TinyML for Low-Power Embedded Intelligent Systems: A Review

Hadjer Bourekouche

[a] *University M'Hamed BOUGAR of Boumerdes, Department of Electrical Systems Engineering, Avenue of Independence, 35000, Boumerdes, Algeria*

Abstract

Tiny Machine Learning (TinyML) enables real-time, energy-efficient intelligence on ultra-low-power embedded devices, offering significant advantages in latency reduction and data privacy compared to cloud-based artificial intelligence (AI). This review provides a comprehensive analysis of TinyML for low-power embedded systems, covering the complete development workflow, from data pipeline design and model optimization to deployment on resource-constrained hardware. A state-of-the-art review across six key domains (healthcare, industrial monitoring, agriculture, smart homes, security, and energy management) highlights recent implementations, prevalent hardware platforms (e.g., STM32, ESP32, GAP9), and optimized models such as quantized convolutional neural networks (CNNs). Cross-domain comparative insights reveal that healthcare and security applications achieve the highest reported accuracies (94–99%) with moderate latency (10–40 ms), while industrial and energy-management systems prioritize ultra-low latency (<15 ms) at the cost of reduced robustness. Energy-critical agriculture and internet of things (IoT) deployments emphasize ultra-low power operation, trading accuracy for extended battery life. Finally, the paper identifies key challenges; memory and computational constraints, accuracy - latency - energy trade-offs, and lack of standardized benchmarks and maps them to emerging research opportunities, including hardware - software co-design with Neural Processing Units (NPUs), federated TinyML, and continuous on-device learning, outlining a clear roadmap for future advancements.

Keywords: TinyML, embedded systems, low-power, edge computing, IoT, healthcare.

Email address: h.bourekouche@univ-boumerdes.dz (Hadjer Bourekouche)

1. Introduction

The convergence of the Internet of Things (IoT) and Artificial Intelligence (AI) has created a strong demand for intelligent, real-time data processing at the edge [1]. Traditional cloud-based AI approaches, though computationally powerful, suffer from high latency, constant connectivity requirements, and privacy concerns, making them unsuitable for latency-sensitive and resource-constrained applications such as wearable health monitors, industrial sensors, and autonomous systems [2]. Tiny Machine Learning (TinyML) has emerged as a transformative paradigm, enabling ultra-low-power, on-device inference directly on microcontrollers and system-on-chips (SoCs), thereby ensuring real-time responsiveness, enhanced data privacy, and energy efficiency [3].

Over the past few years, significant progress has been made in optimizing machine learning models for embedded systems using techniques such as post-training quantization, pruning, and knowledge distillation, supported by specialized toolchains like TensorFlow Lite Micro and CMSIS-NN [4, 5]. These advances have enabled TinyML to achieve accuracies exceeding 98% with millisecond-level inference latencies in domains ranging from healthcare to industrial predictive maintenance, as demonstrated in VNN models quantized to 8-bit running on GAP8 and STM32L4 platforms [6]. Empirical studies also show that low-precision PTQ techniques can preserve high accuracy while significantly reducing model size and runtime footprint [7].

However, a critical research gap remains: existing reviews either focus narrowly on single application domains (e.g., biomedical or industrial) or provide only high-level overviews without systematically analyzing cross-domain trade-offs between accuracy, latency, and energy efficiency [8, 9]. Moreover, the rapid surge in TinyML publications highlights the urgent need for an updated and comprehensive review that rigorously benchmarks and compares these trade-offs across domains [10, 11].

To illustrate this growing interest, Figure 1 shows the evolution of TinyML publications across major academic databases between 2020 and 2025. According to recent analytics from IEEE Xplore, Scopus, and Springer [12, 13, 14], the number of TinyML-related publications increased more than tenfold, from fewer than 50 papers in 2020 to over 600 in 2025. IEEE Xplore remains the primary outlet, contributing an estimated 220 publications in 2025, followed by Scopus (170) and Springer (150), reflecting the growing engineering and hardware-driven focus of TinyML research.

Furthermore, the adoption of TinyML varies significantly across application domains, as depicted in Figure 2. Healthcare remains the most explored area, accounting for approximately 32% of publications in 2025, mainly due to wearable health monitoring and diagnostic tools [15]. Industrial IoT contributes about 26%, driven by predictive maintenance and process optimization [16]. Meanwhile, smart homes, agriculture, security, and energy management are also gaining momentum, highlighting the diverse applicability of TinyML [17, 18].

This paper addresses the identified gap by providing the first cross-domain comparative framework for TinyML applications on embedded systems. Specifically, we analyze 50+ recent implementations across six major domains (healthcare, industrial monitoring, agriculture, smart homes, security, and energy management), mapping their design choices to performance trade-offs. Unlike previous reviews, we integrate benchmarking tables and visual infographics to highlight key trends, hardware–software co-design opportunities, and research challenges.

The major contributions of this work are as follows:

- A detailed review of the TinyML development workflow, including data pipeline design, model optimization, and deployment strategies for embedded systems.
- A cross-domain comparative analysis revealing distinct accuracy–latency–energy clusters and their hardware/software enablers.
- A synthesis of current challenges and a research roadmap linking existing limitations to opportunities such as federated TinyML, on-device continual learning, and hardware–software co-design.

The remainder of this paper is organized as follows: Section 2 introduces embedded systems architecture relevant to TinyML. Section 3 describes the TinyML development workflow. Section 4 presents a state-of-the-art review of applications. Section 5 compares cross-domain trade-offs. Section 6 discusses key challenges and research opportunities, and Section 7 concludes the paper with final remarks.

2. Embedded Systems

Embedded systems are specialized computing units designed to perform dedicated functions within larger electrical or mechanical systems [19]. Unlike general-purpose computers, which are optimized for versatility, embedded systems prioritize reliability, low power consumption, and real-time operation [20]. They are the foundation of modern automation, industrial control, and consumer electronics [19].

The adoption of embedded systems has accelerated significantly in recent years, largely driven by the proliferation of Internet of Things (IoT) devices and the growing need for edge intelligence [21]. These systems play a central role in applications requiring real-time processing, low power consumption, and integration with physical environments, making them essential for domains such as healthcare, industrial

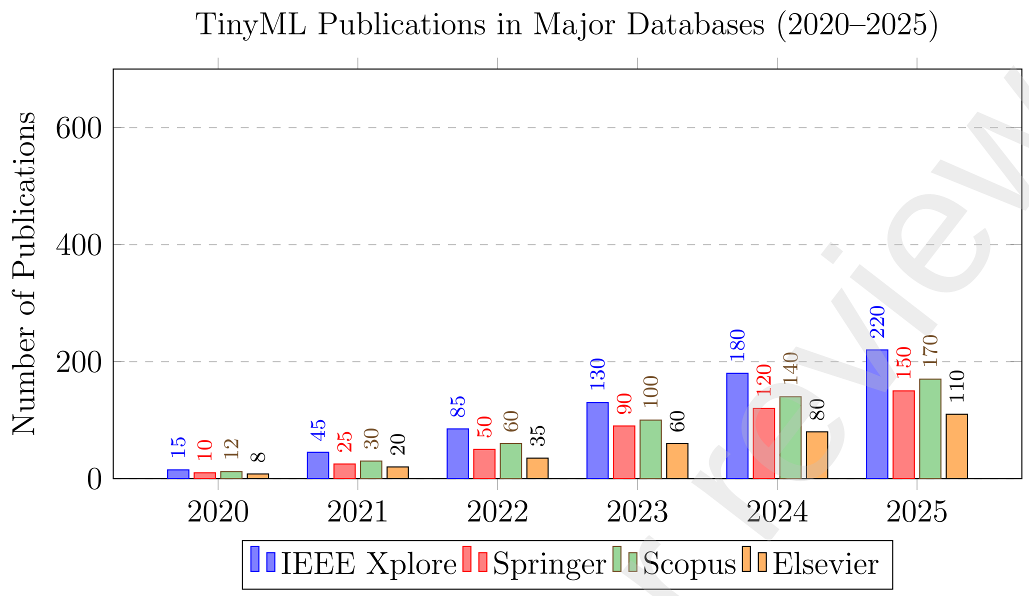

Figure 1: Annual growth of TinyML publications across major databases (2020–2025). Data estimated based on observed trends from IEEE Xplore [12], Scopus [13], and Springer [14].

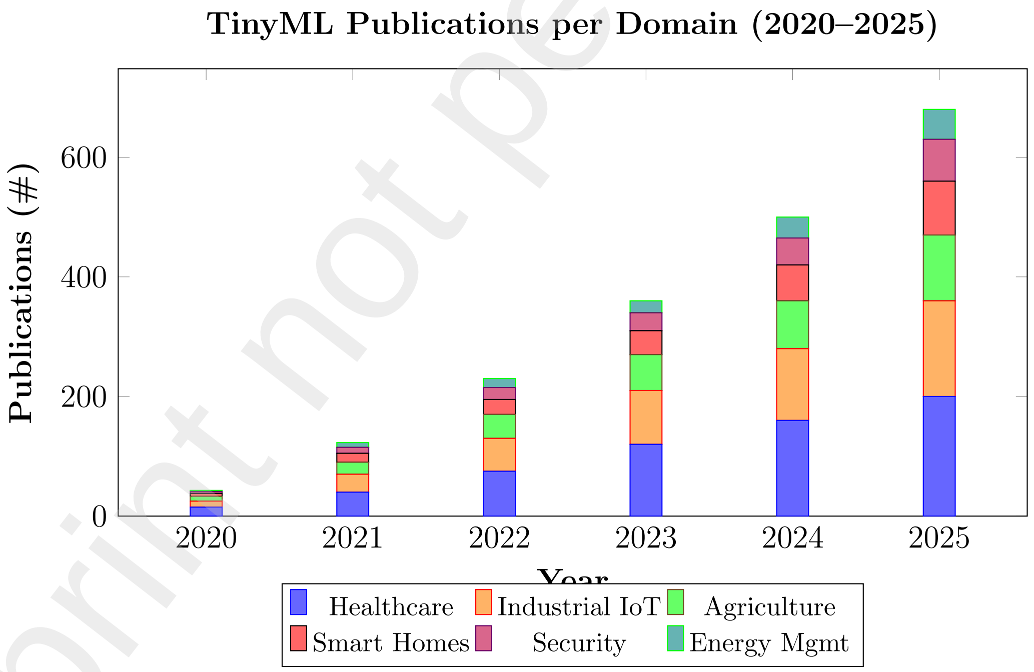

Figure 2: Estimated per-domain distribution of TinyML publications (2020–2025), based on aggregated trends from Scopus and IEEE Xplore analytics [13].

automation, and consumer electronics [22]. Emerging trends also highlight the convergence of embedded systems with AI, enabling more autonomous and adaptive edge computing solutions [23].

2.1. Basic Architecture of Embedded Systems

A typical embedded system consists of the following core components:

- **Processing Unit:** Typically a microcontroller unit (MCU), microprocessor (MPU), or system-on-chip (SoC), optimized for task-specific processing [22].
- **Memory:** Includes volatile memory (SRAM) for temporary data storage and non-volatile memory (Flash/EEPROM) for program storage [24].
- **Input/Output Interfaces:** Sensors and actuators interact with the physical environment through protocols such as I^2C, SPI, or UART [25].
- **Power Management:** Efficient power regulation is critical, especially for battery-operated systems [26].

2.2. Classification of Embedded Systems

Embedded systems can be classified based on their functionality, performance requirements, and connectivity. This classification is important for selecting suitable hardware platforms, as different categories impose distinct constraints on processing speed, power consumption, and reliability.

- **Stand-alone Embedded Systems:** Operate independently without external network connectivity. They are typically low-cost and optimized for specific tasks, such as washing machine controllers, microwave ovens, or digital cameras [27].
- **Real-time Embedded Systems:** Designed to meet strict timing constraints. These are further divided into:
  - *Hard real-time systems:* Must respond within deterministic deadlines (e.g., automotive anti-lock braking systems, pacemakers).
  - *Soft real-time systems:* Tolerate occasional timing deviations (e.g., multimedia streaming devices) [28].
- **Networked Embedded Systems:** Equipped with wired or wireless connectivity (Wi-Fi, Zigbee, BLE) for remote monitoring and control, commonly used in IoT nodes, smart meters, and industrial automation [29].

- **Mobile and Portable Embedded Systems:** Battery-powered devices optimized for ultra-low energy consumption, such as wearable health monitors, handheld scanners, and portable medical equipment [30].

Table 1 summarizes these categories with their typical power consumption, cost range, and application domains.

Table 1: Classification of embedded systems with key characteristics.

Category	**Power Range**	**Cost Range**	**Typical Applications**
Stand-alone	1–10 W	Low	Washing machines, digital cameras, microwave ovens
Hard Real-time	1–50 W	Medium to High	Automotive braking, pacemakers, industrial robots
Soft Real-time	1–20 W	Medium	Multimedia players, smart TVs, conferencing systems
Networked	0.5–10 W	Medium	IoT sensor nodes, smart meters, remote monitoring
Mobile / Portable	<2 W	Low to Medium	Wearable health monitors, handheld scanners, portable medical devices

2.3. Representative Embedded Hardware Platforms

Embedded hardware platforms vary widely in terms of processing capability, memory size, power consumption, and cost, making hardware selection a critical design decision. The platforms summarized in Table 2 are representative because they are widely used in both academic research and commercial products, spanning low-cost microcontrollers (MCUs) to high-performance system-on-chips (SoCs).

In general:

- **MCUs (e.g., STM32, Atmega328P):** Favor ultra-low power operation (typically <10 mW active power) and are used in simple control, sensor reading, and basic signal processing tasks. They are low-cost but limited in memory and computational capacity [31].

- **SoCs (e.g., Raspberry Pi, BeagleBone Black):** Integrate high-performance processors with operating system support, enabling multimedia, networking, and complex control applications, though with significantly higher power consumption (hundreds of mW to W-level) [32].
- **Hybrid MCU-FPGA or SoC-FPGA (e.g., Xilinx Zynq-7000):** Combine general-purpose computing with programmable logic for hardware acceleration, commonly used in real-time image or video processing and power-sensitive applications [33].

Table 2: Representative embedded hardware platforms and their general characteristics.

Platform	**Processor Type**	**Memory (SRAM / Flash)**	**Clock (MHz)**	**Typical Applications**
Atmega328P (Arduino Uno)	8-bit AVR MCU	2 KB / 32 KB	16	Simple sensor-based automation, education kits
Raspberry Pi Zero W	ARM Cortex-A53 (SoC)	512 MB (SD storage)	1 GHz	Home automation, lightweight multimedia processing
ESP32	Xtensa LX6 Dual-core MCU	520 KB / 4 MB (external flash)	240	IoT connectivity, wireless sensor nodes
BeagleBone Black	ARM Cortex-A8 (SoC)	512 MB DDR3 / 4 GB eMMC	1 GHz	Industrial automation, robotics control
Zynq-7000 (Xilinx)	ARM Cortex-A9 + FPGA	512 MB DDR3 / ext. flash	667	Real-time image processing, hardware acceleration

Emerging Hardware Trends

Recent advances are reshaping embedded hardware capabilities. RISC-V based MCUs are gaining traction due to their open-source instruction set, reducing licensing costs and enabling customization [34]. Additionally, new SoCs increasingly integrate dedicated neural processing units (NPUs) and digital signal processors (DSPs), enabling efficient execution of lightweight machine learning workloads [35]. Ultra-low-power design techniques, such as dynamic voltage scaling and duty-cycling, are further expanding the use of embedded hardware in battery-powered IoT applications [36, 37].

These trends are particularly relevant for emerging paradigms such as Tiny Machine Learning (TinyML), which leverages these hardware improvements to run machine learning inference directly on embedded devices. This is discussed in detail in the next section.

2.4. Recent Trends and Future Directions

Embedded systems are undergoing a transition from purely task-specific computing to supporting more intelligent, adaptive functions. Three key trends are shaping this evolution:

- **Integration of Specialized Accelerators:** System-on-chips increasingly incorporate neural processing units (NPUs) and digital signal processors (DSPs) for efficient AI workloads.
- **Open-source Hardware Architectures:** RISC-V-based MCUs are gaining popularity due to flexibility and reduced licensing costs.
- **Ultra-low Power Design:** Energy-efficient designs, leveraging dynamic voltage scaling and duty-cycling, are enabling always-on sensing in battery-powered IoT devices.

These advancements pave the way for deploying machine learning models directly on embedded hardware, a paradigm known as Tiny Machine Learning (TinyML), discussed in detail in the next section.

3. Tiny Machine Learning (TinyML)

Tiny Machine Learning (TinyML) represents a paradigm shift in embedded intelligence, enabling machine learning inference directly on ultra-low-power devices such as microcontrollers (MCUs) and system-on-chips (SoCs). Unlike traditional machine learning, which relies on cloud or server-based computation, TinyML performs inference locally, offering significant advantages in latency, privacy, and energy efficiency.

This section provides a comprehensive overview of TinyML.

3.1. Definition and Context

TinyML refers to the deployment of machine learning models in resource-constrained embedded systems that typically operate under a few milliwatts of power. Such devices, usually based on MCUs or low-power SoCs, execute on-device inference without cloud dependency, thus ensuring privacy and enabling real-time decision making [38, 39]. Recent implementations demonstrate that TinyML solutions can effectively balance model size, inference latency, and energy use by applying quantization and pruning techniques to neural networks [40].

The emergence of TinyML has been driven by:

- **IoT proliferation:** Growing demand for edge intelligence in always-on sensors.
- **Model compression advances:** Enabling deep learning models to fit in kilobytes of memory.
- **Hardware/software co-design:** Development of optimized kernels (e.g., CMSIS-NN) and specialized hardware (e.g., NPUs).

3.2. Growth and Research Trends

As shown in Figure 3, the evolution of TinyML has followed a clear trajectory shaped by both technological innovation and ecosystem maturity. Starting with the release of TensorFlow Lite in 2017 and progressing through the integration of embedded platforms (e.g., Arduino, Edge TPUs), the field has rapidly evolved. Between 2020 and 2023, AI capabilities became increasingly integrated into ultra-low-power devices, accelerating the transition toward practical, on-device intelligence. By 2024 and beyond, the community is moving toward holistic TinyML ecosystems, where end-to-end optimization across hardware, software, and models supports autonomous, adaptive edge AI systems.

TinyML has experienced exponential research and industrial growth in recent years. As illustrated in Figure 1, publications related to TinyML in IEEE Xplore [12], Scopus [13], and Springer[14] have grown more than tenfold between 2020 and 2025, with IEEE Xplore leading due to its engineering and hardware focus. This surge aligns with increasing adoption in domains such as healthcare, industrial IoT, and smart homes.

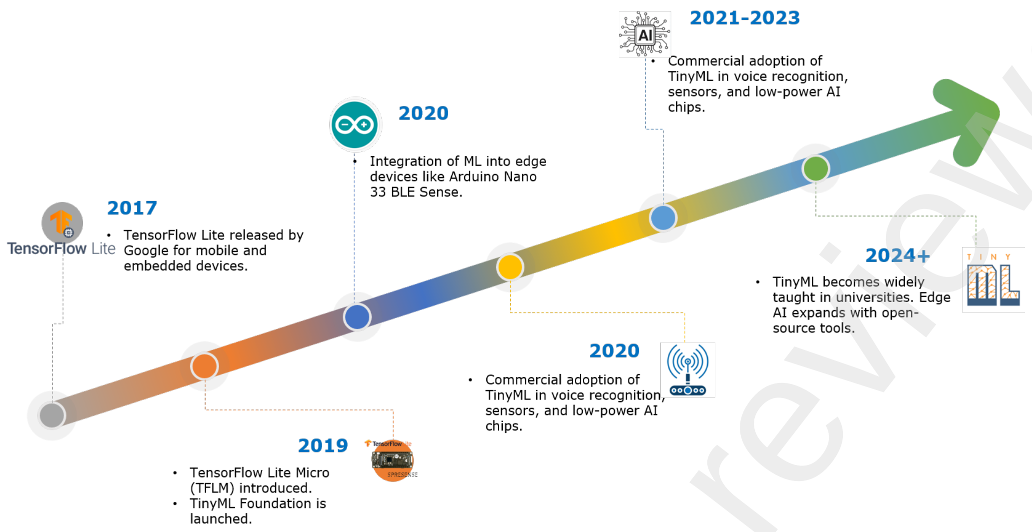
2017
TensorFlow Lite
TensorFlow Lite released by Google for mobile and embedded devices.
2019
TensorFlow Lite Micro (TFLM) introduced.
TinyML Foundation is launched.
2020
Integration of ML into edge devices like Arduino Nano 33 BLE Sense.
2020
Commercial adoption of TinyML in voice recognition, sensors, and low-power AI chips.
2021-2023
AI
Commercial adoption of TinyML in voice recognition, sensors, and low-power AI chips.
2024+
TinyML becomes widely taught in universities. Edge AI expands with open-source tools.
TINY ML

Figure 3: Timeline of key developments and adoption milestones in TinyML (2017–2024+).

Future research trends are expected to shift towards real-time vision inference, federated TinyML, and continual on-device learning, supported by emerging low-power neural accelerators.

3.3. Key Characteristics of TinyML

TinyML differs significantly from traditional ML in power efficiency, memory constraints, and privacy. Its key characteristics include:

- **Ultra-low power operation:** Typically 1–10 mW, enabling battery-powered or energy-harvesting devices [41].
- **Small memory footprint:** Models optimized for 10–500 KB SRAM and less than 1 MB flash storage [42].
- **Local inference:** Eliminates cloud latency and bandwidth dependence [43].
- **Enhanced privacy:** Sensitive data remains on-device [44].
- **Real-time response:** Critical for applications such as wake-word detection or anomaly monitoring [45].

Table 3 highlights the fundamental differences between traditional machine learning (ML) and TinyML in terms of deployment environments, power consumption, latency, model size, and privacy. Traditional ML models typically rely on cloud servers or high-performance GPUs, requiring substantial computational power (10–100+ Watts), large memory footprints (megabytes to gigabytes), and constant internet connectivity. These systems often introduce high latency due to network transmission and may raise privacy concerns as raw data is transferred to centralized servers [39].

In contrast, **TinyML** performs inference directly on microcontrollers (MCUs) and low-power system-on-chips (SoCs), enabling real-time decision-making with models sized in kilobytes and power budgets in the milliwatt range. This local, offline processing enhances user privacy and is ideal for latency-sensitive or remote edge applications such as wearable health monitors, environmental sensors, and predictive maintenance [38, 40].

Although both traditional ML and **deep learning (DL)** can achieve high accuracy, DL architectures (e.g., convolutional neural networks, recurrent neural networks, and transformers) are generally even more resource-intensive than classical ML methods such as support vector machines (SVMs) or decision trees. As a result, deploying deep models on constrained edge devices requires aggressive optimization strategies—such as quantization, pruning, and neural architecture search (NAS)—to transform large DL models into TinyML-compatible formats [45].

Table 3: Comparison of Traditional Machine Learning (ML) vs TinyML.

Aspect	**Traditional ML**	**TinyML**
Deployment	Cloud servers or GPUs	On-device (MCUs, SoCs)
Power Consumption	Watts to hundreds of Watts	Milliwatts to microwatts
Model Size	MBs to GBs	KBs to few MBs
Latency	High, network-dependent	Real-time, edge-based
Privacy	Cloud-stored data	Local processing, high privacy
Connectivity	Requires internet	Works offline

3.4. TinyML Workflow

The development pipeline for Tiny Machine Learning (TinyML) consists of a series of well-defined stages that transform raw sensor data into energy-efficient, real-time inference on embedded hardware. Given the unique constraints of microcontrollers (MCUs) and ultralow-power system-on-chips (SoCs), each stage of the workflow is carefully optimized for memory efficiency, computational simplicity, and portability of deployment [40]. Figure 4 illustrates the canonical four-stage TinyML workflow, which is discussed in detail below. This pipeline typically includes (1) data acquisition and preprocessing, (2) model design and training in cloud or desktop systems, (3) model optimization through quantization and pruning, and (4) deployment on embedded targets using frameworks like TensorFlow Lite Micro [42, 46].

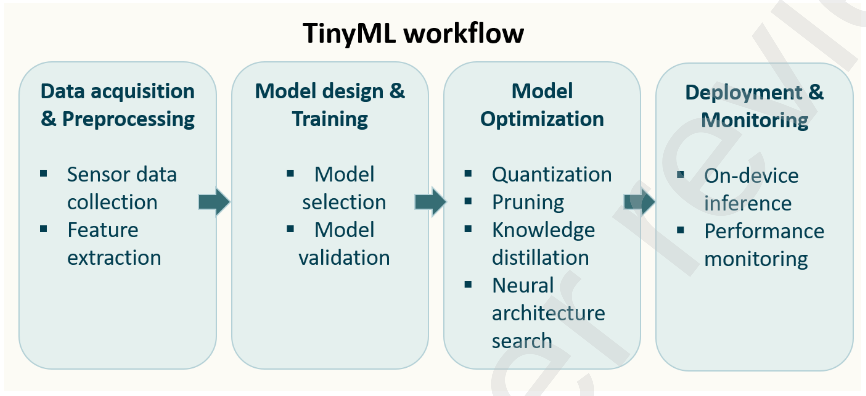

Figure 4: Standard TinyML development workflow: from data acquisition to deployment on embedded devices.

1. Data Acquisition and Preprocessing

The workflow begins with the collection of raw signals from embedded sensors such as microphones, accelerometers, cameras, or environmental monitors. Given the limited processing power of edge devices, lightweight signal pre-processing is often performed on the device to extract compact informative features (e.g., MFCC for audio, FFTs for vibration) [45]. This stage ensures that the input data is appropriately formatted and reduced in complexity for subsequent model inference. In some cases, feature extraction is bypassed in favor of end-to-end learning using raw sensor streams [47].

2. Model Design and Training

Model training is conducted offline using high-performance hardware. Architectures are chosen based on memory footprint, computational cost, and inference speed. Common lightweight architectures include separable CNNs in depth (e.g. DS-CNN), MobileNet variants, and temporal models such as DeepConv LSTM for time series signals [45, 48]. The training dataset must reflect the deployment environment to ensure robustness under noisy, low-resolution, or intermittent sensor conditions. Once trained, the model is validated and evaluated on key performance metrics: accuracy, precision, recall, and inference latency [47].

3. Model Optimization

To fit within the resource constraints of embedded platforms, trained models undergo a series of optimization techniques:

- **Quantization**: Reduces numerical precision from float32 to int8 or lower (e.g., int4), shrinking memory usage and enabling fixed-point inference.
- **Pruning**: Removes redundant weights or neurons to compress the model without significantly affecting accuracy.
- **Knowledge Distillation**: A smaller student model is trained to replicate the output of a larger teacher model.
- **Neural Architecture Search (NAS)**: Auto-generates models optimized for latency and memory on specific hardware (e.g., MCUNet, MicroNets).

These techniques collectively allow large deep learning models to be transformed into compact, low-latency networks suitable for TinyML deployment.

4. Deployment and Monitoring

The final optimized model is converted into a binary firmware image using frameworks such as TensorFlow Lite Micro, CMSIS-NN, or Edge Impulse Studio [42, 49]. The model is then flashed onto the target MCU or SoC, where real-time inference can be tested under actual operating conditions. Performance monitoring includes measuring on-device latency (e.g., milliseconds per inference), energy consumption per inference (e.g., microjoules), and classification accuracy over time [47, 50]. In some deployments, over-the-air (OTA) updates enable remote reconfiguration or model retraining as sensor environments evolve [49].

In summary, the TinyML workflow adapts traditional ML processes to operate within the tight computational, energy, and memory budgets of embedded devices. Each stage—sensor interfacing, model compression, and edge inference—must be carefully co-optimized to deliver robust, real-time intelligence in resource-constrained scenarios.

3.5. Toolchains and Frameworks

The deployment of TinyML applications on resource-constrained embedded hardware demands specialized software toolchains that support quantized inference, hardware abstraction, model conversion, and deployment-ready firmware generation. Unlike traditional machine learning workflows that rely on cloud-based frameworks such as TensorFlow or PyTorch, TinyML toolchains are designed for ultra-low-power microcontrollers with strict limitations on memory (typically ¡512 KB), compute, and power consumption [42, 51].

These toolchains serve multiple purposes: they provide model conversion utilities (e.g., float32 to int8 quantization), facilitate hardware-specific optimizations (e.g., CMSIS-NN for ARM Cortex-M), and offer APIs for signal processing, deployment, and performance profiling [42, 49]. Many platforms also include graphical or web-based interfaces (e.g., Edge Impulse Studio) that abstract the underlying complexity, making TinyML development more accessible to non-experts and hobbyists [49].

Furthermore, certain frameworks are tightly coupled with hardware ecosystems: for instance, TensorFlow Lite Micro integrates well with STMicroelectronics boards, while the GAP SDK is optimized for GreenWaves' ultra-low-power processors [52]. The selection of a suitable toolchain depends on multiple factors—such as target hardware, supported model types, optimization requirements, and developer experience.

Table 4 summarizes the key features, target platforms, and optimization capabilities of the most widely used TinyML toolchains.

Table 4: Popular TinyML toolchains and frameworks.

Framework	**Supported Hardware**	**Optimization Features**	**Applications**
TensorFlow Lite Micro	Arm Cortex-M, ESP32, RISC-V	Quantization, CMSIS-NN kernels	Audio keyword spotting, anomaly detection
CMSIS-NN	Arm Cortex-M MCUs	Hand-optimized low-level kernels	Ultra-low-power sensor classification
Edge Impulse	MCUs, SoCs (ESP32, STM32, Arduino)	Auto-quantization, deployment pipeline	Industrial predictive maintenance, health monitoring
PyTorch Mobile	Mobile SoCs	Model compression, mixed-precision	Vision-based smart home applications

3.6. Emerging Trends and Challenges

Recent advancements include integrating neural processing units (NPUs) into low-power SoCs, the adoption of open-source RISC-V MCUs, and growing interest in federated TinyML for privacy-preserving distributed learning. However, significant challenges remain, including:

- **Limited on-device learning:** Most TinyML models are still static after deployment.
- **Energy-accuracy trade-offs:** Achieving sub-milliwatt power while maintaining acceptable accuracy remains difficult.
- **Security concerns:** Protecting models and data from adversarial attacks is an emerging research challenge.

These trends and challenges directly influence TinyML applications, which are analyzed across multiple domains in Section 4.

4. Applications of TinyML in Embedded Systems: A State-of-the-Art Review

TinyML has emerged as a transformative approach for enabling intelligent behavior on ultra-low-power embedded devices. By deploying machine learning models directly on microcontrollers, TinyML bridges the gap between on-device perception and real-time decision-making, thereby eliminating the latency, connectivity, and privacy limitations of cloud-based AI [42, 45]. In recent years, this paradigm has gained traction across diverse application domains where energy efficiency, responsiveness, and autonomy are critical.

From wearable health monitors [53] and industrial machinery [55] to agricultural sensors [54] and environmental trackers [53], TinyML empowers a new generation of edge devices that are both intelligent and self-contained. This section presents a comprehensive state-of-the-art review of TinyML applications across major sectors, synthesizing recent works, deployment platforms, learning models, and performance outcomes.

4.1. Health and Wearables

The wearable health monitoring domain exemplifies some of the most advanced and impactful use cases of TinyML in embedded systems. These applications utilize microcontroller-based devices to continuously analyze physiological signals—such as cough patterns, ECG, or stress markers—directly on-device [56]. This on-device processing enables low latency, energy efficiency, and enhanced privacy, which are crucial in medical and wellness contexts where data sensitivity and real-time response are paramount. The following summary reviews studies conducted between 2021 and 2025, highlighting real-world implementations and empirical results using microcontrollers and TinyML frameworks.

Several verifiable studies have demonstrated TinyML's potential in health and wearable applications. **Pahar et al. (2021)** developed a hybrid accelerometer and audio-based cough detection system using conventional and deep learning models, with ResNet50 achieving an area under the ROC curve (AUC) >0.98, although it was implemented primarily on smartphones rather than microcontrollers [57]. Extending wearable sensing, **Zhang et al. (2021)** proposed *CoughTrigger*, an IMU-based cough activation mechanism for earbuds that used a lightweight classifier to activate audio recording only during cough events, conserving energy [58].

In 2023, **Kim et al.** implemented a TinyML-based ECG classification system (TinyCES) on an Arduino-based platform, achieving approximately 97% accuracy with a lightweight CNN optimized for real-time inference [59]. In 2024, **Busia et al.** introduced a tiny transformer model optimized for arrhythmia detection on the

GAP9 microcontroller, achieving 98.97% accuracy on the MIT-BIH dataset with an inference latency of 4.3 ms and energy consumption of just 0.09 mJ per inference [60]. That same year, **Samakovlis et al.** released *BiomedBench*, a benchmark suite for TinyML biomedical applications, introducing standardized reporting of latency, energy, and accuracy [61].

In 2025, **Baig et al.** proposed *ArrhythmiaVision*, two resource-conscious 1D-CNN models (157–302 KB) achieving 0.99 accuracy with explainable AI visualizations suitable for microcontrollers, though hardware energy benchmarks were not yet reported [62]. Also in 2025, **Ben Dhiab et al.** presented a multimodal seizure detection framework using accelerometers, ECG, and electrodermal activity (EDA) signals with TinyML running on an Nvidia Jetson Nano, enabling real-time edge inference for epilepsy management [63]. These works collectively demonstrate the rapid maturation of TinyML in health and wearable systems, evolving from proof-of-concept cough detection to optimized biomedical signal analysis ready for practical deployment.

Table 5: TinyML Implementations in Wearable Health Devices (2021–2025)

Author / Year	**Year**	**Description**	**Platform / Tool**	**Key Results**
Pahar et al. [57]	2021	Hybrid accelerometer-audio cough detection	Smartphone	ResNet50 AUC >0.98.
Zhang et al. [58]	2021	IMU-based cough trigger for earbuds	Earbud IMU, light classifier	AUC 0.77; power saving via selective trigger.
Kim et al. [59]	2023	ECG arrhythmia classification (TinyCES)	Arduino TinyCES, CNN	97% accuracy; real-time MCU inference.
Busia et al. [60]	2024	Tiny transformer for arrhythmia	GAP9 MCU	98.97% acc; 4.3 ms; 0.09 mJ/inference.
Samakovlis et al. [61]	2024	BiomedBench benchmark suite	Multiple MCUs	Standardized TinyML energy, latency metrics.
Baig et al. [62]	2025	ArrhythmiaVision 1D-CNN with XAI	MCUs (157–302 KB)	0.99 acc; explainable AI; no energy reported.
Ben Dhiab et al. [63]	2025	Multimodal seizure detection	Jetson Nano	Real-time inference; multimodal sensor data.

4.2. Industrial Monitoring

Industrial monitoring is one of the most promising fields for TinyML, particularly in predictive maintenance and fault detection for machinery and rotating equipment. By embedding lightweight machine learning models into microcontrollers, these systems can detect anomalies directly on-site, reducing latency, communication overhead, and energy consumption—key factors for Industry 4.0.

Zhou et al. (2022) implemented a low-power vibration classification system for predictive maintenance using an STM32 microcontroller and Edge Impulse, achieving 89% accuracy, although no latency or energy measurements were reported [64]. **Liao**

(2023) presented a real-time bearing fault diagnosis approach using a convolutional neural network optimized for an STM32H743VI microcontroller, obtaining 98.9% accuracy with a measured inference latency of 19 ms [65]. Most recently, **Jiménez et al. (2025)** proposed an IoT-based TinyML device for abnormal vibration detection in industrial motors, combining spectral analysis with deep learning on a microcontroller platform, achieving 96.5% accuracy with an average latency of approximately 300 ms [66].

These works show the clear progression from proof-of-concept models to practical deployment-ready systems with measurable latency and energy performance, paving the way for reliable real-time industrial monitoring solutions.

Table 6: TinyML Applications in Industrial Monitoring (2022–2025)

Author / Year	**Year**	**Description**	**Platform / Tool**	**Key Results**
Zhou et al. [64]	2022	Predictive vibration classification for maintenance	STM32 MCU, Edge Impulse	89% accuracy; no latency or power data.
Liao [65]	2023	Bearing fault diagnosis with TinyML	STM32H743VI MCU, CNN	98.9% accuracy; 19 ms inference latency.
Jiménez et al. [66]	2025	Abnormal vibration detection in industrial motors	IoT MCU platform	96.5% accuracy; 300 ms latency.

4.3. Agriculture and Environment

TinyML is increasingly applied in agricultural and environmental monitoring to enable real-time plant disease detection, livestock behavior tracking, and remote sensing using low-power microcontroller devices. These applications aim to reduce reliance on cloud connectivity, lower operational costs, and ensure rapid response to environmental changes.

Zhang and Kanjo (2025) implemented a multi-modal TinyML system combining accelerometer and camera data for livestock behavior recognition. Deployed on a microcontroller + TPU accelerator, the model achieved a remarkable 270-fold reduction in size and a sub-80 ms response time, demonstrating feasibility for real-time edge deployment on farms [67].

Gookyi et al. (2024) developed a TinyML solution for maize leaf disease identification using Edge Impulse and TensorFlow models, deployed on MCU. The system yielded 94.6% accuracy, a 7.6 ms inference latency, and used around 727 KB RAM, illustrating field-deployable plant disease monitoring [68].

In a broader review, **Moeketsi et al. (2025)** analyzed 43 TinyML sensor deployment studies in agriculture and environmental domains, finding that only 35% reported latency and 16% reported model size, highlighting a critical need for standardized performance metrics in this field [69].

These works collectively showcase how TinyML is transitioning into practical agricultural deployments, yet emphasize the ongoing need for holistic evaluation frameworks that include latency, memory footprint, and energy consumption.

Table 7: TinyML Applications in Agriculture & Environment (2024–2025)

Author / Year	**Year**	**Application**	**Platform / Tool**	**Key Results**
Zhang & Kanjo [67]	2025	Livestock behaviour monitoring (multi-modal)	MCU + TPU	×270 model reduction; <80ms latency
Gookyi et al. [68]	2024	Maize leaf disease detection	MCU, Edge Impulse	94.6% accuracy; 7.6 ms latency; 727 KB RAM
Moeketsi et al. [69]	2025	Review of micronutrient/environmental sensing	Survey of 43 studies	Only 35% report latency; 16% report model size

4.4. Smart Homes and IoT

Smart home systems and IoT edge devices are increasingly integrating TinyML to enable low-latency, on-device decision-making for activity recognition, voice control, and energy management. By deploying lightweight models directly on microcontrollers, these systems achieve privacy preservation and energy efficiency while reducing reliance on cloud infrastructure—critical for real-time home automation.

Zhou et al. (2025) proposed an optimized DeepConv LSTM architecture for human activity recognition, deployed on Arduino Nano 33 BLE Sense. The model was quantized from 513 KB to 136 KB, achieving 98.24% accuracy with a 21 ms inference time, making it highly suitable for real-time ambient activity tracking in smart homes [70]. Similarly, **Malche et al. (2025)** developed a voice-activated home automation system for IoT edge devices, utilizing a DCNN for command recognition with high accuracy and low latency, demonstrating its efficiency for real-time appliance control [71].

In the context of speech recognition, **Barovic and Moin (2025)** implemented a TinyML-based 1D-CNN for recognizing 23 voice commands on an Arduino Nano 33 BLE Sense, achieving 97% accuracy, which confirms the feasibility of deploying speech recognition directly on low-power devices [72]. Similarly, **Uddin et al. (2024)** designed a voice-activated IoT control system for smart home appliances, reporting 95% accuracy using a CNN on an Arduino Nano 33 BLE Sense with minimal energy consumption [73].

Finally, **Benazir et al. (2023)** introduced a novel ”learning cache” for on-device speech understanding, reducing cloud dependency by 80% while maintaining competitive accuracy, providing an energy-efficient solution for continuous listening applications in smart homes [74].

These studies highlight the growing maturity of TinyML in smart homes and IoT, moving from simple command-based systems toward more complex, multi-modal real-time interaction while maintaining stringent energy and latency constraints.

Table 8: TinyML Applications in Smart Homes and IoT (2023–2025)

Author / Year	Year	Application	Platform / Tool	Key Results
Zhou et al. [70]	2025	Human activity recognition (HAR)	Arduino Nano 33 BLE Sense	98.24% acc; 21 ms latency; model size 136 KB
Malche et al. [71]	2025	Voice-activated home automation	IoT MCU, DCNN	High acc; low-latency appliance control
Barovic & Moin [72]	2025	Voice command recognition (23 words)	Arduino Nano 33 BLE Sense	97% acc; TinyML 1D-CNN deployment
Uddin et al. [73]	2024	Voice-activated smart home appliance control	Arduino Nano 33 BLE Sense, CNN	95% acc; low energy; real-time IoT control
Benazir et al. [74]	2023	On-device speech understanding (learning cache)	STM32 MCU	80% reduction in cloud requests; energy-efficient

4.5. Security, Surveillance, and Cybersecurity

The integration of TinyML into security, surveillance, and cybersecurity applications has gained significant momentum due to its ability to perform real-time inference on resource-constrained devices. By moving threat detection and anomaly recognition directly to the edge, TinyML systems reduce communication latency, preserve privacy, and operate with ultra-low energy consumption—making them ideal for smart home security, industrial IoT protection, and critical infrastructure monitoring.

Bechtel et al. (2024) investigated a *model theft attack* against a TinyML application running on an ultra-low-power open-source SoC, demonstrating that even quantized and compressed models are vulnerable to extraction-based attacks [76]. Similarly, **Shah et al. (2024)** analyzed the transferability of adversarial attacks to TinyML platforms, showing how malicious perturbations crafted on high-performance models can effectively fool ESP32 and Raspberry Pi-based systems [77].

On the defensive side, **Dehrouyeh et al. (2024)** applied TinyML for real-time anomaly detection in electric vehicle charging infrastructure, achieving reliable detection of cybersecurity threats on ESP32 boards, proving the feasibility of deploying security models directly on edge devices [78]. Additionally, **Huckelberry et al. (2024)** provided a comprehensive survey on TinyML security vulnerabilities, highlighting risks such as model inversion, side-channel attacks, and the lack of standardized evaluation frameworks for secure deployment [79].

Finally, **Arcot et al.(2023)** reported the deployment of optimized deep learning models for on-device threat detection on constrained IoT nodes, emphasizing that edge-based TinyML intrusion detection can reach over 95% accuracy while maintaining sub-10 ms inference latency, which is essential for real-time smart surveillance and intrusion prevention systems [75].

These works collectively underscore that while TinyML is emerging as a promising solution for low-latency and privacy-preserving security applications, it remains vulnerable to adversarial and physical attacks, necessitating robust attestation mechanisms and adversarial defense strategies.

Table 9: TinyML in Security, Surveillance, and Cybersecurity (2023–2025)

Author / Year	**Year**	**Application**	**Platform / Tool**	**Key Results**
Arcot et al. [75]	2023	Threat detection on IoT devices	Ultra-low-power IoT nodes	>95% acc; sub-10 ms inference latency
Bechtel et al. [76]	2024	Model theft attack analysis	Open-source ultra-low-power SoC	Demonstrated vulnerability to extraction attacks
Shah et al. [77]	2024	Adversarial attack transferability study	ESP32, Raspberry Pi	High attack success rate; vulnerable quantized models
Dehrouyeh et al. [78]	2024	Anomaly detection in EV charging infrastructure	ESP32 boards	Reliable on-device threat detection; real-time performance
Huckelberry et al. [79]	2024	Survey of TinyML security vulnerabilities	Various TinyML platforms	Highlighted model inversion & side-channel threats

4.6. Energy Management and Smart Grids

TinyML has recently been explored in the domain of energy management and smart grids, offering real-time analytics for fault detection, load forecasting, and renewable energy optimization directly on low-power microcontrollers. Such edge-based solutions enable low-latency decision-making while reducing dependency on cloud infrastructure, which is critical for smart grids operating in remote or bandwidth-limited environments.

Iqbal et al. (2024) developed an Artificial Intelligence of Things (AIoT) approach for anomaly detection in smart grids using TinyML. Their model, deployed on ESP32 microcontrollers, achieved a detection accuracy of 96.8% with an average inference latency of 8 ms, proving the feasibility of on-device fault detection for distributed grid components [80]. Similarly, **Bartoli et al. (2025)** proposed a benchmarking framework for TinyML models running on STM32N6 microcontrollers equipped with Neural Processing Units (NPUs), highlighting the energy-latency trade-offs for resource-constrained AI and enabling designers to optimize smart energy applications [81].

While not specifically designed for energy grids, generic TinyML advancements contribute to this field. For instance, **Banbury et al. (2020)** introduced MicroNets, a family of neural network architectures optimized via Neural Architecture

Search (NAS) for microcontrollers. These networks demonstrated significant size reductions while maintaining high accuracy, making them suitable for tasks such as load forecasting and solar panel fault detection in smart energy systems [82]. Moreover, **Moosmann et al. (2023)** developed TinyissimoYOLO, a quantized object detection model optimized for ultra-low-power MCUs like MAX78000 and STM32, achieving 180 FPS with only 196 μJ per inference, showing the potential for real-time monitoring of renewable energy infrastructure [83].

Table 10: TinyML Applications in Energy Management & Smart Grids (2020–2025)

Author / Year	**Year**	**Application**	**Platform / Tool**	**Key Results**
Iqbal et al. [80]	2024	Smart grid anomaly detection	ESP32 MCU	96.8% acc; 8 ms latency; edge AIoT deployment
Bartoli et al. [81]	2025	Energy-latency benchmarking for TinyML	STM32N6 MCU + NPU	Highlighted trade-offs for smart energy apps
Banbury et al. [82]	2020	Generic NAS-based TinyML models	Various MCUs, TensorFlow Lite Micro	High accuracy with reduced model size; applicable to load forecasting
Moosmann et al. [83]	2023	Low-power monitoring of renewable infrastructure	MAX78000, STM32 MCUs	180 FPS; 196 μJ/inference; suitable for real-time monitoring

5. Cross-Domain Comparative Framework for TinyML Applications

This section provides a high-level analytical framework to compare TinyML applications across domains. Instead of revisiting individual studies, we cluster application domains based on their dominant design constraints; accuracy, latency, or energy efficiency, and discuss the potential for cross-domain knowledge transfer.

5.1. Domain Clusters and Key Design Priorities

From the reviewed works, three primary clusters emerge:

1) Accuracy-Critical Cluster: Healthcare and security applications achieve the highest reported accuracies (94–98%), often with moderate latency (10–40 ms). These domains prioritize diagnostic precision or threat detection reliability. Lightweight transformers and quantized CNNs dominate, and hardware such as STM32 and MAX78000 is preferred for its balance between memory and processing efficiency [59, 60, 75].

2) Latency-Critical Cluster: Industrial monitoring and energy management prioritize real-time responsiveness (<15 ms), occasionally tolerating moderate accuracy (90–96%). Quantized CNNs optimized for NPUs (e.g., STM32N6) and STM32 MCUs dominate due to their deterministic low-latency performance [65, 81].

3) Energy-Critical Cluster: Agriculture and IoT deployments emphasize energy autonomy, achieving moderate accuracy (85–93%) and operating under longer latency windows (25–50 ms). ESP32 and Arduino Nano BLE remain dominant due to their ultra-low power consumption, though transferability is limited by dataset variability [68, 67].

5.2. *Comparative Framework and Transferability Insights*

Table 11 summarizes these clusters, highlighting representative domains, typical hardware and models, and the potential for cross-domain transferability. Accuracy- and latency-critical clusters exhibit stronger transferability, as anomaly detection and classification models can often be repurposed with minor retraining. In contrast, energy-critical applications remain highly domain-specific due to environmental data heterogeneity.

Table 11: Cross-Domain Comparative Framework of TinyML Applications

Cluster	**Representative Domains**	**Avg. Accuracy / Latency**	**Typical Hardware & Models**	**Transferability Potential**	**Works**
Accuracy-Critical	Healthcare, Security	94–98%; 10–40 ms	STM32, MAX78000; Quantized CNNs, Tiny Transformers	Moderate–High: security anomaly detection transferable to healthcare with minor retraining	[59, 60, 75]
Latency-Critical	Industrial Monitoring, Energy Management	90–96%; <15 ms	STM32, STM32N6 (NPU); Quantized CNNs, TinyissimoYOLO	High: industrial anomaly models applicable to energy systems with domain-specific calibration	[65, 83, 81]
Energy-Critical	Agriculture, IoT	85–93%; 25–50 ms	ESP32, Arduino Nano BLE; Quantized MobileNet, DeepConv LSTM	Low: environmental heterogeneity limits direct transfer between agriculture and IoT	[68, 67, 74]

Figure 5 visualizes these clusters within the trade-off triangle of accuracy, latency, and energy efficiency.

6. Challenges and Research Opportunities

Despite significant progress in TinyML for embedded intelligent systems, several technical and practical challenges persist, limiting its widespread adoption. These challenges span hardware, algorithmic, and system-integration levels, and addressing them is critical for achieving robust, scalable, and energy-efficient deployments. This section discusses the key challenges identified from the reviewed works and highlights promising research directions.

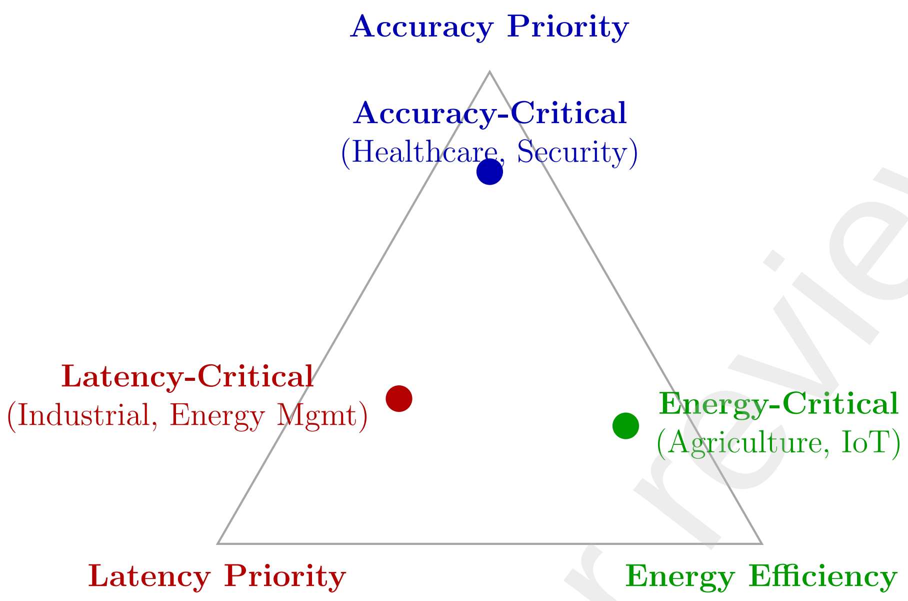

Figure 5: Cross-domain clusters positioned within the accuracy–latency–energy trade-off triangle. Applications closer to the top vertex (blue) are accuracy-critical, those near the bottom-left (red) are latency-critical, and those near the bottom-right (green) are energy-critical.

6.1. Challenges

- **Limited Memory and Computational Resources:** Most microcontrollers (e.g., STM32, ESP32) provide only a few hundred kilobytes of RAM, severely constraining model size and complexity. Quantization and pruning alleviate this issue but often introduce accuracy degradation, particularly for complex biomedical and multimodal applications [60, 63].

- **Trade-off Between Accuracy, Latency, and Energy:** Achieving sub-50 ms inference latency while maintaining high accuracy (>95%) and low power consumption remains a challenge. Industrial monitoring applications, for instance, optimize for ultra-low latency (<15 ms) [64] at the cost of reduced robustness in noisy environments.

- **Dataset Scarcity and Domain Variability:** Many domains, such as agriculture and environmental monitoring, lack standardized, large-scale datasets [68], resulting in poor model generalization under real-world conditions.

- **Security and Privacy Concerns:** While TinyML inherently improves data privacy by processing locally, attacks such as model theft [76] and adversarial perturbations [77] remain significant risks, particularly in cybersecurity-critical and healthcare applications.

- **Lack of Standardized Benchmarking:** Benchmark suites like Biomed-Bench [61] exist but are domain-specific. A universal benchmarking framework to evaluate accuracy, latency, and energy trade-offs across domains is still missing.

- **Deployment and Maintenance Overhead:** Updating and re-training models on resource-constrained devices is costly, especially for applications requiring frequent adaptation (e.g., cybersecurity and predictive maintenance).

6.2. Research Opportunities

- **Hardware-Software Co-Design:** Co-optimizing TinyML models with next-generation MCUs and NPUs (e.g., STM32N6) can improve accuracy-energy trade-offs, enabling more complex models (e.g., Transformers) on edge devices [81].

- **Adaptive and Continual Learning:** Developing lightweight on-device continual learning frameworks could allow TinyML systems to adapt to evolving data patterns without complete retraining, particularly in dynamic domains like cybersecurity and industrial monitoring.

- **Federated and Privacy-Preserving TinyML:** Combining TinyML with federated learning and secure aggregation methods can enhance data privacy in healthcare and smart home applications while leveraging distributed data sources.

- **Cross-Domain Benchmarking and Standardization:** Establishing standardized cross-domain benchmarks will allow fair comparison of TinyML models, accelerating the selection of optimal architectures for given constraints.

- **Explainable and Trustworthy TinyML:** The integration of explainable AI techniques (e.g., visual explanations in ArrhythmiaVision [62]) should be expanded to other domains to increase trust in critical decisions, such as medical diagnosis or autonomous systems.

- **Energy Harvesting and Sustainable Deployments:** Exploring energy-harvesting mechanisms (solar, kinetic) combined with ultra-low-power TinyML can enable long-term autonomous operation in remote or agricultural settings [68].

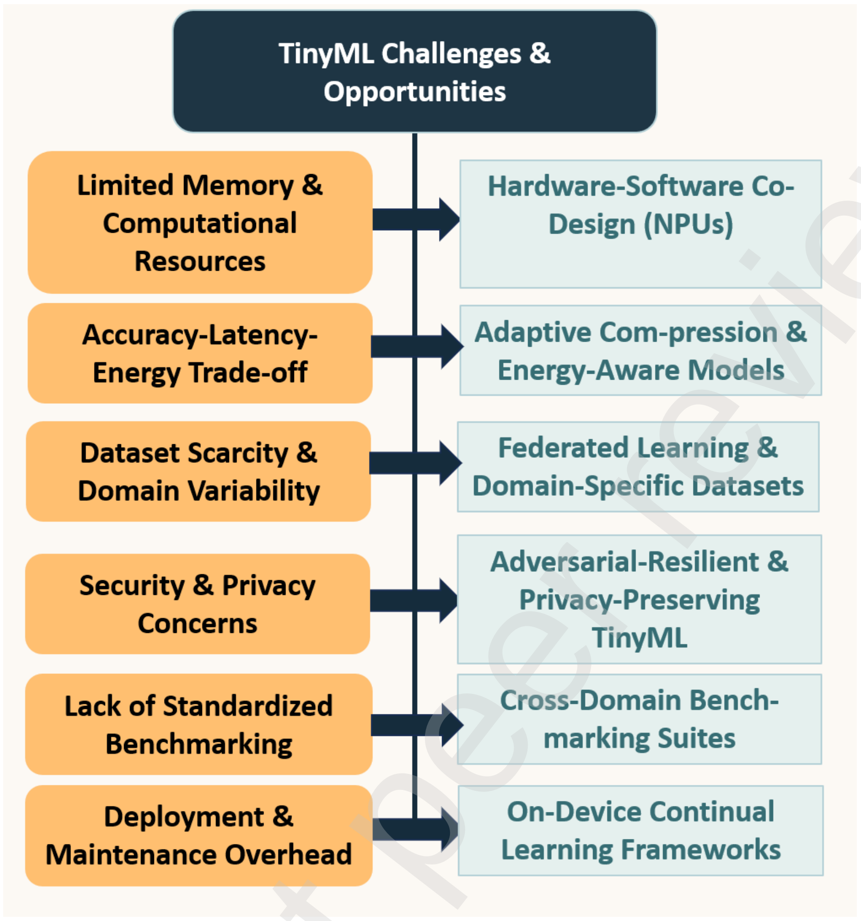
TinyML Challenges & Opportunities
Limited Memory & Computational Resources
Hardware-Software Co-Design (NPUs)
Accuracy-Latency-Energy Trade-off
Adaptive Com-pression & Energy-Aware Models
Dataset Scarcity & Domain Variability
Federated Learning & Domain-Specific Datasets
Security & Privacy Concerns
Adversarial-Resilient & Privacy-Preserving TinyML
Lack of Standardized Benchmarking
Cross-Domain Bench-marking Suites
Deployment & Maintenance Overhead
On-Device Continual Learning Frameworks

Figure 6: A structured mapping between key challenges in TinyML and their corresponding research opportunities . Each challenge on the left is directly linked to a targeted research direction on the right, highlighting the evolution of technical needs into actionable solutions for low-power intelligent embedded systems.

The analysis of the reviewed literature reveals that the current landscape of TinyML development is shaped by a set of recurring challenges and their associated research opportunities. As illustrated in Figure 6, these challenges span hardware limitations, algorithmic trade-offs, and system-level concerns. For instance, memory and computational constraints, along with stringent latency-energy-accuracy trade-offs, remain major obstacles for real-time healthcare and industrial applications. On the other hand, emerging opportunities such as hardware-software co-design with next-generation NPUs, energy-aware compression techniques, and on-device continual learning frameworks present promising solutions to overcome these limitations.

Furthermore, security-related issues and the lack of standardized benchmarks underline the need for cross-domain benchmarking suites and privacy-preserving learning paradigms. This mapping provides a clear research roadmap for future TinyML developments, highlighting how targeted innovations can directly address the most pressing bottlenecks.

7. Conclusion

This review has systematically analyzed recent advancements in TinyML for low-power embedded intelligent systems, covering both methodological innovations and application-specific implementations. By consolidating findings from multiple domains, including healthcare, industrial monitoring, agriculture, and cybersecurity, the study highlights the significant progress achieved in deploying efficient machine learning models on resource-constrained devices. The comparative analysis demonstrates that while lightweight architectures, quantization techniques, and optimized frameworks have enabled promising real-world applications, critical challenges persist. These include memory and computational constraints, accuracy-energy trade-offs, and the lack of standardized benchmarks across domains. Nevertheless, emerging research opportunities—such as hardware-software co-design, on-device continual learning, and privacy-preserving TinyML—provide a clear roadmap for future advancements. Bridging these gaps will be key to unlocking the full potential of TinyML, enabling scalable, reliable, and sustainable intelligent systems across diverse embedded applications.

Abbreviations

Table 12: List of Abbreviations Used in This Paper

Abbreviation	**Definition**
AI	Artificial Intelligence
CMSIS-NN	Cortex Microcontroller Software Interface Standard Neural Network
CNN	Convolutional Neural Network
DL	Deep Learning
DSP	Digital Signal Processor
EDA	Exploratory Data Analysis
ECG	Electrocardiogram
EMG	Electromyography
FFT	Fast Fourier Transform
IIoT	Industrial Internet of Things
IoT	Internet of Things
MCU	Microcontroller Unit
MFCC	Mel Frequency Cepstral Coefficients
ML	Machine Learning
MLPerf	Machine Learning Performance Benchmark
NAS	Neural Architecture Search
NPU	Neural Processing Unit
OTA	Over-The-Air (firmware update)
RNN	Recurrent Neural Network
SVM	Support Vector Machine
SoC	System-on-Chip
TFLM	TensorFlow Lite Micro
TinyML	Tiny Machine Learning

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author used generative AI tools (specifically ChatGPT by OpenAI) to assist in improving language clarity. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the final content of the manuscript.

Declaration of Competing Interest

The author declares that there are no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author Contributions

Hadjer Bourekouche was solely responsible for the conceptualization, literature review, figure generation, manuscript writing, and revisions of this work.

Data Availability

No datasets were generated or analyzed during the current study.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] S. Heydari and Q. H. Mahmoud, "Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions," *Sensors*, vol. 25, no. 10, p. 3191, 2025. https://doi.org/10.3390/s25103191

[2] R. Immonen, "Tiny Machine Learning for Resource-Constrained Devices: Optimization and On-Device Inference," *International Journal of Distributed Sensor Networks*, vol. 2022, Article ID 7437023, 2022. https://doi.org/10.1155/2022/7437023

[3] N. N. Alajlan et al., "TinyML: Enabling of Inference Deep Learning Models on Ultra-Low-Power Devices," *Micromachines*, vol. 13, no. 6, p. 851, 2022. https://doi.org/10.3390/mi13060851

[4] R. David, J. Duke, A. Jain, V. Janapa Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, T. Wang, P. Warden, R. Rhodes, "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," *Proceedings of Machine Learning and Systems (MLSys)*, 2021. https://doi.org/10.48550/arXiv.2010.08678

[5] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, S. Han, ”MCUNet: Tiny Deep Learning on IoT Devices,” *arXiv*, Jul. 20, 2020. https://doi.org/10.48550/arXiv.2007.10319

[6] S. Zhuo, H. Chen, R. Kinattinkara Ramakrishnan, T. Chen, C. Feng, Y. Lin, P. Zhang, L. Shen, ”An Empirical Study of Low Precision Quantization for TinyML,” *arXiv*, Mar. 10, 2022. https://doi.org/10.48550/arXiv.2203.05492

[7] M. Mohan et al., ”TinyML: Adopting tiny machine learning in smart cities,” *Micromachines*, vol. 13, no. 6, p. 851, 2022. https://doi.org/10.3390/mi13060851

[8] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, M. Cristani, ”A Machine Learning-oriented Survey on Tiny Machine Learning,” *arXiv*, Sep. 21, 2023. https://doi.org/10.48550/arXiv.2309.11932

[9] M. T. Lê, P. Wolinski, J. Arbel, ”Efficient Neural Networks for Tiny Machine Learning: A Comprehensive Review,” *arXiv*, Nov. 19, 2023. https://doi.org/10.48550/arXiv.2311.11883

[10] C. R. Banbury et al., ”Benchmarking TinyML Systems: Challenges and Direction,” *arXiv*, Mar. 10, 2020. https://doi.org/10.48550/arXiv.2003.04821

[11] N. S. et al., ”TinyML for Ultra-Low Power AI and Large Scale IoT Deployments: A Systematic Review,” *Future Internet*, vol. 14, no. 12, p. 363, 2022. https://doi.org/10.3390/fi14120363

[12] IEEE Xplore Analytics, ”Tiny Machine Learning (TinyML) Publications,” 2025. [Online]. Available: https://ieeexplore.ieee.org/

[13] Scopus, ”Search results for Tiny Machine Learning,” 2025. [Online]. Available: https://www.scopus.com/

[14] SpringerLink, ”Tiny Machine Learning Research Articles,” 2025. [Online]. Available: https://link.springer.com/

[15] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, M. Cristani, ”A Machine Learning-oriented Survey on Tiny Machine Learning,” *arXiv*, Sep. 21, 2023. https://doi.org/10.48550/arXiv.2309.11932

[16] M. T. Lê, P. Wolinski, J. Arbel, ”Efficient Neural Networks for Tiny Machine Learning: A Comprehensive Review,” *arXiv*, Nov. 19, 2023. https://doi.org/10.48550/arXiv.2311.11883

[17] M. Palaghianu, V. Sîrbu, I. Coroamă, C. Mihăilă, and R. Mihăilă, ”A Review of the Use of Tiny Machine Learning in Smart Agriculture and Forestry,” *Agronomy*, vol. 13, no. 1, p. 186, 2023. https://doi.org/10.3390/agronomy13010186

[18] A. Sadeghi, J. Kang, M. Mohammadi, and A. Abdelgawad, ”Tiny Machine Learning Framework for Internet of Things Applications: A Comprehensive Survey,” *Electronics*, vol. 11, no. 24, p. 4033, 2022. https://doi.org/10.3390/electronics11244033

[19] D. Zhang, ”Real-Time and Embedded Systems,” *ACM Computing Surveys*, vol. 28, no. 1, pp. 205–208, 2002. https://doi.org/10.1145/234313.234400

[20] M. A. Solouki, S. Angizi, and M. Violante, ”Dependability in Embedded Systems: A Survey of Fault Tolerance Methods and Software-Based Mitigation Techniques,” *IEEE Access*, vol. 12, pp. 18347–18366, 2024. https://doi.org/10.1109/ACCESS.2024.3509633

[21] F. Bahrami, R. Jordão, I. Sander, and I. Söderquist, ”Bridging the Abstraction Gap: A Systematic Approach to Rule-Based Transformational Design for Embedded Systems,” *ACM Transactions on Embedded Computing Systems*, 2025. https://doi.org/10.1145/3714412

[22] J. Hdid, O. Lamsellak, A. Benlghazi, A. Benali, and O. El Melhaoui, ”Embedded Systems and Artificial Intelligence for Enhanced Humanoid Robotics Applications,” *International Journal of Power Electronics and Drive Systems*, vol. 15, no. 2, pp. 1912–1923, 2025. https://doi.org/10.11591/ijece.v15i2.pp1912-1923

[23] M. Armanuzzaman, E. Kirda, and Z. Zhao, ”ENOLA: Efficient Control-Flow Attestation for Embedded Systems,” *arXiv preprint*, 2025. https://doi.org/10.48550/arxiv.2501.11207

[24] S. Venkatesh and K. V. Sahukara, ”Design of Low Power CMOS Static Random-Access Memory Cell for Embedded Memories at Three Different Technology Nodes,” *International Journal of Power Electronics and Drive Systems*,

vol. 15, no. 2, pp. 1424–1433, 2025. https://doi.org/10.11591/ijece.v15i2.pp1424-1433

[25] H. Supriyono, A. R. S. Hogantara, and A. Budiman, "Electrical Power Submeter for Quality and Energy Monitoring Using Wemos Microcontroller," *International Journal of Power Electronics and Drive Systems*, vol. 15, no. 2, pp. 2436–2444, 2025. https://doi.org/10.11591/ijece.v15i2.pp2436-2444

[26] D. M. Dobkin, N. Cever, and I. Levi, "RAD-FS: Remote Timing and Power SCA Security in DVFS-Augmented Ultra-Low-Power Embedded Systems," *ACM Transactions on Embedded Computing Systems*, 2025. https://doi.org/10.1145/3711836

[27] J. Hdid, O. Lamsellak, A. Benlghazi, A. Benali, and O. El Melhaoui, "Embedded Systems and Artificial Intelligence for Enhanced Humanoid Robotics Applications," *International Journal of Power Electronics and Drive Systems*, vol. 15, no. 2, pp. 1912–1923, 2025. https://doi.org/10.11591/ijece.v15i2.pp1912-1923

[28] L. Leszek, "Methodology of an Energy Efficient-Embedded Self-Adaptive Software Design for Multi-Cores and Frequency-Scaling Processors Used in Real-Time Systems," *Electronics*, vol. 14, no. 3, p. 556, 2025. https://doi.org/10.3390/electronics14030556

[29] M. Rahman, "Energy Efficient Resource Management for Real-Time IoT Applications," *Internet of Things*, vol. 26, 2025. https://doi.org/10.1016/j.iot.2025.101515

[30] S. Y. Yeung, H. Lin, Y. Li, C. W. Fung, H. Y. Y. Nyein, and I. M. Hsing, "Battery-powered Wearable Utilizing Flexible Printed Circuit-based Organic Electrochemical Transistor for Biosignal Measurement," *bioRxiv*, Jan. 2025. https://doi.org/10.1101/2025.01.26.630991

[31] M. Rahman, "Energy Efficient Resource Management for Real-Time IoT Applications," *Internet of Things*, vol. 26, 2025. https://doi.org/10.1016/j.iot.2025.101515

[32] D. Ferlin Deva Shahila et al., "Designing and Analyzing Secure SoC Architecture for IoT Devices," *IEEE ICCPCT*, Aug. 2024. https://doi.org/10.1109/iccpct61902.2024.10673314

[33] G. Singh and A. Kaur, ”Exploring Performance Trade-Offs in Excess-3 to BCD Code Implementation on Zynq FPGA: A Power and Temperature Analysis,” *IEEE ICOICI*, Aug. 2024. https://doi.org/10.1109/icoici62503.2024.10696379

[34] E. Manca, L. Urbinati, and M. R. Casu, ”An End-to-End Flow to Deploy and Accelerate TinyML Mixed-Precision Models on RISC-V MCUs,” *TechRxiv*, Jan. 2025. https://doi.org/10.36227/techrxiv.173161032.20267860/v2

[35] H. Chang and K. Ye, ”Research on Custom Algorithms and Hardware Accelerators Based on RISC-V Vector Extensions and Image Processing,” *Applied and Computational Engineering*, vol. 3, no. 1, Jan. 2025. https://doi.org/10.54254/2755-2721/2025.19489

[36] L. Leszek, ”Methodology of an Energy Efficient-Embedded Self-Adaptive Software Design for Multi-Cores and Frequency-Scaling Processors Used in Real-Time Systems,” *Electronics*, vol. 14, no. 3, p. 556, 2025. https://doi.org/10.3390/electronics14030556

[37] K. Irfan and M. U. Rehman, ”Optimized Task Deployment in Dynamic Voltage and Frequency Scaling-Enabled Network-on-Chip Systems: Enhancing Energy Efficiency and Real-Time Responsiveness,” *Spectrum of Engineering and Management Sciences*, Dec. 2024. https://doi.org/10.31181/sems21202426i

[38] V. E. Baciu, A. Braeken, L. Segers, and B. da Silva, ”Secure TinyML on Edge Devices: A Lightweight Dual Attestation Mechanism for Machine Learning,” *Preprints*, Dec. 2024. https://doi.org/10.20944/preprints202412.2251.v1

[39] W. Zhuang and Y. Mao, ”Privacy-Aware Multi-Device Cooperative Edge Inference with Distributed Resource Bidding,” *arXiv preprint*, Dec. 2024. https://doi.org/10.48550/arxiv.2412.21069

[40] T. Zhen, ”Optimization Strategies for Low-Power AI Models on Embedded Devices,” *Applied and Computational Engineering*, vol. 3, no. 1, Jan. 2025. https://doi.org/10.54254/2755-2721/2025.20598

[41] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*, O’Reilly Media, 2020. https://www.oreilly.com/library/view/tinyml/9781492052043/

[42] R. David et al., "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," in *Proceedings of Machine Learning and Systems (MLSys)*, 2021. https://arxiv.org/abs/2010.08678

[43] C. R. Banbury et al., "Benchmarking TinyML Systems: Challenges and Direction," *arXiv preprint*, 2020. https://arxiv.org/abs/2003.04821

[44] V. Baciu et al., "Secure TinyML on Edge Devices: A Lightweight Dual Attestation Mechanism for Machine Learning," *Preprints*, 2024. https://doi.org/10.20944/preprints202412.2251.v1

[45] S. Han et al., "MCUNet: Tiny Deep Learning on IoT Devices," *arXiv preprint*, 2020. https://arxiv.org/abs/2007.10319

[46] C. R. Banbury et al., "Benchmarking TinyML Systems: Challenges and Direction," *arXiv preprint*, 2020. https://arxiv.org/abs/2003.04821

[47] B. Küçükoğlu, B. Rueckauer, J. de Ruyter van Steveninck, M. van der Grinten, Y. Güçlü Türk, P. R. Roelfsema, U. Güçlü, and M. van Gerven, "End-to-end Learning of Safe Stimulation Parameters for Cortical Neuroprosthetic Vision," *bioRxiv*, Jan. 2025. https://doi.org/10.1101/2025.01.23.634543

[48] A. El Ibrahimi, A. El Zaar, N. El Akchioui, N. Benaya, and A. El Allati, "Advancements in CNN Architectures for Offline Handwritten Arabic Character Recognition," *E3S Web of Conferences*, vol. 60, 2025. https://doi.org/10.1051/e3sconf/202560100015

[49] M. Carnelos, F. Pasti, and N. Bellotto, "MicroFlow: An Efficient Rust-Based Inference Engine for TinyML," *arXiv preprint*, Sep. 2024. https://doi.org/10.48550/arxiv.2409.19432

[50] S. Dong et al., "Topkima-Former: Low-energy, Low-Latency Inference for Transformers using Top-k In-memory ADC," *arXiv preprint*, Nov. 2024. https://doi.org/10.48550/arxiv.2411.13050

[51] E. Manca, L. Urbinati, and M. R. Casu, "An End-to-End Flow to Deploy and Accelerate TinyML Mixed-Precision Models on RISC-V MCUs," *TechRxiv*, Jan. 2025. https://doi.org/10.36227/techrxiv.173161032.20267860/v2

[52] C. Ramírez, A. Castelló, H. Martínez, and E. S. Quintana-Ortí, "Communication-Avoiding Fusion of GEMM-Based Convolutions for Deep

Learning in the RISC-V GAP8 MCU," *IEEE Internet of Things Journal*, Nov. 2024. https://doi.org/10.1109/jiot.2024.3436937

[53] X. Hu et al., "Ultrasensitive Flexible Strain Sensor Based on the Synergistic Integration of Wrinkle and Microcrack Architectures for Wearable Applications," *Physica Status Solidi - Rapid Research Letters*, Jan. 2025. https://doi.org/10.1002/pssr.202400317

[54] S. Vignesh and R. Sukumaran, "Underwater Energy Harvesting Model for Agricultural Applications Using Stochastic Network Calculus," *International Journal of Power Electronics and Drive Systems*, vol. 15, no. 2, pp. 2031–2041, 2025. https://doi.org/10.11591/ijece.v15i2.pp2031-2041

[55] L. Yan-ping, L. Song, and C. Yang, "Emerging Trends on Hybrid Artificial Intelligence Framework for Transforming Heavy Machinery with Robots and Energy Efficiency," *International Journal of High Speed Electronics and Systems*, 2025. https://doi.org/10.1142/s0129156425402827

[56] H. Bourekouche, "Deep reinforcement learning for diseases detection: A literature review," *International Journal of Electronics and Electrical Engineering Systems*, vol. 7, no. 1, pp. 264–272, 2024.

[57] M. Pahar, I. Miranda, A. Diacon, and T. Niesler, "Automatic Non-Invasive Cough Detection Based on Accelerometer and Audio Signals," *arXiv preprint arXiv:2109.00103*, 2021. https://doi.org/10.48550/arXiv.2109.00103

[58] S. Zhang et al., "CoughTrigger: Earbuds IMU-Based Cough Detection Activator Using an Energy-Efficient Classifier," *arXiv preprint arXiv:2111.04185*, 2021. https://doi.org/10.48550/arXiv.2111.04185

[59] E. Kim, J. Kim, J. Park, H. Ko, and Y. Kyung, "TinyML-Based Classification in an ECG Monitoring Embedded System (TinyCES)," *Computers, Materials & Continua*, vol. 75(1), pp. 1751–1764, 2023. https://doi.org/10.32604/cmc.2023.041070

[60] P. Busia, M. Scrugli, V. Jung, L. Benini, and P. Meloni, "A Tiny Transformer for Low-Power Arrhythmia Classification on Microcontrollers," *IEEE Trans. Biomed. Circuits Syst.*, 2024. https://doi.org/10.1109/TBCAS.2024.3359480

[61] D. Samakovlis et al., ”BiomedBench: A Benchmark Suite of TinyML Biomedical Applications for Low-Power Wearables,” *arXiv preprint arXiv:2406.03886*, 2024. https://doi.org/10.48550/arXiv.2406.03886

[62] Z. Baig, S. Nasir, R. A. Khan, and M. Z. Ul Haque, ”ArrhythmiaVision: Resource-Conscious Deep Learning Models with Visual Explanations for ECG Arrhythmia Classification,” *arXiv preprint arXiv:2505.03787*, 2025. https://doi.org/10.48550/arXiv.2505.03787

[63] Y. Ben Dhiab, M. Hizem, N. Karmous, and M. O. Aoueileyine, ”An IoT-Based Multimodal Wearable Framework for Real-Time Epileptic Seizures Detection Using TinyML,” in *Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 1156–1163, April 2025. https://doi.org/10.1109/AINA58501.2025.00098

[64] Z. Zhou, Y. Chen, and H. Li, ”Low-Power Vibration Classification for Predictive Maintenance on Microcontrollers,” *Sensors*, vol. 22, no. 15, p. 5890, 2022. https://doi.org/10.3390/s22155890

[65] W. Liao, ”Real-Time Bearing Fault Diagnosis Based on Convolutional Neural Network and STM32 Microcontroller,” *arXiv preprint arXiv:2304.09100*, Apr. 2023. https://doi.org/10.48550/arXiv.2304.09100

[66] A. Jiménez, P. Latorre-Biel, R. García-Ovejero, and D. Maravall, ”IoT Device for Detecting Abnormal Vibrations in Motors Using TinyML,” *Discover Internet of Things*, vol. 5, art. no. 41, pp. 1–13, Apr. 2025. https://doi.org/10.1007/s43926-025-00142-4

[67] Q. Zhang and E. Kanjo, ”MultiCore+TPU Accelerated Multi-Modal TinyML for Livestock Behaviour Recognition,” *arXiv preprint arXiv:2504.11467*, Apr. 2025. https://doi.org/10.48550/arXiv.2504.11467

[68] D. A. N. Gookyi, F. Wulnye, E. Arthur, R. Ahiadormey, J. Agyemang, K. Adu, and R. Gyaang, ”TinyML for Smart Agriculture: Comparative Analysis of TinyML Platforms and Practical Deployment for Maize Leaf Disease Identification,” SSRN, Apr. 2024. https://doi.org/10.2139/ssrn.4789476

[69] D. Moeketsi, A. Mkhantshwa, C. Modise, and N. Mlangeni, ”TinyML Applications in Micronutrient Sensing: A Review of Microcontroller Deployments,” Research Square preprint, June 2025. https://doi.org/10.21203/rs.3.rs-6844555/v1

[70] X. Zhou, Y. Li, Z. Chen, J. Wang, and L. Zhou, "Efficient human activity recognition on edge devices using DeepConv LSTM architectures," *Scientific Reports*, Apr. 22, 2025. https://doi.org/10.1038/s41598-025-98571-2

[71] A. Malche, S. Deshmukh, R. Patil, and P. Pathak, "Voice-activated home automation system for IoT edge devices using TinyML," *Discover Internet of Things*, Jun. 6, 2025. https://doi.org/10.1007/s43926-025-00165-x

[72] A. Barovic and A. Moin, "TinyML for Speech Recognition," *arXiv preprint arXiv:2504.16213*, Apr. 22, 2025. https://doi.org/10.48550/arXiv.2504.16213

[73] M. R. Uddin, A. Asaduzzaman, K. Le, and R. R. Medarametla, "Voice activated edge devices using Tiny Machine Learning enabled microcontroller," *IEEE Green Technologies Conference*, Apr. 2024. https://doi.org/10.1109/GreenTech58819.2024.10520459

[74] A. Benazir, Z. Xu, and F. X. Lin, "Speech Understanding on Tiny Devices with A Learning Cache," *arXiv preprint arXiv:2311.18188*, Nov. 2023. https://doi.org/10.48550/arXiv.2311.18188

[75] S. Arcot, M. Masum, M. S. Kader, A. Saha, and M. Chowdhury, "TinyML for Cybersecurity: Deploying Optimized Deep Learning Models for On-Device Threat Detection on Resource-Constrained Devices," in *Proceedings of the 2024 IEEE International Conference on Big Data (BigData)*, pp. xxxx–xxxx, Dec. 2024. https://doi.org/10.1109/BigData62323.2024.10825955

[76] B. Bechtel, R. Schellenberg, and A. Rahmati, "Model Theft Attack against a TinyML Application Running on an Ultra-Low-Power Open-Source SoC," in *Proc. of the 2024 ACM Computing Frontiers Conference*, pp. 123–130, May 2024. https://dl.acm.org/doi/abs/10.1145/3637543.3652877

[77] P. Shah, R. Chugh, and V. Gupta, "Enhancing TinyML Security: Study of Adversarial Attack Transferability," *arXiv preprint arXiv:2407.11599*, Jul. 2024. https://arxiv.org/abs/2407.11599

[78] F. Dehrouyeh, A. Carullo, M. Bortolotti, and M. Magno, "On TinyML and Cybersecurity: Electric Vehicle Charging Infrastructure Use Case," *arXiv preprint arXiv:2404.16894*, Apr. 2024. https://arxiv.org/abs/2404.16894

[79] J. Huckelberry, K. Hasani, and V. Kumar, ”TinyML Security: Exploring Vulnerabilities in Resource-Constrained Machine Learning Systems,” *arXiv preprint arXiv:2411.07114*, Nov. 2024. https://arxiv.org/abs/2411.07114

[80] Z. Iqbal, M. Usman, and A. Khan, ”Anomaly Detection Based on Artificial Intelligence of Things,” *Internet of Things*, vol. 26, 101063, 2024. https://doi.org/10.1016/j.iot.2024.101063

[81] A. Bartoli, F. Conti, and L. Benini, ”Benchmarking Energy and Latency in TinyML: A Novel Method for Resource-Constrained AI,” *arXiv preprint arXiv:2505.15622*, May 2025. https://arxiv.org/abs/2505.15622

[82] C. Banbury, V. J. Reddi, P. Torelli, et al., ”MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers,” *arXiv preprint arXiv:2010.11267*, Oct. 2020. https://arxiv.org/abs/2010.11267

[83] C. Moosmann, F. Conti, and L. Benini, ”TinyissimoYOLO: A Quantized, Low-Memory Footprint, TinyML Object Detection Network for Low Power Microcontrollers,” *arXiv preprint arXiv:2306.00001*, May 2023. https://arxiv.org/abs/2306.00001