cost, Low Connectivity

Challenges	The Edge AI + TinyML Solution	The Core Benefit
Unreliable Power	Low Power Consumption (mW level operation).	Enables long battery life for devices (months/years) and viability in off-grid locations.
Poor Connectivity	Low (or No) Connectivity Reliance.	Devices are functional 24/7 regardless of network uptime, ensuring reliability in remote areas.
High Latency	Low Latency Processing (ms decision-making).	Critical for real-time applications like patient monitoring or automated safety systems.
High Cost of Tech	Low Cost Hardware Requirement (MCUs <$5).	Makes solutions accessible and scalable across low-income demographics and mass deployments.

"MAKE IN INDIA" ECOSYSTEM

A "Make in India" Opportunity: The shift to Edge AI and TinyML is fostering a vibrant domestic ecosystem, moving India from a technology consumer to an innovator.

***Emerging Startups & Innovators*:**

- AI Hardware: Companies developing indigenous RISC-V based processors and low-power AI accelerator chips.
- AgriTech: Startups using on-device computer vision for crop health monitoring and precision spraying.
- HealthTech: Ventures creating affordable, AI-powered diagnostic wearables that work offline.
- Industrial IoT (IIoT): Firms using TinyML for predictive maintenance (e.g., monitoring motor vibrations) in factories.

***Academic & Research Leadership*:**

- Top institutions like IITs (Madras, Delhi, Bombay) and IISc Bangalore are pioneering research in:
- Efficient model compression algorithms.
- Low-power hardware design.
- Novel applications for local societal challenges.

***Government & Policy Support*:**

- *Digital India Mission:* Aims to provide ubiquitous connectivity, creating the backbone for connected edge devices.
- *India Semiconductor Mission (ISM):* Pushing for domestic chip design and manufacturing, which is critical for reducing hardware dependency.
- *MeitY (Ministry of Electronics and IT):* Actively funding research and incubation centers focused on AI and emerging technologies.

CHALLENGES AND ROADBLOCKS

***The Talent & Skill Gap*:**

- There is a significant shortage of engineers who possess expertise in both Machine Learning and Embedded Systems Engineering.
- Most data science programs focus on cloud-based (Python, high-power) AI, not on-device (C/C++, MCU) optimization.

***Hardware Dependency*:**

- India is still heavily reliant on importing advanced microcontrollers, sensors, and AI accelerator chips.
- This creates supply chain vulnerabilities and impacts the final cost and "Make in India" goals.

ROADBLOCKS

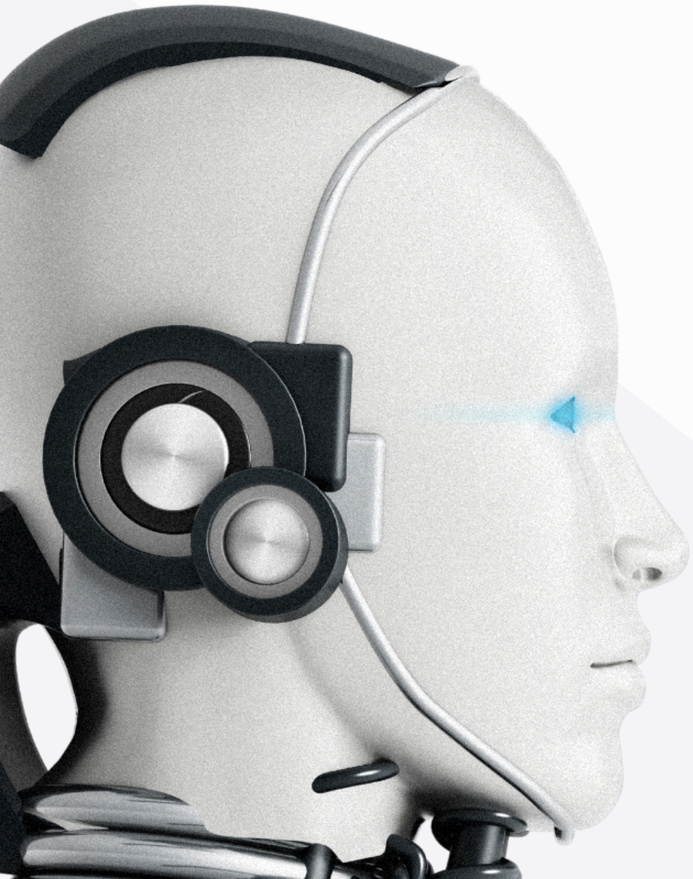
The "Last Mile" Deployment Challenge:

- Deploying and maintaining thousands of hardware devices in remote, harsh environments is logistically complex.
- Managing Over-the-Air (OTA) updates (firmware and model updates) for low-connectivity devices remains a major technical hurdle.

Data Scarcity & Diversity:

- Training robust models requires large, diverse datasets specific to Indian conditions (e.g., local pests, Indian accents, varied road conditions).
- Collecting and annotating this "edge-specific" data is resource-intensive.

TRANSFORMING KEY INDIAN SECTORS



01. Smart Agriculture

02. Smart Infrastructure & Mobility

03. Accessible Healthcare

04. Language & Accessibility

AI is already part of our everyday lives, making tasks easier, faster, and more personalized across different areas.

USE CASES OF TINY ML

1) Automotive Use Case – In-Vehicle Cybersecurity

Why Security Is Needed

- Modern vehicles contain 70+ ECUs using the CAN bus
- CAN bus has no encryption or authentication
- Vulnerable to:
  - DoS attacks – high-priority ID flooding
  - Fuzzy attacks – random data injection
  - RPM spoofing – fake engine speed
  - Gear spoofing – fake gear position

On-Device IDS (LC-IDS) Using TinyML

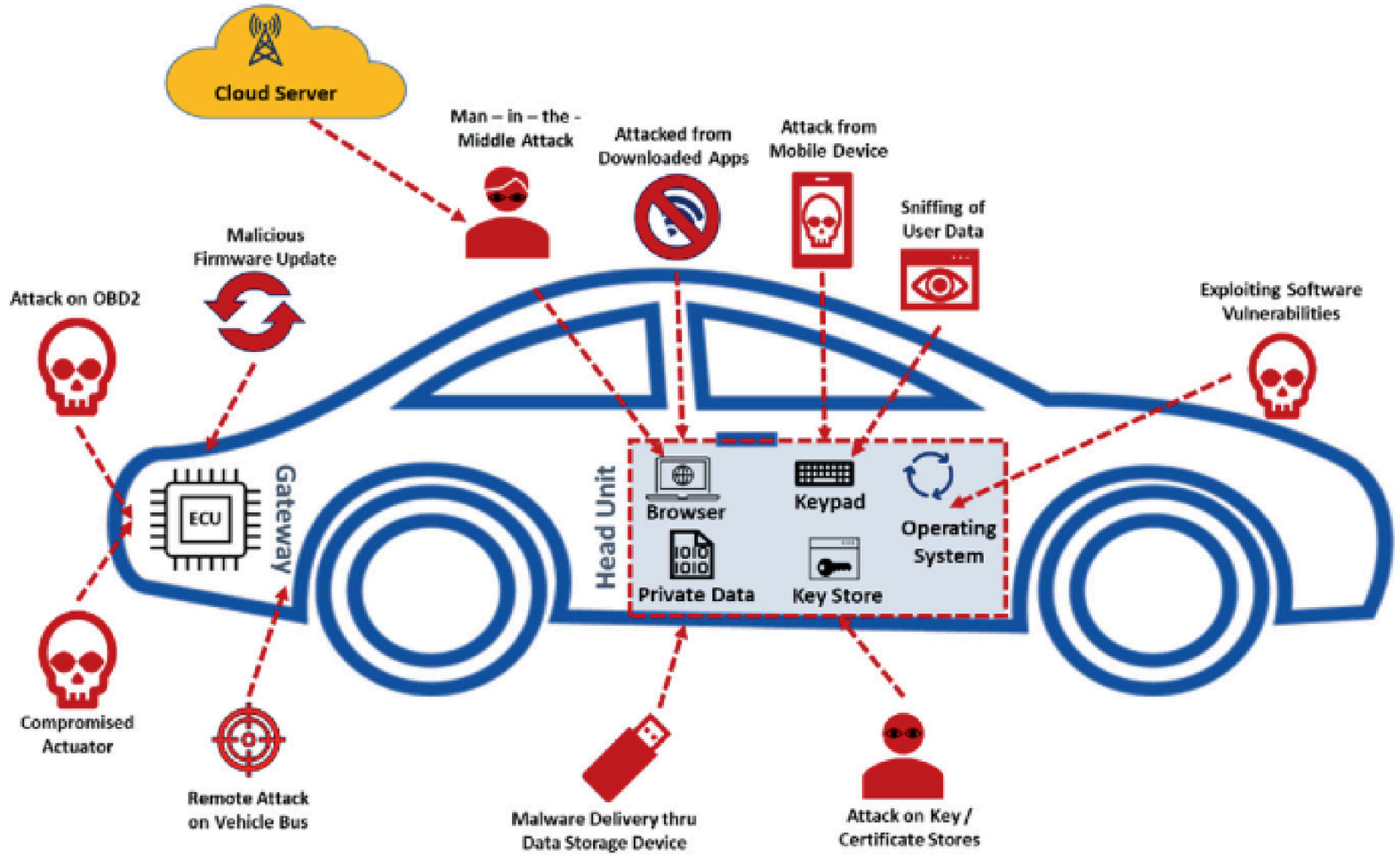
- Runs on low-power MCUs (e.g., nRF52840)
- Detects CAN attacks in real-time

Why TinyML Fits

- ≤ 11 ms inference latency
- 26.44 kB RAM, 20.44 kB Flash
- Works offline
- Local processing = high privacy & security

Practical Automotive Impact

- Prevents malicious CAN messages → avoids unintended braking/acceleration
- Continuous ECU health monitoring
- Enhances V2X and autonomous vehicle safety



USE CASES OF TINY ML

2) Healthcare Use Case – On-Device Anomaly Detection

Why TinyML for Healthcare

- Ideal for remote / rural areas with poor connectivity
- Works on microcontrollers < $5
- Extremely low power → months of battery life

3) Use Case – Medical Equipment Security

Threats

- Spoofed sensor signals
- Manipulated device readings
- Unauthorized command injection
- Attacks on infusion pumps, ventilators, monitors

TinyML-Based IDS Benefits

- Detects abnormal device behavior
- Lightweight, fits low-power medical hardware
- Protects hospital IoT networks
- Enhances safety in critical-care environments

USE CASES OF TINY ML

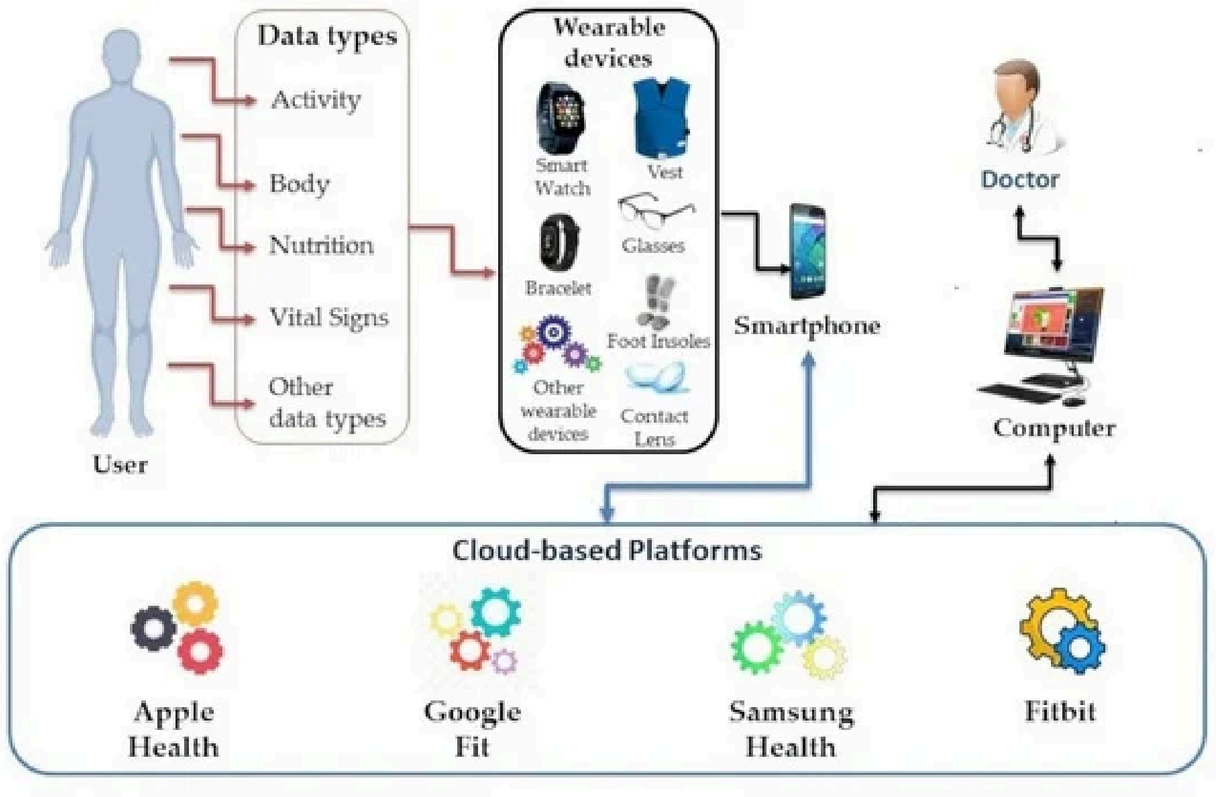
4) Use Case – Wearable Patient Monitoring

TinyML in Wearables

- Heart-rate anomaly detection
- SpO_2 abnormality detection
- On-device arrhythmia analysis
- Fall detection, emergency alerts

Advantages

- Zero latency → instant anomaly detection
- Offline operation → no internet required
- Privacy-preserving → raw data stays on-device
- Long battery life → continuous monitoring

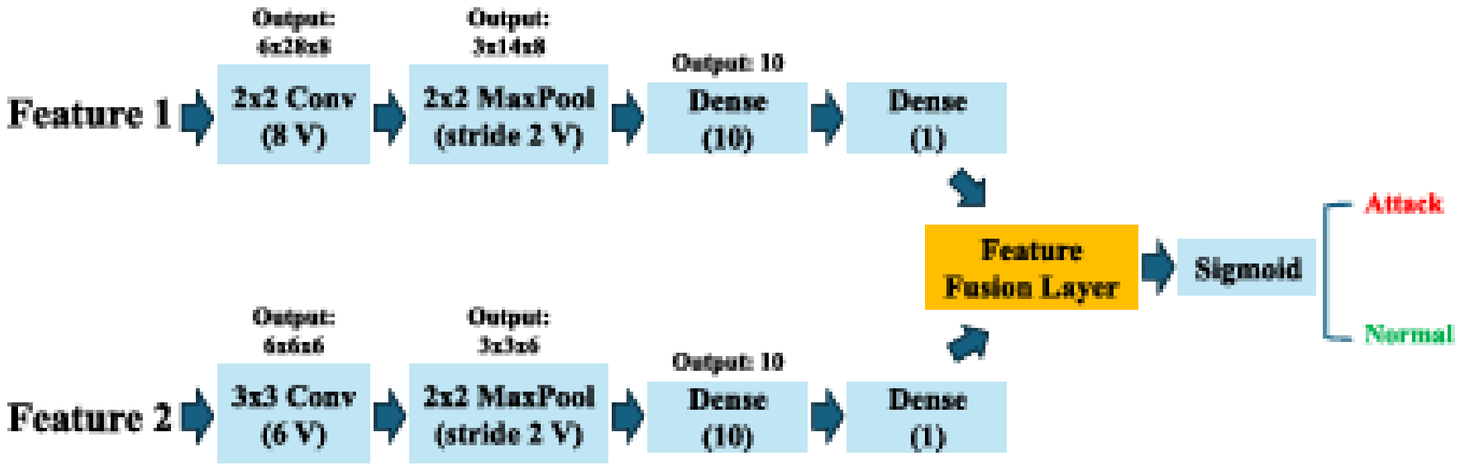


IMPLEMENTATION OF LC-IDS FOR CAN INTRUSION DETECTION

The implementation is based on the TinyML-oriented LC-IDS model proposed by Im & Lee (2025) t
The solution - for embedded systems and deployed on a resource-constrained microcontroller.

Dataset Preparation

- Dataset used: Car Hacking Dataset (OBD-II captured CAN frames).
- Types of attacks: DoS, Fuzzy, RPM spoofing, Gear spoofing.
- Data is organized into sliding window sequences for each sample.
- For each training input:
  - Feature1 — CAN ID sequence (7 IDs) → converted into a binary image of size 7 × 29 × 1
  - Feature2 — Data field of the last frame → converted into 8 × 8 × 1 binary image



“https://ieeexplore.ieee.org/document/10706824"

Model Architecture

The LC-IDS model contains two separate CNN pipelines:

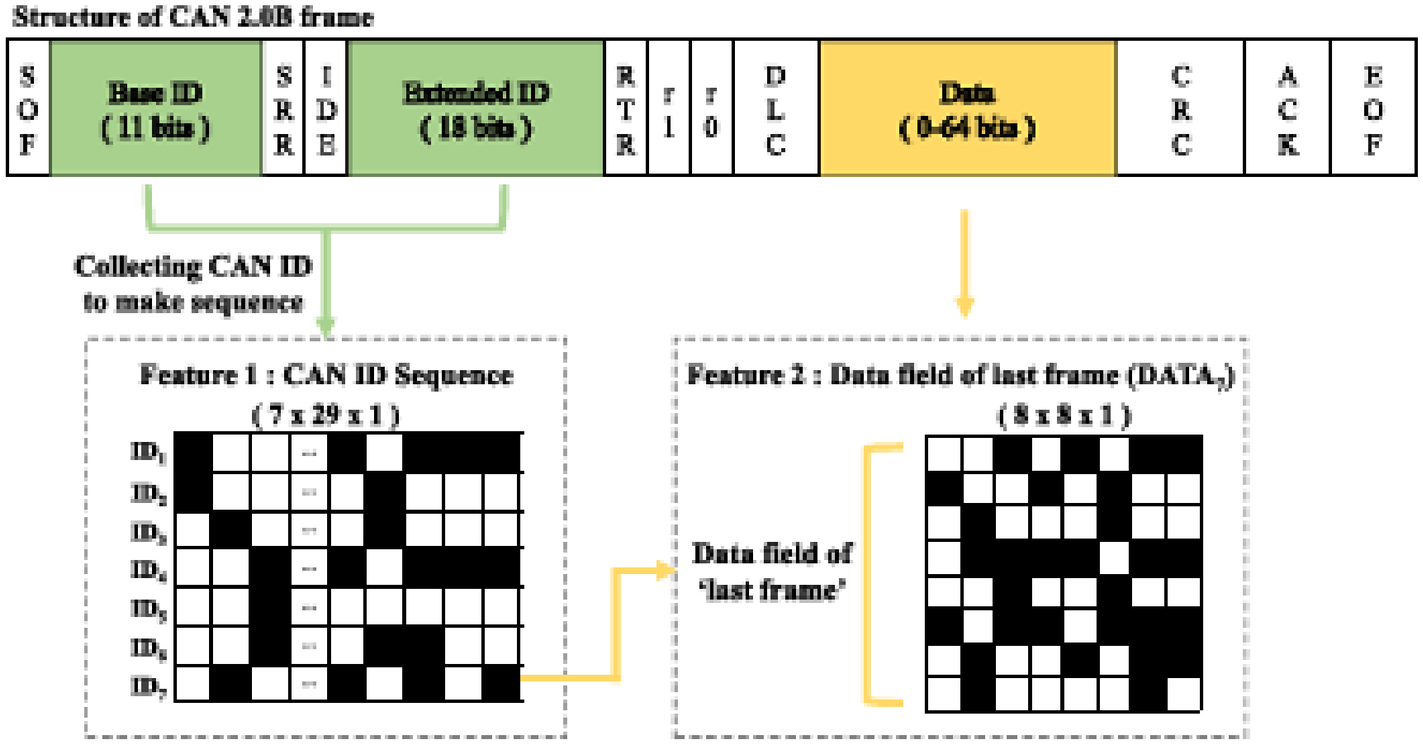
1. CNN branch for CAN ID sequence
2. CNN branch for data field of last frame

Outputs of both branches are flattened and concatenated into fully connected layers.

The model is defined as: $y = f(X_1, X_2, \theta)$

Where:

- X_1: ID sequence image
- X_2: Data-field binary image
- y: Normal or Attack
- θ: Trainable parameters



Feature extraction method for low-complexity IDS.

CNN FLOPs:

$$FLOPs_{CNN} = k_h \times k_w \times C_{in} \times C_{out} \times O_h \times O_w$$

Fully Connected Layer FLOPs: $FLOPs_{FC} = U_{in} \times U_{out}$

The LC-IDS model achieved:

- 8.8% of FLOPs of HyDL-IDS
- 3.3% of parameters of HyDL-IDS

Thus making it ideal for resource-constrained hardware.

Final Outcome

- Achieves *near 100% attack detection* accuracy across all categories.
- Works in real-time with no dependency on cloud.
- Suitable for integration into automotive ECUs and other embedded systems.

Deployment on Microcontroller

- Target device: Arduino Nano 33 BLE Sense (nRF52840 MCU)
- 256 kB RAM
- 1 MB Flash
- 64 MHz ARM Cortex-M4

Memory Usage:

- Flash: 20.44 kB
- RAM: 26.44 kB
- Inference latency: 11.08 ms
- Energy consumption: 0.037 mJ

Model converted to:

- TensorFlow Lite →
- TensorFlow Lite Micro →
- C header file for MCU firmware integration.


THANK YOU