IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 August 2025, accepted 13 November 2025, date of publication 24 November 2025,
date of current version 8 December 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3633575

SURVEY

A Systematic Review of State-of-the-Art TinyML Applications in Healthcare, Education, and Transportation

CHAYMAE YAHYATI[1], ISMAIL LAMAAKAL[1], (Student Member, IEEE), YASSINE MALEH[2], (Senior Member, IEEE), KHALID EL MAKKAOUI[1], (Senior Member, IEEE), IBRAHIM OUAHBI[1], MAY ALMOUSA[3], AND AHMED A. ABD EL-LATIF[4,5], (Senior Member, IEEE)

[1]Multidisciplinary Faculty of Nador, Mohammed Premier University, Oujda 60000, Morocco
[2]Laboratory LaSTI, ENSAK, Sultan Moulay Slimane University, Khouribga 54000, Morocco
[3]Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
[4]EIAS Data Science Laboratory, College of Computer and Information Sciences, Center of Excellence in Quantum and Intelligent Computing, Prince Sultan University, Riyadh 11586, Saudi Arabia
[5]Jadara University Research Center, Jadara University, Irbid 21110, Jordan

Corresponding author: Yassine Maleh (yassine.maleh@ieee.org)

Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2025R752), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. Also, the authors would like to thank Prince Sultan University for their support.

ABSTRACT Tiny Machine Learning (TinyML) has emerged as a transformative paradigm enabling machine learning inference directly on ultra-low-power microcontrollers and edge devices. As AI expands beyond cloud computing to resource-constrained environments, TinyML offers promising solutions for latency-sensitive, bandwidth-efficient, and privacy-preserving applications. This paper presents a Systematic review of state-of-the-art TinyML applications across three critical domains: healthcare, education, and transportation. By analyzing 136 peer-reviewed publications from 2020 to 2025, we identify key trends, representative use cases, and the enabling technologies that support domain-specific deployments. Our review evaluates software frameworks, hardware platforms, model optimization techniques (e.g., quantization, pruning, and neural architecture search), and real-world deployment challenges such as energy consumption, memory limitations, and explainability. We further synthesize the metrics used to assess TinyML systems and highlight open research questions. Unlike previous surveys, our domain-centric approach offers a deeper contextual analysis of how TinyML is being adapted to solve real-world problems across diverse sectors. We conclude by outlining future directions and practical insights to guide researchers and practitioners in designing scalable, resilient, and ethically grounded TinyML systems.

INDEX TERMS Edge AI, edge machine learning, edge intelligence, Internet of Things, model compression, embedded systems, TinyML.

I. INTRODUCTION

A. BACKGROUND AND CONTEXT

The proliferation of connected devices and the growth of the Internet of Things (IoT) have given rise to a new paradigm in intelligent computing one that moves away from centralized cloud processing toward on-device intelligence [11]. In conventional machine learning (ML) systems, data collected from sensors or edge devices is transmitted to cloud servers for processing and inference [13]. While this architecture provides access to powerful computation and storage resources, it also introduces significant drawbacks

The associate editor coordinating the review of this manuscript and approving it for publication was Binit Lukose.

such as high latency, dependency on network connectivity, increased energy consumption due to frequent transmissions, and elevated privacy risks [12]. These limitations become especially critical in real-time [219], mission-critical [68], or privacy-sensitive applications, such as patient monitoring in healthcare, driver behavior tracking in transportation [168], [178], or adaptive learning in education [167].

TinyML addresses these challenges by enabling ML models to run locally on microcontrollers and other highly constrained hardware platforms [194]. These devices typically feature less than 1MB of memory, operate at clock speeds of under 100 MHz, and consume milliwatts or even microwatts of power. Despite these resource limitations, TinyML techniques through quantization, pruning, and neural architecture optimization make it possible to execute meaningful AI tasks [196], such as image classification [9], [10], audio recognition, or time-series forecasting, directly on these devices. This capability unlocks a range of applications in edge environments that demand low-power, low-latency, and offline intelligence [166].

Furthermore, TinyML supports the democratization of AI by lowering the barrier to entry for developers, educators, and engineers [160], [161]. With open-source platforms like TensorFlow Lite for Microcontrollers (TFLM) [151], Edge Impulse, and Arduino IDE [1], users can rapidly prototype and deploy AI applications without access to expensive cloud infrastructure. This is especially impactful in underserved regions or low-resource environments where connectivity and computational resources are scarce [195]. For example, low-cost TinyML-enabled devices can provide real-time analytics in rural health clinics [90], monitor environmental conditions in agriculture, or support interactive learning tools in schools without internet access [194].

The field is also intersecting with advances in embedded hardware and software ecosystems. New microcontrollers are being developed with built-in AI accelerators [201], dedicated digital signal processing (DSP) units, and optimized memory hierarchies to support neural inference efficiently [202]. Software libraries such as CMSIS-NN, uTVM, and STM32Cube.AI are being tailored to squeeze maximum performance from these chips [195]. Additionally, research into event-driven and neuromorphic computing is pushing the boundaries of what is possible in sub-milliwatt ML [1].

As TinyML matures, it is no longer seen merely as a subfield of embedded systems or an extension of ML model compression [165]. Instead, it is emerging as a holistic systems challenge that involves new compiler technologies, low-power circuit design, novel ML paradigms, and human-centered deployment strategies [199]. It also invites new concerns, including algorithmic fairness, explainability, and responsible AI deployment in settings where user trust and system robustness are paramount [200].

Therefore, the significance of TinyML extends beyond its technical novelty. It represents a foundational shift in how AI is delivered moving intelligence from the cloud to the edge, from high-power servers to low-power sensors, and from abstract algorithms to everyday [177], tangible use cases in healthcare, education [179], and transportation. This review seeks to capture this paradigm shift by analyzing current research trends, technical contributions, deployment contexts, and cross-domain implications of TinyML applications.

B. MOTIVATION

While a wide range of ML applications have been proposed over the years, the unique combination of energy-efficiency, cost-effectiveness, and privacy offered by TinyML has created new opportunities for deploying smart systems in previously unreachable environments. In healthcare, TinyML supports real-time patient monitoring, fall detection, and chronic disease prediction using wearable sensors. In education, it empowers low-cost interactive learning tools, emotion recognition, and hands-on AI experiments, especially in resource-constrained classrooms. In the transportation domain, TinyML facilitates smart mobility, predictive maintenance, and on-board safety systems within limited hardware.

Despite these promising advances, the literature on TinyML remains fragmented, with little cross-domain synthesis of its capabilities, limitations, and technological stack. Most existing surveys focus on TinyML in general or on technical implementation aspects such as hardware acceleration or model optimization. There remains a substantial gap in structured, comparative analysis of how TinyML is applied across critical real-world domains, particularly in healthcare, education, and transportation. This review is motivated by the need to consolidate state-of-the-art work in these fields and provide practical insights for both academic researchers and system developers.

C. OBJECTIVES AND CONTRIBUTIONS

This paper aims to provide a systematic review of state-of-the-art TinyML applications with a focus on three strategic domains: healthcare, education, and transportation. These sectors have seen increasing integration of embedded intelligence and present unique challenges for resource-constrained AI. The key contributions of this review are as follows:

- It presents a structured taxonomy of TinyML applications in healthcare, education, and transportation, detailing their use cases, architectures, and performance characteristics.
- It critically analyzes open research challenges including energy profiling, hardware-software alignment, explainability, and data scarcity.
- It reviews commonly used hardware platforms and software tools that underpin TinyML deployments.
- It provides a synthesis of evaluation metrics used in TinyML applications, including accuracy, inference latency, memory footprint, energy consumption, and robustness.

TABLE 1. Summary of key abbreviations used in this review.

Abbr.	Definition	Abbr.	Definition	Abbr.	Definition
BPNN	Backpropagation Neural Network	CNN	Convolutional Neural Network	V-CNN	Volumetric Convolutional Neural Network
ANN	Artificial Neural Network	VRES-CNN	Vision Residual CNN	ST-GCN	Spatiotemporal Graph CNN
DualGCNN	Dual Graph CNN	YOLOv3	You Only Look Once v3	Acc	Accuracy
CK	Cohn-Kanade dataset	OSCASA	Online Student Community Assessment Sentiment Analysis	CBin-NN	Compressed Binary NN
PMVs	Personal Mobility Vehicles	TAC	Tiny Anomaly Compress	FusionGCNN	Fusion Graph CNN
DBP	Diastolic Blood Pressure	SBP	Systolic Blood Pressure	DNN	Deep Neural Network
MLP	Multilayer Perceptron	SVM	Support Vector Machine	M-SVM	Multiclass Support Vector Machine
NAS	Neural Architecture Search	LSTM	Long Short-Term Memory	ConvLSTM	Convolutional LSTM
RNN	Recurrent Neural Network	DCNN	Deep CNN	STFT	Short-Time Fourier Transform
TSDNN	Time Series DNN	OTCD	Online Time-series Classification with Concept Drift	MAPE	Mean Absolute Percentage Error
AAVs	Autonomous Aerial Vehicles	MAE	Mean Absolute Error	RMSE	Root Mean Squared Error
MCC	Matthews Correlation Coefficient	ECG	Electrocardiogram	PPG	Photoplethysmography
HDC	Hyperdimensional Computing	CR	Classification Rate	MCR	Misclassification Rate
G	Geometric	LDA	Linear Discriminant Analysis	GNB	Gaussian Naive Bayes
MFCC	Mel-Frequency Cepstral Coefficients	SOTA	State-of-the-Art	CAE	Convolutional Autoencoder
DS-CNN	Depthwise Separable CNN	dB SNDR	Signal-to-Noise-and-Distortion Ratio	FPGA	Field-Programmable Gate Array
MCUs	Microcontroller Units	VOC Sensors	Volatile Organic Compound Sensors	LOC devices	Lab-on-a-Chip Devices
Prec	Precision	Rec	Recall	Sens	Sensitivity
Spec	Specificity	ObjDet	Object Detection	CCA	Canonical Correlation Analysis
NCA	Neighborhood Component Analysis	mIoU	Mean Intersection over Union	mDice	Mean Dice Coefficient
GFLOPS	Giga Floating-point Operations per Second	DSC	Dice Similarity Coefficient	mAP	mean Average Precision
MAD	Mean Absolute Deviation	FPS	Frames Per Second	MUCM-Net	Multi-modal U-shaped CNN Multi-scale Net
UCM-Net	U-shaped CNN Multi-scale Network	M	Millions of Parameters	RPi5	Raspberry Pi 5
SqueezeNet	Squeezed Network	ResNet	Residual Network	SigNet	Signature Network
FIR Filter	Finite Impulse Response Filter	SG Filter	Savitzky-Golay Filter	TEDA	Typicality and Eccentricity Data Analysis

- It offers future research directions that emphasize emerging trends such as event-driven architectures, federated learning on microcontrollers, and ethical concerns in embedded AI systems and more.

D. ORGANIZATION OF THE PAPER

The remainder of this paper is organized as follows: Section II describes the literature search methodology employed in this review, including search terms, database selection, inclusion and exclusion criteria, and the overall paper screening process. Section III presents the fundamentals of TinyML, including key concepts and definitions, the software and hardware stack, model compression techniques, deployment challenges, and commonly used evaluation metrics. Section IV investigates the role of TinyML in healthcare, focusing on low-power solutions for patient monitoring, medical diagnostics, wearable sensing, and early disease detection. Section V explores TinyML applications in the education sector, highlighting how embedded AI systems enhance learning, engagement, and hands-on classroom activities. Section VI reviews TinyML applications in the transportation domain, including autonomous vehicles, vehicular safety systems, smart traffic monitoring, and real-time driver behavior analysis. Section VII presents a detailed discussion of current limitations and open research challenges in TinyML, followed by future research opportunities in areas such as event-driven inference, federated learning, and sociotechnical integration. Finally, Section VIII summarizes the key findings of this review and outlines prospective directions for advancing the state-of-the-art in TinyML across critical application domains.

E. RELATED WORKS

In this section, we summarize and critically analyze prominent existing surveys on TinyML, highlighting their scope, methodologies, and limitations in relation to domain-specific applications such as healthcare, education, and transportation (see Table 2).

Tsoukas et al. [194] provide a broad survey of TinyML applications and trends, covering domains such as agriculture, healthcare, and smart environments. Their work

highlights hardware-software co-design strategies, deployment tools, and general challenges in the field. While the paper offers a wide overview of the TinyML ecosystem, it lacks deep domain-specific analysis and technical comparisons, making it more suitable for general orientation than focused evaluation. Abadade et al. [195] present a broad survey of the TinyML landscape, covering the full stack from model optimization to deployment on ultra-low-power hardware. They highlight key techniques such as quantization, pruning, and neural architecture search, and evaluate software tools like TensorFlow Lite Micro and TVM. While the paper offers a solid technical foundation, it focuses on general-purpose applications and does not provide in-depth analysis of domain-specific implementations, leaving room for targeted reviews in areas like healthcare, education, and transportation. Lamaakal et al. [1] present a detailed survey focused on the use of TinyML for HBA, exploring applications like activity recognition, fall detection, and emotion sensing. The paper categorizes different system architectures and discusses trade-offs between latency, energy consumption, and model accuracy in edge deployments. While the survey effectively maps the landscape of behavior-centric TinyML systems, its contributions are largely confined to healthcare and monitoring use cases. It does not extend its analysis to broader sectors like transportation or education, making it a valuable but domain-specific reference within the TinyML ecosystem. Rajapakse et al. present a comprehensive survey [197], which explores reformable TinyML systems designed to dynamically adapt to resource constraints at runtime. The authors categorize reformability across three layers model-level (e.g., weight pruning and structural reconfiguration), data-level (e.g., adaptive sampling), and hardware-level (e.g., voltage scaling and duty cycling). Their analysis spans a wide range of application domains, emphasizing the need for intelligent decision-making in environments with unpredictable computational budgets. This work provides a solid foundation for future research on adaptable and resilient TinyML systems, especially in mission-critical or intermittently powered edge deployments.

Capogrosso et al. in their work [198] provide an ML-centric taxonomy of TinyML research, emphasizing key algorithms, optimization techniques, and performance metrics tailored for resource-constrained environments. Their study highlights the trade-offs between model accuracy, latency, memory footprint, and power consumption, while also exploring applications across various domains such as smart agriculture, healthcare, and surveillance. Unlike application-specific reviews, this survey prioritizes a methodological perspective. Dutta and Bharali et al., in their paper [199], provide a broad and technical survey that explores the intersection of Tiny Machine Learning and Internet of Things technologies. The work focuses on architectural frameworks, deployment strategies, hardware-software co-design, and communication efficiency for embedding ML models into constrained IoT environments. The authors categorize TinyML-enabled IoT systems based on latency, power, and scalability demands, offering valuable insights for edge intelligence. However, the paper remains largely system-centric and does not delve into specific vertical applications such as healthcare, education, or transportation, which distinguishes it from application-focused reviews. Elhanashi et al. [200] presents a broad overview of the recent progress in the integration of TinyML with IoT systems. The authors categorize applications across several domains, including smart homes, industrial automation, and environmental sensing, while addressing major limitations such as memory constraints, limited processing capabilities, and energy efficiency. The paper offers a comparative evaluation of various TinyML frameworks and deployment tools, highlighting their suitability for different IoT scenarios. Despite its practical insights into system-level integration, the study maintains a generalized perspective, without deep exploration into vertical domains like healthcare, education, or transportation. Its main contribution lies in summarizing the evolving relationship between TinyML and IoT, offering a solid foundation for understanding implementation trade-offs and identifying areas where optimization is essential.

II. LITERATURE SEARCH STRATEGY

This section presents the methodology followed to systematically identify, select, and evaluate relevant academic publications for this review on TinyML applications in healthcare, education, and transportation. A rigorous and replicable search protocol was established to ensure the inclusion of high-quality, domain-relevant research.

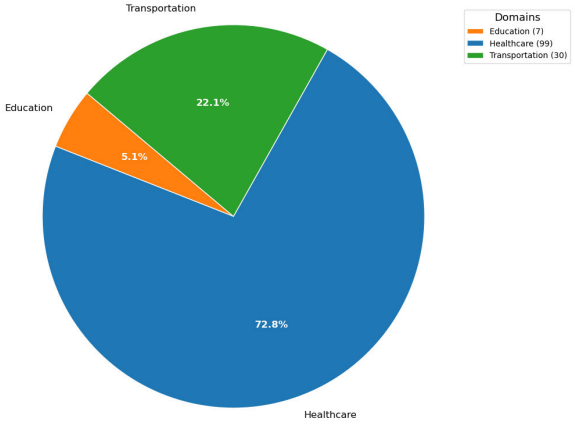

FIGURE 1. TinyML publications distribution by domain.

A. PAPER SELECTION PROCESS

The paper selection process was carried out in four stages. Initially, an extensive set of documents was retrieved using well-defined keyword queries across multiple scientific databases. This initial set was then cleaned by removing duplicates. In the screening phase, titles and abstracts were reviewed to eliminate clearly unrelated studies. The eligibility of remaining documents was determined by reviewing their

TABLE 2. Summary of related works.

Ref.	Year	Contributions
[199]	2021	Comprehensive overview of TinyML within IoT. Discusses architectural design, communication optimization, and deployment techniques. Focused on system-level perspective rather than specific use cases.
[195]	2023	Provides a full-stack overview of TinyML, focusing on model optimization, software frameworks (e.g., TensorFlow Lite Micro, TVM), and deployment. Does not deeply cover domain-specific applications.
[197]	2023	Explores reformable TinyML systems capable of runtime adaptability. Contributions span model, data, and hardware-level optimization strategies, suitable for dynamic, resource-constrained deployments.
[194]	2024	Broad survey on TinyML across multiple domains including healthcare and agriculture. Highlights hardware-software co-design, general deployment strategies, and trends, but lacks detailed domain-specific technical analysis.
[198]	2024	ML-centric classification of TinyML models. Emphasizes optimization trade-offs (latency, accuracy, power). Application coverage is secondary to algorithmic methodology.
[200]	2024	Highlights integration of TinyML in IoT. Offers domain examples (e.g., smart home, automation), discusses memory and energy constraints. Contributions are general and lack vertical depth.
[1]	2025	Focused review of TinyML for human behavior analysis. Covers applications in healthcare such as fall detection and emotion sensing. Offers system-level taxonomy but excludes sectors like education or transport.
Our Review	2025	A domain-specific review of TinyML applications across healthcare, education, and transportation. Offers a comparative analysis of existing literature, introduces taxonomies, discusses deployment metrics, model optimization strategies, and presents figures and diagrams that summarize state-of-the-art use cases. The review highlights practical challenges, open research directions, and proposes future-oriented solutions tailored to critical sectors.

full texts against a defined set of criteria, focusing on relevance to TinyML deployment in resource-constrained domains. Additional studies were discovered via backward citation analysis and citation tracking to ensure that pivotal works not captured by keyword searches were also included. In total, 136 papers were retained for the final review, comprising 99 in healthcare, 30 in transportation, and 7 in education (see Figure 1).

B. DATABASES

The literature search was conducted across a range of established academic databases known for indexing peer-reviewed and technically rigorous research. These included IEEE Xplore, ACM Digital Library, Scopus, Web of Science, SpringerLink, ScienceDirect, and Wiley Online Library. These platforms collectively provided access to literature spanning embedded systems, machine learning, educational technology, digital health, and intelligent transport systems ensuring that the search spanned both application and technical disciplines relevant to TinyML.

C. SEARCH TERMS

The keyword search was constructed using both general TinyML terms and domain-specific phrases. Boolean operators were applied to increase the precision and recall of the query results. Core terms included "TinyML" and "Tiny Machine Learning," which were combined with domain-relevant keywords such as "Healthcare," "Remote Monitoring," "Transportation," "Wearables," "Fall Detection," "Education," "Learning Analytics," "EdTech," "Smart Mobility," "Autonomous Vehicles," and "Traffic Monitoring." Additional terms such as "Embedded AI," "Microcontroller Inference," and "Low-Power Machine Learning" were also used to broaden the scope where necessary. Variants of these terms were tested to capture interdisciplinary research and use-case-specific studies involving TinyML systems operating on edge devices.

D. INCLUSION/EXCLUSION CRITERIA

The inclusion criteria were designed to focus on studies that are both recent and relevant to TinyML in embedded settings (see Table 3). Only publications dated between January 2020 and June 2025 were considered. Eligible documents had to be peer-reviewed journal articles, full conference papers, or highly cited white papers that report on TinyML deployments involving real-world or simulated implementations. Importantly, only works targeting the domains of healthcare, education, and transportation were retained. The selected studies were required to present empirical evidence, such as accuracy, model size, inference latency, energy consumption, or deployment benchmarks on microcontrollers or ultra-low-power edge devices.

In contrast, studies were excluded if they focused solely on traditional machine learning or cloud-based inference without hardware or software constraints. Papers lacking implementation details, relying solely on theoretical proposals, or those unrelated to the review's thematic scope were also filtered out. Furthermore, non-peer-reviewed sources, short

TABLE 3. Inclusion/exclusion criteria for selecting studies on TinyML in healthcare, education, and transportation.

Selection Criteria	Description
Inclusion Criteria:	
• Studies that implement TinyML solutions within healthcare, education, or transportation domains.	• Ensures alignment with the thematic scope of the review and focus on practical application.
• Peer-reviewed journal articles, conference papers, and high-quality white papers.	• Maintains academic rigor and credibility of included research.
• Publications dated from 2020 to mid-2025.	• Focuses on the most recent and relevant advancements in TinyML technologies.
• Studies involving real-world or simulated deployment on embedded or ultra-low-power devices.	• Confirms the operational feasibility and applicability of the proposed TinyML approaches.
• Research presenting measurable outcomes such as accuracy, latency, model size, or energy efficiency.	• Provides objective criteria for comparison and analysis across studies.
Exclusion Criteria:	
• Papers that reference TinyML without presenting domain-specific applications.	• Filters out studies lacking practical relevance to healthcare, education, or transportation.
• Theoretical proposals or simulations without hardware or deployment context.	• Emphasizes applied research with hardware validation or embedded testing.
• Editorials, blog posts, short abstracts, or poster-only submissions.	• Excludes informal or incomplete scientific communication lacking depth.
• Duplicate publications or extended versions without new experimental results.	• Avoids redundancy and ensures novelty within the reviewed literature.
• Non-English language papers.	• Ensures accessibility to a broader international research audience.

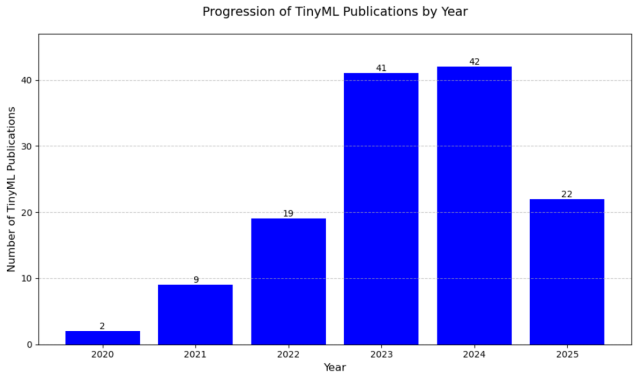

FIGURE 2. Yearly distribution of TinyML publications across sectors: Education, healthcare, and transportation.

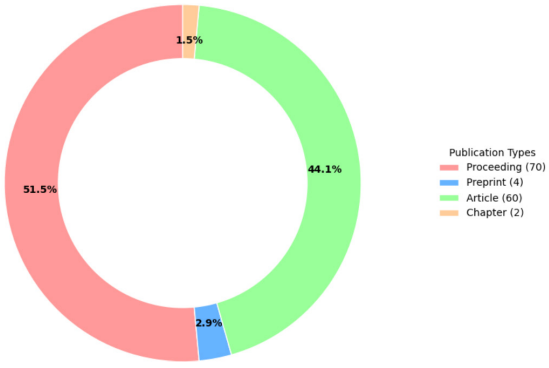

FIGURE 3. Distribution of TinyML publications across sectors (Education, Healthcare, Transportation) by publication type.

abstracts, editorials, blog posts, and design-only hardware reports without connection to TinyML were discarded.

1) SUMMARY OF SELECTED PUBLICATIONS

The final corpus of 136 papers was distributed across the three main domains as follows: 99 papers in healthcare, 30 in transportation, and 7 in education. The temporal distribution shows growth over time, with 3 papers in 2020, 9 in 2021, 19 in 2022, 41 in 2023, 42 in 2024, and 22 in early 2025 (see Figure 2). In terms of publication types, the set includes 63 conference papers, 60 journal articles, 4 preprints, 7 workshop papers, and 2 book chapters (see Figure 3).

In the education domain, publications were limited but varied: 2 from 2023, 1 from 2022, and 4 from 2025, comprising 3 conference papers and 4 journal articles. Healthcare showed strong representation with papers spanning from 2017 to 2025, including 55 conference papers, 36 journal articles, 4 preprints, 2 workshop papers, and 2 book chapters. Transportation-related TinyML research included 5 conference papers, 5 workshop papers, and 20 journal articles, distributed across 2020 (2), 2021 (4), 2022 (6), 2023 (6), 2024 (9), and 2025 (3).

III. FUNDAMENTALS OF TinyML

A. KEY CONCEPTS AND DEFINITIONS

TinyML [1], [194], [198] refers to the deployment of machine learning models directly on low-power, memory-constrained edge devices such as microcontrollers (MCUs), digital signal processors (DSPs), or embedded systems with kilobytes of RAM and limited computational throughput (see Figure 4). What distinguishes TinyML from conventional edge AI or embedded ML is its extreme resource efficiency typically operating within power budgets of less than 1 mW and memory footprints under 256 KB while maintaining real-time inference capabilities [195].

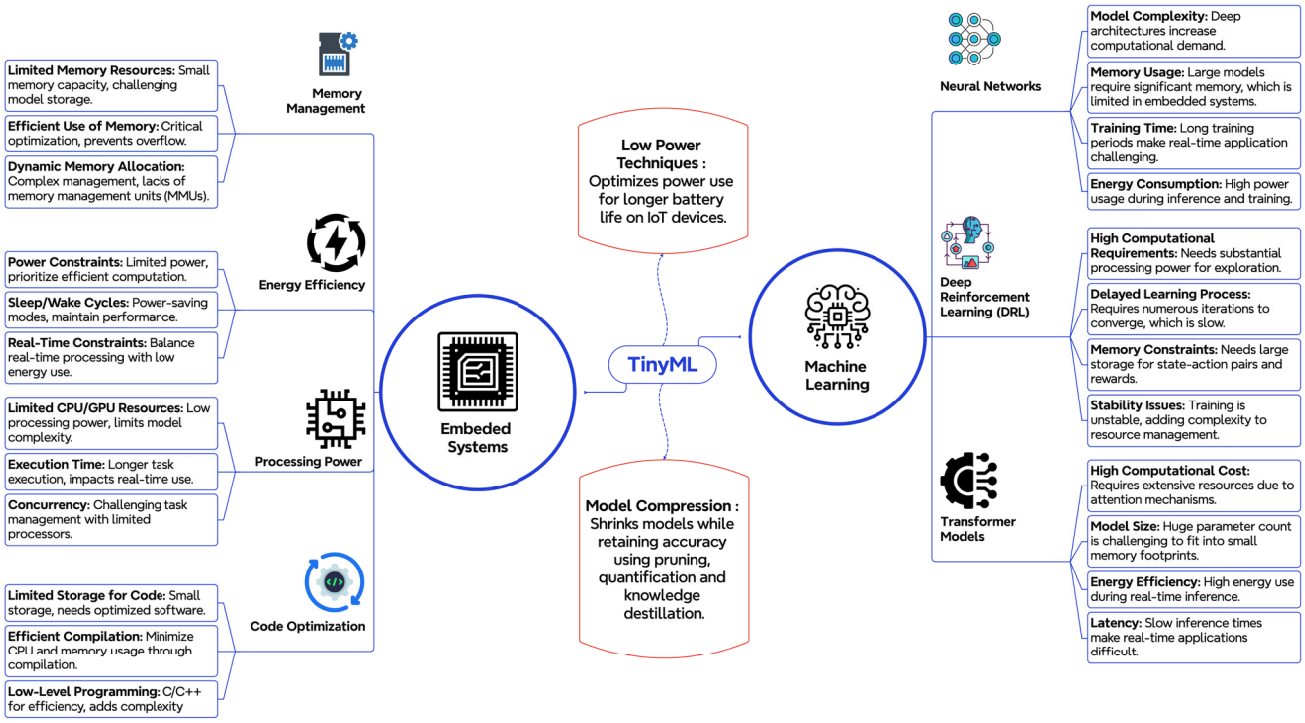

FIGURE 4. **This figure highlights the seamless integration of machine learning into embedded systems through the lens of TinyML. It underscores critical challenges in embedded systems, such as memory management, energy efficiency, processing power, and code optimization. On the machine learning side, it showcases the computational demands of neural networks, deep reinforcement learning, and transformer models, addressing considerations like model complexity, memory usage, and power consumption. The diagram emphasizes the pivotal role of model compression and low-power techniques in achieving efficient on-device inference within resource-constrained environments.**

At the core of TinyML is the principle of on-device intelligence: performing all model inference locally to minimize latency, preserve user privacy, and reduce data transmission overhead [199]. Unlike traditional ML systems that depend on cloud-based servers for prediction, TinyML devices must execute compressed and quantized versions of neural networks within stringent runtime constraints. This necessitates a design philosophy where energy-aware computing, approximate algorithms, and application-specific trade-offs are embedded from the outset [200].

Key components of a typical TinyML pipeline include sensor signal acquisition (e.g., IMU, camera, microphone), lightweight preprocessing (such as Fast Fourier Transform or Mel-spectrogram generation) [203], followed by inference using a compact ML model often a quantized neural network or decision tree. Given the low compute budget of typical MCUs (e.g., ARM Cortex-M0 to Cortex-M4 class cores running at 16–100 MHz) [1], models must be optimized not only for size but also for memory access patterns, fixed-point arithmetic compatibility, and real-time responsiveness.

TinyML also brings a new paradigm for software development that bridges embedded systems engineering with machine learning. Developers must be fluent in both domains understanding model architecture, training and quantization techniques, as well as hardware-level constraints such as SRAM layout, stack usage, and interrupt handling. Toolchains like TFLM [98], Edge Impulse [100], and CMSIS-NN [99] are foundational in this ecosystem, allowing for seamless conversion and deployment of trained models onto embedded targets.

From a systems perspective, TinyML applications span a wide spectrum, including wearable health monitors, smart agriculture sensors [204], industrial automation modules [205], autonomous vehicles [206], and environmental sensing networks [207]. Each application imposes unique constraints latency budgets, sampling rates, inference schedules, and duty cycles which influence the model design and deployment strategy. Importantly, TinyML fosters decentralization of intelligence, moving ML computations closer to the data source, thereby unlocking new possibilities for secure, context-aware [1], and ultra-low-power AI at the edge.

1) COMPARISON BETWEEN TinyML, TRADITIONAL ML, AND LLMs

TinyML, Traditional Machine Learning (ML), and Large Language Models (LLMs) [196] represent three distinct paradigms in artificial intelligence, each optimized for different constraints, applications, and deployment contexts. This section compares these paradigms across multiple dimensions such as computational footprint, data requirements,

TABLE 4. Transposed comparative analysis: TinyML vs traditional ML vs LLMs.

Feature	TinyML	Traditional ML	LLMs
Model Size	< 1MB	10–500MB	10–300GB
Training Complexity	Low (offloaded to cloud/PC)	Moderate (depends on algorithm)	Extremely High (requires GPU clusters)
Inference Cost	Ultra-low power (<10mW)	Moderate (mobile/server CPUs)	Very high (multi-GPU/TPU)
Internet Dependency	*None*	Optional	*Required (often)*
Data Requirement	KB–MB	MB–GB	TBs–PBs
Scalability	Low (single-device specific)	Medium (model reuse across domains)	High (general-purpose NLP understanding)
Latency Sensitivity	✓Critical	Depends on app	× High latency tolerated in some apps
Hardware Examples	Arduino Nano, ESP32, STM32	Laptop, Jetson Nano, RPi 4	A100, H100, TPU v4, Lambda Labs
Explainability Tools	Very limited (e.g., rule-based XAI)	✓Available	✓Ongoing (Chain-of-Thought, attention maps)
Deployment Footprint	*<100KB code and weights*	MBs with dynamic libraries	100+ GBs including embeddings
Power Usage	*<1mW to 50mW*	1W–30W	>200W
Security/Privacy	High (no cloud needed)	Medium (depends on app)	Low (requires cloud/service trust)
Preprocessing Needs	Minimal (basic filtering)	Moderate (e.g., feature extraction)	Complex (tokenization, embedding)
Update Frequency	Rare (infrequent model updates)	Moderate	Frequent (continuous fine-tuning)
Cost per Device	$1–$10 (MCUs)	$50–$500	$10K–$100K+ (for deployment)
Hardware Requirements	Microcontrollers (MCUs), Edge SoCs	Training done offline, general CPUs/GPUs	High-end GPU clusters
Latency	✓Real-time	✓Offline Inference	× Higher acceptable latency
Notes	Compact, low-power, embedded-focused	Balanced trade-off, versatile	Complex, general-purpose, resource-heavy

latency, scalability, hardware compatibility, and training methodologies.

While traditional ML focuses on flexibility and performance on general-purpose hardware, TinyML targets energy-efficient, real-time inference on ultra-constrained devices, often without network connectivity. In contrast, LLMs emphasize vast pre-trained language knowledge and contextual understanding but require massive computational and storage infrastructure for both training and inference.

2) DIFFERENCES BETWEEN TRADITIONAL ML AND TinyML WORKFLOWS

While TinyML builds upon many foundational concepts from traditional ML, its development workflow diverges significantly due to the unique constraints and requirements of embedded systems. As illustrated in Figure 5, the standard ML lifecycle encompasses three main stages: data collection, model development, and deployment. TinyML retains these stages but introduces critical adaptations at nearly every phase to accommodate the strict limitations of microcontroller-based systems.

In a traditional ML pipeline, model development typically begins with large-scale datasets, which are processed and fed into high-capacity architectures trained on GPUs or cloud-based infrastructure. Once trained and evaluated, these models are deployed in environments with ample compute and memory resources. However, TinyML targets ultra-constrained hardware platforms often featuring less than 256 KB RAM, low clock frequencies (e.g., 16–80 MHz), and energy budgets in the milliwatt range. As a result, the traditional design–train–deploy loop is insufficient, and a new optimization-centric loop becomes necessary [1].

One major difference lies in the optimization and compression stage, which is essential for TinyML but often peripheral in classical ML workflows. Techniques such as post-training quantization, weight pruning, and knowledge distillation are not just optional enhancements but prerequisites for feasibility [194], [199]. These techniques are applied before or after training to reduce model size, memory usage, and computational complexity. Additionally, TinyML pipelines must account for hardware-aware training strategies such as quantization-aware training (QAT) [196], which ensures

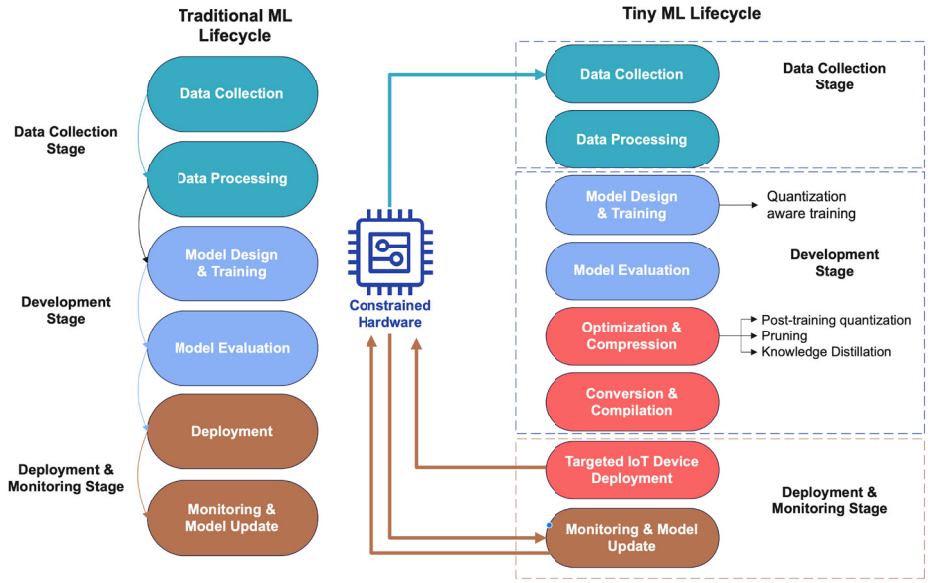

FIGURE 5. Comparison between traditional ML and TinyML lifecycles.

that accuracy is preserved when models are converted to fixed-point integer formats.

Another fundamental divergence is the conversion and compilation phase, which transforms the optimized model into a binary executable tailored for a specific microcontroller's architecture. This step includes toolchains like TFLM, TVM, or CMSIS-NN, which generate memory-efficient, inference-ready code. These frameworks also handle low-level optimizations, such as kernel fusion, memory tiling, and SRAM-aware buffer allocation challenges rarely addressed in traditional ML deployments [197].

Deployment in TinyML is no longer just a matter of uploading a model to a cloud endpoint or mobile device. Instead, it requires flashing compiled binaries onto embedded devices, often using interfaces like JTAG, USB serial, or OTA (over-the-air) updates [208]. Moreover, model monitoring in TinyML must operate without cloud telemetry due to bandwidth or privacy limitations. Therefore, lightweight update mechanisms and local evaluation tools become critical to ensure continuous model reliability.

Lastly, model evaluation in TinyML cannot rely solely on metrics like accuracy or loss. Energy consumption per inference, memory footprint, latency, and duty cycle behavior are equally important and must be integrated into the evaluation loop. These system-level metrics define real-world usability and determine whether a model can function effectively in an environment with intermittent power, constrained compute, or real-time requirements [198].

Overall, while the overarching stages between ML and TinyML workflows may appear similar, their internal processes, evaluation criteria, and success metrics are drastically different. TinyML development requires a co-design approach between algorithm, software, and hardware, shifting the paradigm from model-centric to system-aware AI engineering.

3) ADVANTAGES OF TinyML

TinyML provides several significant benefits that distinguish it from traditional machine learning and large-scale AI systems. These advantages are especially impactful in edge computing, low-power environments, and latency-sensitive applications.

1) Ultra-Low Power Consumption and Battery Efficiency: TinyML models are specifically designed to operate within the stringent power constraints of microcontrollers (e.g., ARM Cortex-M series), often consuming less than 1 mW during inference. This ultra-low power footprint enables deployment in battery-powered or energy-harvesting systems, making them ideal for remote sensing, wearable devices, and long-term field operations without frequent maintenance [1], [194], [195]. For instance, a vibration anomaly detector using TinyML can run on a coin-cell battery for months or even years. This efficiency opens up opportunities in applications where continuous cloud connectivity is infeasible or undesirable due to cost and power limitations.
2) On-Device Inference and Real-Time Processing [209]: Unlike traditional ML or cloud-based models, TinyML performs inference directly on the embedded device. This eliminates round-trip delays introduced by communication with external servers, thus significantly reducing latency. Real-time response is critical in domains like medical monitoring (e.g., detecting arrhythmia), industrial safety systems, or autonomous drones. With inference times often under 10–100 ms, TinyML systems can make split-second decisions locally.

3) Enhanced Data Privacy and Security: TinyML processes data locally, meaning sensitive information (e.g., biometric signals, voice, location) never leaves the device. This inherent design minimizes the risk of data interception or misuse and aligns well with regulatory requirements such as GDPR and HIPAA [210]. For example, a fall detection system for elderly care can operate entirely on-device, without transmitting private movement patterns or health data to cloud servers. As cyber-physical systems grow.
4) Lower Cost and Accessibility: The microcontrollers used in TinyML applications such as Arduino Nano 33 BLE Sense or ESP32 are inexpensive, often costing less than $10. This low hardware cost, combined with open-source software stacks like TFLM or CMSIS-NN, makes it accessible for developers, educators, and small-scale deployments [200]. In educational settings, students can deploy gesture recognition or environmental monitoring models with minimal investment, encouraging democratization of embedded AI.
5) Network Independence and Offline Operation: TinyML models are particularly well-suited for environments with limited or no internet access, such as rural areas, underwater sensor networks, or in-flight systems. The capability to function entirely offline allows devices to operate reliably in infrastructure-poor regions. Applications include wildlife monitoring with sound recognition models, offline agriculture diagnostics using image classification, or emergency alert systems where uptime is critical regardless of connectivity.
6) Low Latency with High Reliability: Due to proximity between data acquisition and processing, TinyML architectures inherently avoid delays caused by server load, bandwidth constraints, or intermittent connectivity [1]. This makes them highly suitable for mission-critical systems where timing is essential. For example, an autonomous AAV equipped with TinyML-based collision avoidance logic can react in real-time without querying cloud-based vision APIs.
7) Optimized for Customization and Application-Specific Design: TinyML workflows allow for the creation of task-specific models optimized for a given sensor, use case, or data modality. Developers can fine-tune quantized CNNs, RNNs, or decision trees for specific memory footprints or latency budgets [197], [199]. Such flexibility enables targeted deployment in domains ranging from smart farming to asset tracking, without needing a generalized and resource-heavy model.
8) Scalable for Mass Deployment: The small model size and ultra-low energy demand make TinyML suitable for mass-scale, distributed edge AI deployments such as thousands of smart meters, environmental monitors, or industrial sensors. The small binary footprint ($<$100 KB) allows firmware-over-the-air (FOTA) [211] updates and over-the-network distribution, enabling scalability and centralized management across devices.

B. HARDWARE AND SOFTWARE STACK

1) TinyML SOFTWARE ECOSYSTEM

This subsection highlights the key software frameworks and toolchains tailored specifically for deploying machine learning models in constrained hardware environments.

The uTVM framework [97] extends the capabilities of the TVM compiler to support bare-metal microcontroller platforms, eliminating the need for an operating system. It employs loop-level and operator-specific optimizations to maximize efficiency and includes AutoTVM for automated performance tuning. TensorFlow Lite Micro [98], a stripped-down variant of TensorFlow Lite, is designed for microcontrollers like Cortex-M and ESP32. It excludes dynamic memory allocation and complex operators to ensure lightweight, memory-conscious deployment. CMSIS-NN [99], developed by ARM, is a software library that translates trained models from platforms such as TensorFlow and PyTorch into efficient C++ implementations for Cortex-M cores, focusing on performance and portability.

Edge Impulse's EON Compiler [100] provides an end-to-end pipeline that transforms machine learning models into minimalistic, memory-efficient C++ code by pruning redundant operations and fusing operators where possible. uTensor [101], an open-source solution for Mbed-enabled devices, takes TensorFlow Lite models and converts them into statically compiled C++ code with a focus on aggressive quantization and execution speed. STM32Cube.AI [102], offered by STMicroelectronics, facilitates the translation of neural network models into highly optimized C code for STM32 microcontrollers, including features like quantization-aware mapping and kernel fusion to enhance runtime efficiency.

NanoEdge AI Studio [103], also from STMicroelectronics, is an AutoML-based software suite tailored for anomaly detection and predictive maintenance. It produces compact models specifically tuned for STM32-based hardware, balancing inference speed and memory requirements. Eloquent MicroML and TinyML [104] focus on translating classical models such as decision trees and support vector machines into lightweight C code that is Arduino-compatible and suitable for real-time applications on low-end MCUs. EmbML [105] targets model conversion from tools like Weka and Scikit-Learn into memory-optimized C++ code for embedded deployment, leveraging quantization to minimize memory usage.

Sklearn Porter [106] serves as a model conversion utility that exports Scikit-Learn models to multiple programming languages C, Java, and JavaScript thus enabling broad deployment possibilities on embedded platforms. Finally, SONIC and TAILS® [107] are highly specialized inference engines designed for energy-constrained and intermittently powered devices, such as those based on MSP430 microcontrollers. SONIC focuses on optimizing loop operations and managing sparse computation patterns,

TABLE 5. Noteworthy TinyML softwares for MCUs.

Softwares	Supported Models	Training Libraries Supported	Compatible Platforms	Free	Open Source
uTVM [97]	NN	Keras, PyTorch, TFL	ARM Cortex-M	✓	✓
TFLM [98]	NN	TFL	Espressif ESP32, ARM Cortex-M, Himax WE-I Plus	✓	✓
CMSIS-NN [99]	NN	PyTorch, Caffe, TFL	ARM Cortex-M	✓	✓
EON Compiler [100]	k-means, NN, Regressors	Scikit-Learn, TFL	TENSAI SoC, ARM Cortex-A, TI CC1352P, Himax WE-I Plus, Espressif ESP32, ARM Cortex-M,	✗	✓
STM32Cube.AI [102]	SVM, Regressors, NN, NB, kNN, DT, RF, K-means	Keras, Microsoft Cognitive Toolkit, ConvnetJS, PyTorch, MATLAB, Scikit-Learn, Lasagne, TFL, Caffe	ARM Cortex-M (STM32 series)	✓	✗
NanoEdge AI Studio [103]	Unsupervised learning	-	ARM Cortex-M (STM32 series)	✗	✗
uTensor [101]	NN	TFL	ARM Cortex-M (Mbed-enabled)	✓	✓
EloquentML [104]	Regressors, RF, k-means, SEFR, DT, NN, SVM, RVM, XGBoost, NB	Scikit-Learn, TFL	AVR RISC, ARM Cortex-M, Espressif ESP8266, Espressif ESP32	✓	✓
Sklearn Porter [106]	RF, SVM, AdaBoost, NN, DT, NB	Scikit-Learn	-	✓	✓
SONIC,TAILS® [107]	NN	TFL	TI MSP430	✓	✓
EmbeML [105]	SVM, Regressors, NN, DT	Weka, Scikit-Learn	ARM Cortex-M, AVR RISC	✓	✓

while TAILS® improves energy efficiency and inference throughput through hardware acceleration with the LEA (Low Energy Accelerator).

Table 5 highlights notable TinyML software platforms, detailing their supported models, compatible training libraries, and the microcontroller platforms they can be deployed on, along with their availability and open-source status.

2) HARDWARE ARCHITECTURES FOR TinyML

The success of TinyML solutions is closely tied to the capabilities of the underlying hardware. These hardware platforms are engineered to function under severe constraints related to memory, processing power, and energy consumption, yet still enable reliable inference for real-world applications (refer to Table 6 for comparative details). This section provides an in-depth look at several prominent TinyML hardware platforms and their use cases.

The Arduino Nano 33 BLE Sense [108] is a compact microcontroller that integrates various onboard sensors, making it especially suitable for tasks such as gesture detection [110], wearable sensing systems [109], and small-scale speech recognition applications [111]. In contrast, the NVIDIA Jetson Nano [112] delivers higher computational power through its quad-core ARM Cortex-A57 processor, making it ideal for robotics, edge-based vision systems, and other AI-driven use cases requiring real-time inference. For ultra-low-power scenarios, the Adafruit Feather M0 [113] provides a minimalist yet effective option, especially in long-term IoT deployments. The Google Coral Dev Board [114], equipped with an Edge TPU accelerator, is designed for fast, energy-efficient inference, particularly in tasks involving computer vision and object detection. Microcontroller boards like the Seeed Studio Wio Terminal [115] and Adafruit CLUE [116] bundle extensive sensors and are often used in education and prototyping for IoT and embedded learning.

Boards such as the PYNQ-Z2 [117] and BeagleBone Black [118] cater to more demanding embedded AI tasks, offering support for programmable logic and general-purpose processing, making them suitable for hybrid hardware-software co-processing. The Syntiant TinyML hardware [119], with its ultra-low-power neural decision processor, is tailored for continuous audio monitoring tasks like wake-word detection in always-on listening environments. The ESP32-S3-DevKitC [120] facilitates fast development of Wi-Fi and Bluetooth-enabled IoT solutions, while the Himax WE-I [121] focuses on high-efficiency vision and audio processing, supporting real-time applications such as motion-triggered imaging or keyword spotting. Sony's Spresense platform [122] brings advanced signal processing with GPS functionality, allowing for spatial awareness applications such as directional sound localization and geofencing. The Raspberry Pi 4 Model B [123] remains a staple in the edge AI ecosystem due to its balance of compute, connectivity, and flexibility powering solutions ranging from smart surveillance to home automation. Designed specifically for TinyML, the Arducam Pico4ML-BLE [124] combines BLE and camera support to enable compact vision applications. The Portenta H7 [125], with its dual-core architecture, is capable of supporting more demanding workloads including real-time object tracking and multi-threaded robotics control. ESP-EYE [126] is another specialized board aimed at embedded image and audio recognition, commonly integrated into smart locks and surveillance cameras. SparkFun's RedBoard Artemis [127] excels in energy-constrained contexts, such as mobile environmental monitoring and wearable health devices. Finally, the Raspberry Pi Pico [128] offers a budget-friendly entry point for basic ML inference tasks, sensor fusion, and educational prototyping.

TABLE 6. Hardware architectures for TinyML.

Hardware	Flash Memory	SRAM	CPU	MCU	CPU Clock	Dims
Arduino Nano 33 BLE Sense [108]	256 KB	1 MB	Cortex-M4	nRF52840	64 MHz	45x18 mm
Jetson Nano [112]	×	4 GB LPDDR4	Quad-core ARM A57	×	1.43 GHz	70x45 mm
Adafruit Feather M0 [113]	256 KB	32 KB	ARM Cortex-M0	ATSAMD21G18	48 MHz	50.8x22.9 mm
Google Coral Dev Board [114]	16 GB eMMC	1 GB LPDDR4	Quad-core Cortex-A53	Edge TPU	1.5 GHz	88x60 mm
Seeed Studio Wio Terminal [115]	4 MB	192 KB	ARM Cortex-M4F	ATSAMD51P19	120 MHz	72x57 mm
Adafruit CLUE [116]	256 KB	32 KB	ARM Cortex-M4	nRF52840	64 MHz	40x20 mm
PYNQ-Z2 [117]	512 MB DDR3	512 MB	ARM Cortex-A9 ×2 cores	Zynq-7000	650 MHz	90x60 mm
BeagleBone Black [118]	4 GB eMMC	512 MB DDR3	ARM Cortex-A8	AM335x	1 GHz	86x53 mm
Syntiant TinyML [119]	256 KB	32 KB	Cortex-M0+	NDP101	48 MHz	24x28 mm
ESP32-S3-DevKitC [120]	×	512 KB	32 bit Xtensa dual core	ESP32-S3-WROOM-1	240 MHz	×
Himax WE-I [121]	2 MB	2 MB	ARC 32-bit DSP	HX6537-A	400 MHz	40x40 mm
Sony Spresense [122]	8 MB	1.5 MB	ARM Cortex-M4F ×6 cores	CXD5602	156 MHz	50x20.6 mm
Raspberry Pi 4 Model B [123]	×	2 GB, 4 GB, or 8 GB SDRAM	Quad-core Cortex-A72	BCM2711	1.5 GHz	56.5x86.6 mm
Arducam Pico4ML-BLE [124]	2 MB	264 KB	Dual-core ARM Cortex-M0+	RP2040	133 MHz	51x21 mm
Portenta H7 [125]	16 MB NOR Flash	8 MB SDRAM	Cortex M4, Cortex M7	STM32H747	240 MHz , 480 MHz	62x25 mm
ESP-EYE [126]	4 MB	8 MB	32 bit Xtensa dual core	ESP32	240 MHz	41x21 mm
SparkFun RedBoard Artemis [127]	1 MB	384 KB	ARM Cortex-M4F	Apollo3	48 MHz	53x23 mm
Raspberry Pi 4 Pico [128]	2 MB	264 KB	Dual-core ARM Cortex-M0+	RP2040	up to 133 MHz	51x21 mm

Altogether, these hardware platforms span a broad spectrum of performance and power profiles, enabling scalable and efficient TinyML applications in fields ranging from healthcare and security to smart cities and industrial automation.

C. MODEL OPTIMIZATION

Efficient deployment of machine learning models on memory-constrained and energy-limited devices requires sophisticated compression techniques. These methods reduce the model's footprint in terms of size, latency, and power consumption, while preserving acceptable levels of accuracy. The most prominent techniques include knowledge distillation, quantization, pruning, and NAS. These are often combined to achieve optimal performance on TinyML platforms (see Table 7). To ground these methods in practice, we illustrate each with representative TinyML tasks: keyword spotting (KWS) on Google Speech Commands, person detection via Visual Wake Words (VWW), and acoustic anomaly detection (AD) in industrial settings (see, e.g., MLPerf Tiny benchmark tasks).

1) KNOWLEDGE DISTILLATION

Knowledge distillation [187] involves training a compact *student model* to imitate the behavior of a more complex *teacher model*. The student does not merely learn from the ground-truth labels but also from the softened output distributions of the teacher, which provide richer information about class relationships. The overall loss function is a weighted sum of task loss and distillation loss, defined as:

$$\mathcal{L} = (1-\alpha)\cdot\mathcal{L}_{\text{task}} + \alpha\cdot T^2\cdot \text{KL}(P_t \parallel P_s) \quad (1)$$

where $P_t = \text{softmax}(z_t/T)$ and $P_s = \text{softmax}(z_s/T)$ are the temperature-scaled probability distributions from the teacher and student models, T is the temperature factor, and α controls the trade-off between supervised and distillation losses. This method allows compact models to inherit performance characteristics from larger ones, making it ideal for TinyML inference tasks.

a: CASE STUDY (VWW AND KWS)

In MCU-class vision, MicroNets use distillation from a MobileNetV2 teacher (temperature $T = 4$, coefficient 0.5) while searching student architectures, improving accuracy within tight SRAM/latency budgets on STM32 boards [236]. For audio KWS, distillation has been used to replace heavier speech front-ends with lightweight encoders while maintaining accuracy suitable for on-device inference [244]. These examples show that KD is most effective when (i) the student is quantized and (ii) the teacher exposes class similarities the student would otherwise miss.

2) QUANTIZATION

Quantization [188] compresses model parameters by reducing the bit-width used to store weights and activations. This technique substantially decreases memory usage and accelerates computations. Several quantization strategies exist:

1) *Low-precision floating-point formats* like FP16 or Bfloat16 reduce memory overhead while retaining sufficient dynamic range. These use 16 bits to encode floating-point numbers, defined as:

$$x = (-1)^s \times m \times 2^e \quad (2)$$

where s, m, and e represent the sign bit, mantissa, and exponent, respectively.

2) *Fixed-point representation* [192] stores numbers using fixed integer and fractional segments. It avoids floating-point operations by computing values as:

$$x = \text{integer part} + \frac{\text{fractional part}}{2^n} \tag{3}$$

where n is the number of fractional bits. This representation is hardware-friendly and particularly well-suited for microcontroller-based deployments.

3) *Binarization* [192] restricts weights to two possible values, typically -1 and $+1$, using:

$$\hat{w} = \text{sign}(w) \tag{4}$$

4) *Ternarization* [193] further expands the quantized value set to $\{-1, 0, +1\}$, using a threshold Δ:

$$\hat{w} = \begin{cases} 1 & \text{if } w > \Delta \\ 0 & \text{if } |w| \leq \Delta \\ -1 & \text{if } w < -\Delta \end{cases} \tag{5}$$

These extreme forms of quantization significantly reduce both memory and compute demands.

5) *Logarithmic quantization* maps weights to powers of two, allowing multiplications to be replaced by inexpensive bit-shift operations:

$$\hat{w} = \text{round}(\log_2(w)) \tag{6}$$

a: CASE STUDY (KWS, VWW, AD)

On MCUs, 8-bit per-channel integer quantization is the de facto deployment path in TensorFlow Lite Micro (TFLM), typically yielding ∼4× smaller models and 2–3× CPU speedups with minimal accuracy loss; quantization-aware training further closes the gap to FP32 [239], [243]. For KWS on STM32F746, the MicroNet-KWS-M (8-bit) model attains 95.8% accuracy on Google Speech Commands with ∼103 KB peak SRAM and ∼0.187 s latency; the smaller MicroNet-KWS-S uses ∼53 KB SRAM at 95.3% accuracy [236]. For VWW person detection, 8-bit MicroNet variants span 69.5–285 KB SRAM with 78–88% accuracy, enabling developers to trade memory for accuracy within a fixed energy budget [236]. For industrial AD, 8-bit MicroNet-AD-S reaches 95.35% AUC at ∼114 KB SRAM and sub-0.2 s latency on STM32F746 [236], aligning with MLPerf Tiny AD targets [237]. Sub-byte (4)-bit) quantization can expand capacity under the same SRAM, at the cost of higher kernel emulation overhead; QAT helps keep accuracy competitive [236], [239].

3) PRUNING

Pruning [189] eliminates weights or neurons that contribute minimally to model output, creating a sparse architecture that is faster and more energy-efficient. Two main approaches are:

Structured pruning: removes entire layers, neurons, or filters, simplifying hardware parallelization.

Unstructured pruning: removes individual weights below a predefined threshold:

$$w_p = \begin{cases} 0, & \text{if } |w| < \tau \\ w, & \text{otherwise} \end{cases} \tag{7}$$

where τ is the pruning threshold.

a: CASE STUDY (VWW / PLATFORM-AWARE PRUNING)

Platform-aware structured pruning (e.g., channel/filter removal) maps cleanly to MCU kernels and yields direct latency and energy reductions measured on the target device. Methods like NetAdapt iteratively prune with on-device latency/energy measurements to meet resource budgets without large accuracy drops [240]. Classic "deep compression" pipelines (pruning + quantization + coding) show 35–49× storage reductions on ImageNet models [241]. On Arm Cortex-M MCUs, optimized kernels (CMSIS-NN) further translate structured sparsity and lower precision into up to ∼4.6× speed and ∼4.9× energy improvements at inference [99].

4) NEURAL ARCHITECTURE SEARCH (NAS)

NAS [190] automates the discovery of optimal model topologies tailored to target constraints. It explores combinations of depth, width, and layer types by optimizing for accuracy or efficiency:

$$\mathcal{A}^* = \arg\max_{\mathcal{A} \in \mathcal{S}} \text{Acc}(\mathcal{A}) \tag{8}$$

where $\mathcal{A}$ is a candidate architecture in search space $\mathcal{S}$, and $\text{Acc}(\mathcal{A})$ is its performance score.

Case Study (Image Classification, KWS, AD): MCUNet jointly designs both the MCU-ready architecture (TinyNAS) and a lightweight inference engine (TinyEngine), enabling ImageNet-scale inference under MCU memory limits [238]. For the MLPerf Tiny tasks, MicroNets use differentiable NAS plus quantization (and KD on VWW) to produce deployable models with measured SRAM, latency, and energy on STM32 boards—e.g., KWS models at 95–96.5% accuracy with 53–209 KB SRAM; VWW at 78–88% with 69–285 KB SRAM; and AD at 95–97% AUC with 114–384 KB SRAM [236].

a: PUTTING IT TOGETHER

In practical TinyML pipelines, combinations of these methods are often employed. For example, quantization-aware training with per-channel int8 quantization is commonly applied after platform-aware structured pruning, optionally followed by NAS-driven fine-tuning or distillation from a floating-point teacher. Such compound strategies routinely yield >90% reductions in model size and compute, while meeting MLPerf Tiny accuracy targets for KWS (≥90%), VWW (≥80%), and AD (≥0.85 AUC) [237]. This makes sophisticated neural models deployable on low-cost microcontrollers with sub-100 KB SRAM footprints when the task permits [99], [191], [236], [245].

TABLE 7. Comprehensive overview of TinyML model compression techniques.

Compression Method	*Compression Categories*	*Typical Compression Ratio*	*Speedup*	*Memory Footprint*	*Deployment Insights*
Knowledge Distillation	Offline (Teacher–Student), Soft Targets, Feature Matching	2x–10x (varies by architecture)	Depends on student model design	Small to moderate, based on student capacity	Effective for custom student design; supports low-latency inference on MCUs with fewer layers
Quantization	Post-training, Quantization-aware training, Weight-only, Activation-only, Mixed-precision, Dynamic, Static, Logarithmic, Binarization, Ternarization	2x–32x depending on precision (e.g., INT8, INT4, binary)	1.5x–4x for INT8; higher for binary networks	Significantly reduced; 50–90% memory savings	Widely supported by CMSIS-NN, TFLite Micro; effective on Cortex-M, ESP32 platforms
Pruning	Structured (channels, filters), Unstructured (weights), Global, Layer-wise, Dynamic, Lottery Ticket Hypothesis	1.5x–16x depending on sparsity	1.2x–3x (limited by hardware sparsity support)	Moderate to large reduction in weights	Works well with quantization; unstructured pruning needs sparse-aware execution libraries
Neural Architecture Search	Hardware-aware, Proxyless NAS, Differentiable, Evolutionary, Reinforcement learning-based	Task-dependent; can exceed 10x reduction	Depends on search space; latency-aware NAS can improve inference speed	Optimized for target platform, reducing both compute and memory	Produces platform-specific models; tools like MNasNet, FBNet optimize for edge latency
Low-rank Approximation	Matrix factorization, Tensor decomposition	2x–4x typically	1.5x–2x depending on layer factorization	Reduced for large FC or Conv layers	Best for large layers; may degrade accuracy without retraining
Weight Sharing	Hash-based, Cluster-based quantization	2x–8x depending on group size	Slight, depending on lookup overhead	Reduces storage; compute cost may increase slightly	Effective for offline compression; needs custom lookup execution at runtime
Tensor Decomposition	CP, Tucker, SVD on weight tensors	Up to 10x for large layers	Layer-specific; often used in vision models	Reduced rank yields fewer parameters	Works best for ConvNets; model re-training improves effectiveness
Binarization	BinaryConnect, BinaryNet	Up to 32x	2x–8x depending on support for binary operations	Extremely small; 1-bit weights and activations	Excellent for logic-based MCUs; limited accuracy in complex tasks
Ternarization	TernaryNet, Trained Ternary Quantization	16x–24x typically	1.5x–3x	Very low memory, minimal model overhead	Good balance of compression and accuracy retention for edge

D. EVALUATION METRICS IN TinyML

TinyML systems are typically evaluated not just by model performance, but also by their efficiency, responsiveness, and energy requirements, due to the severe constraints of microcontrollers and edge devices. Below is a comprehensive overview of the key evaluation metrics commonly used in TinyML contexts.

1) *Accuracy, Precision, Recall, and F1-Score*: These metrics are fundamental for classification tasks in TinyML applications like gesture recognition or voice detection.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{11}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

These metrics allow evaluation of performance trade-offs, particularly in highly imbalanced datasets common in sensor-triggered tasks.

2) *Mean Absolute Error (MAE) and Mean Squared Error (MSE)*: Used primarily in regression-oriented TinyML applications such as environmental monitoring or biomedical signal estimation.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{13}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{14}$$

MAE is more interpretable while MSE penalizes larger errors more severely an important consideration when predicting physical measurements.

3) *Model Size (KB/MB)*: Model size directly determines whether the model fits in the microcontroller's Flash and SRAM. A typical TinyML model must be under 100–200 KB to fit on devices like the Arduino Nano 33 BLE Sense or STM32. This includes weights, activations, and quantized parameters.

4) *Inference Latency*: Latency measures how quickly a model returns a prediction after receiving input, which is critical for real-time applications like fall detection or

object tracking.

$$\text{Latency} = t_{\text{end}} - t_{\text{start}} \tag{15}$$

Microcontrollers generally target latency below 100 ms to ensure responsive user interactions and control decisions.

5) *Energy per Inference (EPI)*: Energy is a key constraint in TinyML deployments, especially in battery-powered or energy-harvesting settings.

$$E_{\text{inf}} = P \cdot t_{\text{inf}} \tag{16}$$

where P is power in watts and t_{inf} is the inference time in seconds. Reducing EPI extends device lifetime and supports sustainability.

6) *Throughput (Inferences per Second)*: Throughput refers to the number of inferences the system can perform per unit time. It complements latency for high-speed applications:

$$\text{Throughput} = \frac{1}{\text{Latency}} \tag{17}$$

Higher throughput indicates better scalability for batch processing or continuous signal monitoring.

7) *Memory Footprint (SRAM/Flash)*: This includes all memory consumed by weights, model structure, buffers, and intermediate activations. It is often separately reported for Flash (code/weights) and SRAM (runtime buffers). Tools like TensorFlow Lite Micro analyzer or MAP files from compilers are used for measurement.

8) *Accuracy vs. Efficiency Trade-off*: TinyML systems require balancing model performance with constraints on energy, memory, and compute. Pareto curves and trade-off ratios (e.g., accuracy per kilobyte) are increasingly used to guide model design in constrained scenarios.

E. DEPLOYMENT CHALLENGES

Despite the promising capabilities of TinyML, transitioning from development to real-world deployment involves several non-trivial challenges. These challenges arise from limitations in hardware, variability in data quality, ecosystem fragmentation, and a lack of robust deployment pipelines. This subsection provides a comprehensive analysis of the core barriers that hinder scalable and sustainable TinyML deployment.

1) Memory and Storage Constraints: Most TinyML models are deployed on microcontrollers with extremely limited SRAM (often between 8–256 KB) and Flash memory (256 KB to 1 MB). These constraints not only limit the size of the model and supporting software but also restrict runtime operations such as input buffering, preprocessing, and intermediate tensor storage [194]. For instance, running a CNN on an STM32 or ESP32 device requires aggressive quantization and layer fusion, and even then, performance may degrade under tight memory budgets. Efficient use of CMSIS-NN, operator-level memory reuse, and static memory allocation are often necessary but increase complexity in deployment.

2) Model Inference Latency and Real-Time Guarantees: Ensuring low-latency inference (e.g., under 50–100 ms) is crucial in applications like fall detection, gesture control, or predictive maintenance. However, inference time is highly sensitive to model architecture, data format, and hardware clock speed. Limited parallelism on MCUs typically running at 48–120 MHz means that even moderately sized models may not meet real-time constraints [195]. Furthermore, variations in sensor sampling rate or preprocessing complexity can introduce unpredictable latency, violating real-time response guarantees. Optimizing runtime performance often requires manual tuning of pipeline stages and careful selection of activation functions, buffer reuse strategies, and fused operations.

3) Lack of End-to-End Toolchains for Edge Deployment: The TinyML software ecosystem remains fragmented, with different tools for model training (TensorFlow, PyTorch), conversion (TFLite Converter, ONNX), optimization (TVM, STM32Cube.AI), and deployment (Arduino IDE, Mbed, PlatformIO). This fragmentation often results in incompatible workflows, lack of automation, and time-consuming debugging [1]. Developers must often resort to manual code translation or reverse engineering of generated code, especially when optimizing for specific chipsets or integrating third-party sensor drivers. End-to-end deployment solutions remain immature compared to cloud-based ML pipelines.

4) Limited On-Device Debugging and Monitoring Capabilities: Unlike traditional software systems, TinyML deployments have limited visibility into runtime states due to the absence of full OS stacks or hardware debuggers. Debugging tools like JTAG or SWD provide only low-level access, and real-time logging is often constrained by memory or communication bandwidth [197]. This makes it difficult to detect issues such as input anomalies, silent model failures, memory overflows, or data drift. Moreover, since inference is performed in a highly constrained stack, even small changes in data flow or quantization can result in model divergence without clear feedback.

5) Sensor Noise and Data Drift in Field Environments: Models trained under controlled conditions often perform poorly when deployed in variable environments with real-world noise, occlusion, and sensor drift. For example, an accelerometer-based model trained for fall detection may misclassify walking downstairs if the sensor orientation changes or the user's gait differs significantly from training samples [198]. Lack of domain adaptation or online learning support further aggravates this issue. Although some research has explored on-device domain generalization and continual learning, these methods remain computationally expensive or require retraining pipelines that are not feasible on MCUs.

6) Firmware Integration and Update Complexity: Integrating a TinyML model with embedded firmware typically

requires careful coordination of sensor drivers, communication protocols (e.g., BLE, UART), and operating loops. Memory alignment, DMA configuration, and ISR handling must be optimized manually to avoid conflicts [199]. Updating deployed models through over-the-air firmware updates (FOTA) is still uncommon in resource-constrained environments due to limited Flash memory and bootloader complexity. Secure updates add another layer of cryptographic validation, increasing firmware size and complexity.

7) Interoperability and Ecosystem Lock-in: Due to proprietary toolchains and vendor-specific SDKs (e.g., STM32CubeMX, Arduino Core, Espressif IDF), developers often get locked into specific ecosystems, limiting portability and reuse. A model optimized for one platform may not translate well to another due to differences in supported operators [200], kernel optimizations, or clock frequency. Moreover, lack of standardized APIs for TinyML deployments hampers community-driven benchmarking and cross-platform evaluation.
8) Energy Profiling and Lifecycle Estimation: Although TinyML is designed for ultra-low-power environments, few tools support holistic energy profiling across sensing, preprocessing, inference, and communication stages. As a result, energy budgeting is often underestimated, leading to battery drain in long-term deployments [1], [198]. Lifecycle estimation is further complicated by temperature variance, sensor degradation, and intermittent power scenarios (e.g., energy-harvested systems). Practical deployments must account for these uncertainties by integrating runtime energy monitors and adaptive sampling techniques.
9) Regulatory and Certification Barriers: In domains like healthcare or industrial safety, TinyML systems must comply with strict regulations (e.g., FDA, CE marking). However, the lack of explainability and formal verification in TinyML models raises concerns about auditability, bias, and failure modes [1], [200]. Most deployment toolchains are not designed to generate artifacts suitable for regulatory review, such as traceable logs, validation certificates, or static model properties. Building certifiable TinyML workflows remains a major research and engineering challenge.

IV. TinyML IN HEALTHCARE

A. APPLICATIONS

TinyML is increasingly transforming healthcare by enabling intelligent sensing, diagnostics, and monitoring directly on low-power wearable or embedded devices. Its ability to perform real-time inference with minimal energy and latency makes it suitable for applications like remote patient monitoring, early disease detection, and rehabilitation support. Figure 6 illustrates the current landscape of state-of-the-art TinyML applications in the healthcare sector.

1) CARDIOLOGY AND CARDIAC MONITORING

a: ECG / ARRHYTHMIAS / ATRIAL FIBRILLATION

In [14], the authors proposed a low-power, real-time ECG classification system based on a CNN deployed on the Arduino Nano 33 BLE Sense. Targeting five heartbeat classes Normal, Supraventricular ectopic, Ventricular ectopic, Fusion, and Unknown the model was trained on the MIT-BIH Arrhythmia Database using 130-sample beat windows. The CNN architecture comprised two 1D convolutional layers and dense layers with ReLU activations and dropout. After quantization via TFLM, the model achieved 96.4% on-device accuracy with low latency and power consumption, validating its feasibility for portable cardiac monitoring. In [15], a TinyML-based, non-invasive cardiac monitoring system was introduced using ECG and PPG signals. Data were collected using AD8232 and MAX30102 sensors and processed via time, frequency, and nonlinear domain features. A 1D CNN was trained and deployed on an Arduino Nano 33 BLE Sense. The ECG model achieved 98.8% accuracy, and the PPG model reached 100%, demonstrating strong performance in resource-constrained, real-time health monitoring environments. In [16], VATML was proposed for life-threatening ventricular arrhythmia detection using intracardiac electrograms (IEGMs) in implantable cardioverter defibrillators (ICDs). The lightweight 1D CNN required only 17.33 KiB memory and 11.51K MACs. Trained on TDC 2022/2023 IEGM data, VATML achieved an F_1-score of 98.33%, generalization of 93.75%, and 97.89% sensitivity on STM32F303K8, outperforming traditional ML and TinyML baselines.

In [17], the authors developed a TinyML system for atrial fibrillation (AFIB) detection using single-lead ECG signals on an ESP32. Data from PTB-XL were preprocessed (filtering, R-peak detection, segmentation), and a 1D CNN was trained and quantized. The model achieved 99.33% inference accuracy on-device, with only 5.3 KB RAM and 69.4 KB Flash usage and <10 ms latency, achieving an F_1-score of 94% for both AFIB and sinus rhythm. In [18], WECG-SW2 was introduced as a TinyML algorithm for AF detection from 30-second ECG signals. The approach extracted QRS-based features and used a 1D CNN with just 3,633 parameters. Trained on 16,790 proprietary ECG strips and validated against clinical and MIT-BIH data, it achieved 99.63% sensitivity and 99.85% specificity, with <20 KB memory, highlighting its suitability for wearable healthcare devices.

FusionGCNN was proposed in [19] as a spatiotemporal GCN for real-time arrhythmia detection on constrained devices. It combines SigNet for local features, DualGCNN for global dependencies, and a fusion gating mechanism. Evaluated on MIT-BIH data, the system achieved 96.41% accuracy and 96.45% F_1-score, while operating efficiently on platforms like Arduino and Raspberry Pi. The work in [20] introduced a 1D-CNN system for cardiovascular disease classification using MIT-BIH ECG signals, segmented into

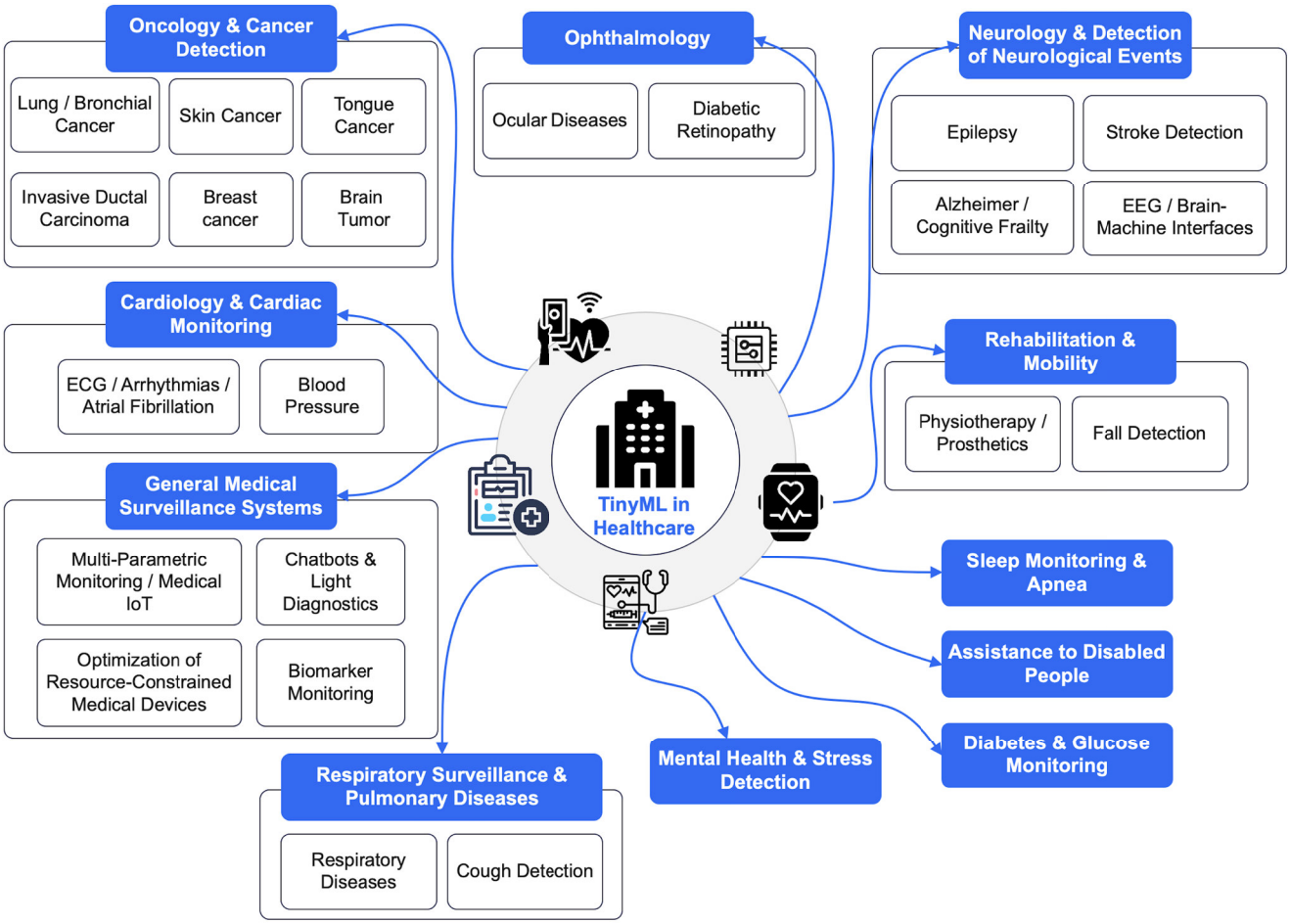

FIGURE 6. State-of-the-art TinyML applications in healthcare.

1024-sample windows. Implemented with an Arduino Uno for acquisition and a Raspberry Pi 3B+ for inference, the quantized TFLite model achieved 93.75% real-time accuracy with 10.8 ms latency. Integration with Node-RED enabled remote monitoring and visualization.

In [21], a TinyML classifier for detecting bundle branch blocks (RBBB and LBBB) was proposed using wavelet-transformed features from MIT-BIH data. A multi-layer perceptron implemented in Edge Impulse achieved an F_1-score of 0.98, with 32.1 KB Flash, 1.5 KB RAM, and 6–8 ms inference time, validating its use in low-resource diagnostic contexts. In [22], 1D and 2D CNNs for ventricular arrhythmia (VA) detection were deployed on STM Nucleo-32 boards. Optimized via hyperparameter tuning and pruning, the 1D CNN achieved an F_β-score of 97.8% (26.2 ms, 5.99 KiB), and the 2D CNN reached 95.15% (11 ms, 3.05 KiB), demonstrating effectiveness for resource-limited ICD applications. In [23], an optimized 1D CNN for VA detection on microcontrollers was proposed, incorporating adaptive pooling, point-wise convolutions, and gradient reversal for subject-level generalization. Trained on IEGM data from ICDs, the model achieved an F_β-score of 0.99265, with 24.3 KiB memory and 2.6 ms latency on STM32F303K8.

A TinyML-based biometric ECG authentication system was presented in [24], using spectrogram features and a DNN deployed on ESP32. Trained on 37,752 heartbeats from 290 PTB subjects, it achieved 97.6% accuracy, outperforming baseline SVM and spectral DNNs, with 271 ms latency and 26.7 KB RAM usage. EQ-NAS was proposed in [25] as a reinforcement learning-based neural architecture search framework for optimizing ECG classification models. Tested on Physionet 2017, it produced sub-256 KB models with accuracy comparable to expert-designed architectures while outperforming NAS baselines like DARTS, proving suitable for embedded deployment. In [26], ECG TinyML introduced a novel model compression method using piecewise linear approximation and knowledge distillation. A ResNet was compressed into a 62.3 KB model with <1% performance loss and 5782× reduced computational cost, enabling efficient deployment on wearable or implantable TinyML hardware. In [27], a lightweight RNN model for fetal heart rate (FHR) classification was developed for edge deployment. Preprocessing included DWT filtering

and BPM-based labeling (L1: >119, L2: ≤119). Trained on PhysioNet data, the quantized model achieved 75% accuracy with low resource use, offering potential for prenatal monitoring in remote settings. In [28], a benchmarking competition was held to develop real-time VA detection models for ultra-low-power ICDs. CNNs and decision trees were optimized for the NUCLEO-L432KC microcontroller. Top models achieved $F_\beta > 0.93$, <1 ms latency, and <20 KB memory, demonstrating the practicality of TinyML for life-critical implantable systems. In [29], a TinyML system for myoelectric prosthetic control was presented, using an MLP trained on sEMG data. Quantized float16 and int8 models were deployed on Arduino Nano 33 BLE, reducing memory 17–24× and latency 3× while maintaining 90% accuracy.

Finally, in [30], an optimized attention-based ResNet was proposed for arrhythmia classification using single-lead ECG. Combining depthwise separable convolutions, pruning, clustering, and quantization, the model achieved a compressed size of 144 KB with 90% real-time clinical accuracy on Arduino Nano 33 BLE, supporting continuous, wearable monitoring under hardware constraints.

b: BLOOD PRESSURE

In [31], the authors presented a continuous blood pressure (BP) monitoring framework leveraging CNNs optimized for resource-constrained edge devices under the TinyML paradigm. Several well-known architectures including AlexNet, LeNet, SqueezeNet, ResNet, and MobileNet were adapted through input dimensionality reduction, pruning, and quantization to fit on microcontrollers such as the ESP32 and Raspberry Pi Pico. The models were trained on photoplethysmography (PPG) signals from the MIMIC-IV dataset, which includes data from over 12,000 ICU patients. PPG signals were segmented into 4-second windows, followed by outlier removal to enhance reliability. The optimized models achieved accuracy levels in line with server-based implementations while satisfying clinical BP estimation standards (AAMI and BHS), demonstrating their suitability for low-power, real-time healthcare applications.

Building on this direction, [32] introduced an edge-computing solution for continuous BP estimation using synchronized electrocardiogram (ECG) and PPG signals. Key cardiovascular features including heart rate and pulse arrival time were extracted from MIMIC-IV data and used to train six machine learning models: Logistic Regression (LR), Support Vector Machine (SVM), Polynomial Regression (PR), Decision Tree (DT), Random Forest (RF), and AdaBoost (AB). These models were optimized through feature reduction and translated into embedded C for real-time inference on microcontrollers such as the Arduino Uno and ESP32. The system achieved a mean absolute error (MAE) of 6.85 mmHg for diastolic pressure, meeting British Hypertension Society (BHS) Grade B criteria, while offering a privacy-preserving, low-latency alternative to cloud-based solutions.

In a related effort, [33] proposed a lightweight convolutional autoencoder (CAE) model for continuous BP monitoring, specifically tailored for deployment on the Arduino Nano 33 BLE Sense. The system extracted four physiological features heart rate, stroke volume, peak width, and diastolic time from PPG signals sourced from the MIMIC-II dataset. The CAE was designed to capture temporal dynamics and suppress noise, with deployment achieved via TensorFlow Lite. Real-time testing on eight volunteers yielded MAEs of 2.25 ± 2.82 mmHg (SBP) and 5.01 ± 2.10 mmHg (DBP), both aligning with AAMI standards and demonstrating superior performance compared to existing edge-based estimators.

Complementing these approaches, [72] introduced a cuffless, non-invasive BP monitoring system using dual accelerometers and TinyML for continuous on-device estimation. The setup involved placing two accelerometers on the common carotid artery to capture pulse transit time (PTT) and ballistocardiography (BCG) signals. A Temporal Convolutional Network (TCN), optimized via platform-aware NAS, was deployed on an STM32 microcontroller, operating within a 100 KB memory footprint and sampling at 3 Hz. A brief 30-second calibration with a cuff-based device enabled personalized transfer learning. Testing on 18 participants (11 pre-clinical, 7 validation) showed post-calibration errors of ±1.7 mmHg (systolic) and ±0.6 mmHg (diastolic), outperforming single-sensor systems by 2.8–4.1×.

2) NEUROLOGY AND DETECTION OF NEUROLOGICAL EVENTS

a: EPILEPSY

In [34], the authors introduced a TinyML-based solution for real-time epileptic seizure prediction using EEG signals, optimized for deployment on embedded platforms such as the Raspberry Pi. Their approach involved preprocessing EEG data from the Bonn dataset and training a Long Short-Term Memory (LSTM) model for binary classification (seizure vs. non-seizure). The model underwent pruning and quantization via TensorFlow Lite, reducing its size tenfold (to 256 KB) while significantly lowering inference latency. Despite these optimizations, performance remained comparable to the original model, demonstrating the viability of TinyML for eHealth applications with benefits such as offline inference, low latency, and enhanced data privacy.

Complementing this, [35] proposed a real-time seizure detection model suitable for implantable neurostimulation devices. Leveraging intracranial EEG (iEEG) signals from a Kaggle dataset containing 1000 labeled training records and 500 test samples, the authors developed a lightweight neural network using the Edge Impulse platform. The model achieved 98% validation accuracy and 99% test accuracy, with an extremely low resource footprint (1.4 KB RAM and 17.3 KB Flash), confirming its suitability for deployment in ultra-constrained implantable systems.

In another direction, [36] presented a wearable seizure detection system integrating multimodal biosignal sensors

including accelerometer, gyroscope, ECG, PPG, and electrodermal activity coupled with a lightweight neural network running on an NVIDIA Jetson Nano. Sensor data were processed in real time for on-device classification, enabling responsive and privacy-preserving operation. While initial validation relied on synthetic seizure events, the study outlined a roadmap for extending the platform to real-world biosignals, supporting proactive and holistic epilepsy management. Meanwhile, [37] focused on a motion-based seizure detection method using accelerometer data collected with an Arduino Nano 33 BLE. The study simulated seizure and normal activity data, applied spectral preprocessing through high-pass, low-pass, and unfiltered filtering schemes, and trained neural networks via Edge Impulse. Among the configurations, the high-pass filter yielded the best performance with 96.3% accuracy, surpassing both the low-pass (84.1%) and unfiltered (93.3%) versions. These results underscored the feasibility of using motion cues for lightweight seizure detection on embedded devices. Finally, [38] proposed LightSeizureNet, a deep learning model designed for efficient real-time seizure detection using raw EEG signals. The architecture featured dilated 1D convolutions, global average pooling, and kernel-wise pruning to minimize computational demands. Evaluated on the CHB-MIT dataset, the patient-independent model achieved 97.09% accuracy with 6.2 million multiply-accumulate operations (MACs), while the patient-specific variant reached 99.77% accuracy with only 3.7 million MACs. Beyond performance, LightSeizureNet offered interpretability by identifying brain regions and frequency bands active during seizures, making it suitable for both clinical and wearable applications under strict power and memory constraints.

b: ALZHEIMER /COGNITIVE FRAILTY

The study in [39] investigated the use of TinyML machine learning tailored for ultra-low-power, edge devices to support research and diagnostics in Alzheimer's disease. Neural network models were developed to process biomedical data aimed at early detection and continuous monitoring of cognitive decline. The dataset combined clinical and neuroimaging inputs, including cognitive assessment scores and biomarker profiles. These models were trained and optimized for deployment on constrained hardware platforms, enabling on-device inference. Experimental results demonstrated that TinyML could deliver real-time, accurate assessments of Alzheimer's progression, providing a scalable and cost-effective alternative to conventional diagnostic systems particularly in resource-limited or remote healthcare environments.

Building on the role of edge intelligence in cognitive health, [73] proposed a point-of-care (POC) system for classifying Cognitive Frailty (CF) in older adults through the integration of TinyML and Lab-on-a-Chip (LOC) technology. The system featured automated blood and urine sample collection via LOC hardware and real-time classification using machine learning models executed on Android smartphones. Feature selection employed statistical techniques including the Kolmogorov–Smirnov test, chi-squared test, and binary logistic regression to isolate relevant clinical indicators such as serum folate, C-reactive protein, and lymphocyte count. The dataset, sourced from the Malaysian Elders Longitudinal Research (MELoR) study, comprised 1,015 samples divided into ''Robust'' and ''Non-Robust'' cognitive states. Among seven evaluated models, Gaussian Naive Bayes (GNB), Linear Discriminant Analysis (LDA), and Support Vector Machine (SVM) achieved classification accuracies exceeding 70%, outperforming baseline methods by approximately 10%.

c: EEG /BRAIN-MACHINE INTERFACES

To enable energy-efficient decoding of EEG signals on resource-constrained microcontrollers, [40] proposed a methodology for designing compact deep CNNs using a neural architecture search algorithm known as μNAS. This approach jointly optimized multiple objectives including classification accuracy, model size, latency, and peak memory usage (PMU) to facilitate real-time inference on low-power MCUs. The models were trained and evaluated using the BCI Competition IV 2a dataset, which contains motor imagery EEG recordings from nine subjects. Experimental results showed that the discovered architectures matched the accuracy of established Shallow ConvNet baselines while reducing PMU by a factor of 20 and improving latency by up to 1.7×, highlighting their suitability for wearable brain-computer interface (BCI) systems.

Complementing EEG-based approaches, [41] presented a real-time, low-power BCI system for robotic control based on electrooculography (EOG) signal classification. The system distinguished between left/right winks and voluntary/involuntary blinks using dry electrodes and an STM32L476RG microcontroller. A lightweight 1D CNN, optimized with TinyML techniques, performed classification after preprocessing steps such as peak detection and min-max normalization. Trained on data derived from the BCI Competition IV 2a dataset, the system achieved an average accuracy of 99.3% and was successfully deployed to control a three-wheeled robotic platform in real time, demonstrating its practical utility for assistive and rehabilitation technologies.

Further advancing on-device processing capabilities, [74] introduced a neural signal compression scheme tailored for brain-machine interfaces, leveraging convolutional autoencoders (CAEs) to reduce wireless transmission bottlenecks such as bandwidth and thermal output. The encoder was implemented on RAMAN, a custom TinyML accelerator optimized for edge computing. The framework incorporated sparsity-aware techniques including zero-skipping, stochastic pruning, and hardware-aware balancing to minimize computational and memory demands. Evaluated on neural recordings from two macaque monkeys performing reach-and-grasp tasks, the system achieved compression ratios of

up to 150×, with strong reconstruction performance: SNDR values of 22.6 dB and 27.4 dB, and R^2 scores of 0.81 and 0.94.

d: STROKE DETECTION

Aiming to provide a rapid and accessible diagnostic tool, [42] proposed a portable, cost-effective system for hemorrhagic brain stroke detection using microwave imaging (MWI) in conjunction with TinyML. The hardware architecture incorporated a Vivaldi antenna array, a 3D-printed human head phantom, and a Raspberry Pi-controlled rotational mechanism to acquire scattering parameters from multiple angles. These signals were then preprocessed and augmented before being fed into a deep learning framework comprising convolutional and dense layers. The trained model achieved a test accuracy of 97.08%, and through pruning and quantization, its size was reduced to 13.12 MB while preserving 93% accuracy enabling deployment on edge devices with constrained resources. In addition to demonstrating real-time diagnostic capabilities (approximately 6 minutes per scan), the study provided an open-source dataset to support future research in MWI-based medical imaging.

While not focused on cerebrovascular stroke detection, [75] presented a related application of TinyML through the development of a wearable body sensor network (BSN) for classifying table tennis strokes. This system utilized IMU sensors specifically accelerometers and gyroscopes mounted on the forearm and shoulder to capture motion data. Preprocessing steps involved FIR and Savitzky-Golay filtering to reduce noise and enhance motion segmentation. A lightweight three-layer neural network was trained to classify three activities: forehand, backhand, and no-stroke. Deployed on an Arduino Nano 33 BLE, the model achieved 90.7% real-time accuracy, with a quantization-aware training (QAT) version reducing memory usage by 75% while maintaining 85.7% accuracy.

3) RESPIRATORY SURVEILLANCE AND PULMONARY DISEASES

a: RESPIRATORY DISEASES

To enable real-time respiratory disease detection at the edge, the authors in [43] developed a TinyML-based system focused on asthma diagnosis using lung sound recordings. Their approach involved bandpass filtering of audio signals, followed by the extraction of Mel-Frequency Cepstral Coefficients (MFCCs) as input features. Three models were evaluated for deployment on low-power platforms such as the Arduino Nano 33 BLE: a custom CNN, a CNN generated via Edge Impulse, and a custom LSTM. Among these, the custom CNN delivered the best performance, achieving 96% accuracy along with 97% precision, recall, and F1-score, while remaining efficient in resource usage (12 KB RAM, 249.6 KB Flash). These results underscore the model's suitability for portable, real-time diagnostics in resource-constrained environments. Building on the concept of edge-based respiratory health monitoring, the authors in [44] introduced a compact diagnostic kit that detects respiratory illness by analyzing exhaled breath using TinyML. Volatile organic compounds (VOCs) were captured by gas sensors, preprocessed, and classified via a lightweight neural network trained using Edge Impulse. The model reached 95.4% accuracy while utilizing only 15.7 KB RAM and 14.5 KB Flash, delivering inference in 6 ms on the Arduino Nano 33 BLE Sense. The system featured LED and LCD outputs for immediate user feedback and operated fully offline, demonstrating the feasibility of privacy-preserving, low-cost diagnostics for underserved regions.

In a related effort, [45] addressed data scarcity in Chronic Obstructive Pulmonary Disease (COPD) detection by proposing a TinyML pipeline powered by synthetic exhaled breath data. Synthetic datasets were generated using Mostly AI to augment limited real-world data, and a neural network model was trained and deployed on STM32-based embedded systems. The synthetic-data model achieved 94.2% accuracy, closely matching the 96.9% accuracy of its real-data counterpart, while maintaining a compact memory footprint (5.4 KB RAM, 92.9 KB Flash) and delivering inference within 12 ms. This work affirmed the practicality of synthetic data augmentation in enabling robust TinyML-based health diagnostics. Similarly, the authors in [46] developed an offline, embedded AI system for detecting respiratory conditions, particularly COPD. Their hardware comprised a custom-designed kit with eight gas sensors, with signal preprocessing and feature extraction conducted prior to neural network training on Edge Impulse. The model was deployed on an STM32 microcontroller and tested via Proteus simulation, achieving 95.3% accuracy while consuming only 1.5 KB RAM and 15.9 KB ROM. With inference latency as low as 1 ms, the system demonstrated a viable solution for non-invasive respiratory screening using embedded AI.

Expanding to infectious disease monitoring, [47] introduced a wearable TinyML-IoT system for COVID-19 detection that leveraged physiological parameters such as heart rate, blood oxygen saturation, and body temperature. The hardware included an ESP32 microcontroller integrated with MAX30102 and NTC thermistor sensors, while an XGBoost model was trained on a dataset from Beijing University's Big Data High-accuracy Center. The model attained high diagnostic performance, with precision between 0.89 and 0.93 and recall between 0.90 and 0.94. On-device inference supported rapid, privacy-preserving detection, complemented by optional cloud monitoring via Thingspeak and adaptive user surveys to refine predictions highlighting the system's relevance for early pandemic intervention and remote healthcare.

b: COUGH DETECTION

A variety of TinyML-based approaches have been proposed for real-time cough detection using resource-constrained hardware. One such system, described in [48], employed a CNN trained on MFCC features extracted from labeled cough

and noise audio data using Edge Impulse Studio. Deployed on an Arduino Nano 33 BLE Sense, it achieved 96.9% accuracy with minimal latency (217 ms) and low memory usage (17 KB RAM), outperforming baseline models such as HMM (82%) and k-NN (93%). To further improve performance in pandemic-specific scenarios, [49] developed a real-time classifier for COVID-19-related cough detection, also leveraging the Nano 33 BLE Sense. Using a similar MFCC-CNN pipeline, this system achieved 99.5% accuracy on a small, manually collected dataset, providing results via BLE or serial output.

Expanding the classification scope, [50] introduced a system capable of differentiating five cough types croup, dry, wet, whooping, and noise. Trained over 1,877 seconds of audio and tested using real-time LED feedback, the system reached 80.83% testing accuracy and 76% success in hardware evaluations, demonstrating feasibility for portable respiratory symptom monitoring. Similarly, [51] proposed an embedded cough classifier using diverse sound categories, yielding a 98% overall accuracy, 94.3% precision, and 98.6% F1-score. Integration with a mobile app enabled validation and real-time usage, positioning it for broader healthcare monitoring applications such as tuberculosis screening. To address privacy concerns inherent to audio-based detection, [52] and [79] explored accelerometer-based solutions. These systems leveraged tri-axial motion data to detect cough events without recording sound. For instance, [52] trained a two-layer NN on time-domain features, achieving 94.38% accuracy on the nRF5340 SoC, with inference times under 1 ms. In parallel, [79] extracted both statistical and spectral features, obtaining 100% cough sensitivity and 0.95 F1-score on a Nordic Thingy:53 MCU using just 13.7 KB Flash.

Another innovative alternative was proposed in [76], where a shirt-collar-mounted accelerometer fed cough classification models with high accuracy achieving 88.8% real-time performance and 100% sensitivity while maintaining privacy and energy efficiency. Additionally, [53] introduced a low-power system embedded in ANC earphones. Their model, TinyCOUNET, achieved 90.0% accuracy and 89.5% F1-score in a clinical field study, while consuming only 5.2 mW additional power, suggesting promise for unobtrusive, wearable cough detection. Complementing sensor-based efforts, [77] and [78] integrated cough detection into broader telemonitoring platforms for COVID-19 patients. In [77], a hybrid wearable system using an ESP32 and MLX90614 sensor detected fever and cough events, achieving robust classification with low resource usage and alerting medical personnel via IoT infrastructure. Meanwhile, [78] combined vital sign monitoring (SpO_2, heart rate, temperature) with cough analysis, deploying two ML models to estimate infection probability, with testing accuracy reaching 99.52%.

Additionally, the authors in [186] proposed a wearable embedded system that simultaneously detects human proximity and coughing events as part of a real-time social distancing and health surveillance solution. The system integrated an ultrasonic sensor, PIR motion detector, and microphone, processed by a lightweight CNN deployed on an nRF52840 microcontroller using Edge Impulse. The model achieved 92.9% accuracy in proximity classification (distinguishing humans, animals, and vegetation) and 68.2% accuracy in cough detection. Despite its multi-modal capabilities, the device maintained inference latency under 250 ms.

4) DIABETES AND GLUCOSE MONITORING

Recent studies have explored TinyML-based methods for non-invasive diabetes detection and continuous glucose monitoring on resource-constrained devices. One such approach [54] utilized an electronic nose (e-nose) comprising MOS sensors to analyze volatile organic compounds (VOCs) in exhaled breath. Data from 44 participants (22 diabetic, 22 healthy) were processed using Discrete Wavelet Transform (DWT) and classified using XGBoost, DNN, and 1D-CNN models. The XGBoost model achieved the highest accuracy (95%), while deep learning models yielded 94.44%. The system was deployed on an Arduino Nano 33 BLE Sense, demonstrating real-time, low-power operation. To enhance model robustness and overcome data limitations, a follow-up study [55] integrated synthetic data generation using Conditional Tabular GANs (CTGAN) with real e-nose data from 58 subjects. After DWT and Z-score normalization, models including LightGBM and Random Forest achieved 86% accuracy for blood glucose level (BGL) prediction and 94.14% for diabetes classification, confirming the value of synthetic augmentation in improving model generalizability for edge deployment.

In another direction, [56] proposed a deep learning framework for non-invasive BGL estimation using photoplethysmography (PPG) signals. By introducing a 1-second segmentation method, the study improved both accuracy and computational efficiency. Among ResNet34, VGG16, and CNN-LSTM models, ResNet34 outperformed with an RMSE of 19.7 mg/dL and 76.6% of predictions within Clarke Error Grid Zone A. The optimized model was successfully deployed on an STM32 microcontroller with a 6.4-second inference latency, enabling real-time, wearable BGL monitoring. Additionally, [57] addressed calibration and drift compensation in CGM systems by generating synthetic sensor data. The study combined simulated glucose profiles from the FDA-approved UVa/Padova simulator with error models reflecting physiological variability and sensor drift. With 500 synthetic traces over 15 days, machine learning models achieved MAEs of 16.13 and 16.22 mg/dL (Random Forest and SVR, respectively), supporting potential deployment on MCUs for real-time correction. Radar-based glucose sensing was explored in [58] through GlucoRadar, a 60 GHz FMCW radar system paired with a compact CNN classifier. In vitro glucose solution signals were processed using FFT-based spectral analysis, achieving

91. 4% precision at 16 glucose levels (50–200 mg/dL). The model's compact architecture (598 kB, 153,074 parameters) supports future on-chip deployment on low-power embedded platforms.

Further contributions include a lightweight convolutional autoencoder (CAE) for glucose monitoring based on PPG signals [59]. Four physiologically interpretable features were extracted and processed to filter noise and model temporal dependencies. Using data from 33 training and 8 test subjects, the CAE achieved a low MAE (5.55 mg/dL cloud, 5.16 mg/dL edge) with 100% of predictions in Clarke Error Grid Zone A. The efficient design (46.32 kB) and successful deployment on an Arduino Nano 33 BLE Sense reinforce the feasibility of low-power, real-time BGL estimation.

5) ONCOLOGY AND CANCER DETECTION

a: LUNG / BRONCHIAL CANCER

The authors in [60] proposed a privacy-preserving federated learning (FL) framework combined with TinyML to enable early detection of bronchus cancer using distributed medical imaging data. Their methodology involved local contrast haze reduction (LCHR) for feature extraction, canonical correlation analysis (CCA) for feature fusion, and neighborhood component analysis (NCA) for dimensionality reduction, followed by classification using a multi-class SVM (M-SVM). The dataset comprised 4,062 CT/X-ray images (including adenocarcinomas, squamous cell carcinomas, and other tumor types) from multiple institutions, processed locally on edge devices without centralized data sharing. The model achieved 95.95% accuracy after NCA-based feature reduction, with sensitivity and specificity exceeding 95%, while minimizing computational costs. The FL-TinyML integration ensured data privacy by training models locally and aggregating updates at a central server.

b: SKIN CANCER

Recent advancements have demonstrated the potential of TinyML for efficient and accurate skin cancer detection and segmentation on resource-constrained devices. In [61], the authors introduced MUCM-Net, a hybrid model combining CNNs, multilayer perceptrons (MLPs), and Mamba state-space models for robust skin lesion segmentation. The architecture incorporated a novel Mamba-UCM block for enhanced feature learning and utilized a composite loss function (BCE, Dice, and squared Dice) to improve segmentation precision. Evaluated on the ISIC2017 and ISIC2018 datasets, MUCM-Net achieved Dice Similarity Coefficients (DSC) of 0.91 and 0.89, respectively, along with high sensitivity (0.93) and specificity (0.95). Despite strong performance, the model maintained low computational requirements (0.055–0.064 GFLOPs) and a compact parameter footprint (71k–139k), making it well-suited for mobile deployment.

Complementing this, [62] proposed resource-efficient CNN architectures (V-CNN and VRES-CNN) for skin cancer classification. Trained on the HAM10000 dataset and enhanced through RandomOverSampler for class balance and input resizing (32×32), these models outperformed larger counterparts like ResNet and EfficientNet in constrained settings. V-CNN achieved over 98.5% accuracy, with significant reductions in model size and inference latency, supporting real-time classification on portable medical devices.

Further innovations include the Double-Condensing Attention Condenser (DC-AC) architecture introduced in [80], which balances deep feature extraction and lightweight computation. Fine-tuned on the ISIC 2020 dataset (33,126 images), the DC-AC model integrated selective self-attention to focus on lesion-relevant regions while limiting computational load. With only 1.6 million parameters and 0.32 GFLOPs, the model achieved AUROC scores of 0.90 (public test) and 0.89 (private test), surpassing larger models like Cancer-Net SCa and MobileViT-S by over 0.13 AUROC, validating its effectiveness for edge deployment. [81] presented a lightweight CNN optimized for edge hardware including the Raspberry Pi and Jetson Nano. Trained on the HAM10000 dataset, the model leveraged separable convolutions and data augmentation, and was converted to TensorFlow Lite for embedded inference. Hardware evaluations demonstrated 98.25% classification accuracy, with the Raspberry Pi 5 achieving the fastest detection time (0.01 seconds) and the Jetson Nano providing energy-efficient operation (3.385 W/h in MAXN mode). These findings reinforce the viability of TinyML for real-time dermatological diagnostics in resource-limited environments.

The authors in [82] proposed two lightweight deep learning models, UCM-NetV2 and BNN-UCM-NetV2, designed for high-precision skin lesion segmentation on resource-constrained devices. The methodology involved enhancing the UCM-Net architecture with a hybrid CNN-MLP module (UCM-NetV2) and further optimizing it using binary neural networks (BNN-UCM-NetV2) to reduce computational costs. The models were trained and evaluated on the ISIC2017 and ISIC2018 datasets, employing a novel dynamic group loss function combining BCE, Dice, and Squared Dice losses. Results demonstrated that UCM-NetV2 achieved a Dice score of 0.9154 with less than 0.04 GFLOPs, while BNN-UCM-NetV2 reduced computations to 0.02 GFLOPs while maintaining competitive accuracy.

Lastly, the authors in [83] proposed a novel hybrid model combining MLP and CNN to address the challenges of skin lesion segmentation, such as computational inefficiency and high parameter counts in existing deep learning methods. The methodology involves the innovative UCM-Net Block, which integrates CNN and MLP for robust feature learning while minimizing parameters and computational overhead, alongside a new group loss function to enhance learning accuracy. The model was evaluated on public datasets PH2, ISIC 2017, and ISIC 2018, demonstrating superior performance with fewer than 50K parameters and less than 0.05 Giga-Operations Per Second (GLOPS). Results showed that UCM-Net outperformed state-of-the-art models like EGE-UNet, achieving higher mean Intersection over Union

(mIoU) and Dice scores (e.g., 89.62% mIoU on PH2) while being significantly faster and more memory-efficient.

c: TONGUE CANCER

The authors in [63] proposed an automated, low-cost prescreening system for detecting benign and premalignant oral tongue lesions using TinyML on resource-constrained edge devices. Their methodology involved retraining a MobileNetV2 model via transfer learning on a clinically annotated dataset of nine types of tongue lesions, followed by quantization to 8-bit integer precision (int8) using TFLM. The dataset comprised 2,232 images, split into training/validation (90%) and testing (10%) sets. The quantized int8 model achieved 98.69% accuracy on the test set while reducing RAM and flash memory usage by over 60% and 80%, respectively, compared to the 32-bit floating-point (float32) model, with minimal latency (1.1 ms). The solution was successfully deployed on an OpenMV Cam H7 Plus edge device.

d: INVASIVE DUCTAL CARCINOMA

In [84], the authors proposed a lightweight TinyML model for the classification of Invasive Ductal Carcinoma (IDC) in breast histopathology images, targeting the limitations of conventional approaches that often require high computational power and memory. Their methodology involved benchmarking multiple deep learning architectures Multilayer Perceptron (MLP), Bidirectional Long Short-Term Memory (Bi-LSTM), and CNNs on the Breast Histopathology Images dataset. After identifying CNNs as the most accurate, the model was further optimized for TinyML deployment through memory and compute footprint reduction. The resulting model achieved 89.3% classification accuracy while consuming only 363.3 KB RAM and 52.9 KB Flash, and was successfully deployed on a Raspberry Pi, demonstrating real-time inference capabilities suitable for resource-constrained clinical settings.

Expanding on the theme of optimized cancer detection, the authors in [85] introduced lightweight versions of MobileNetV1 and MobileNetV2 tailored for real-time histopathological image classification on edge devices. The models were optimized using selective layer unfreezing, structured pruning, and 8-bit post-training quantization, reducing both model size and latency without compromising accuracy. The LC25000 dataset containing 25,000 histopathology images of lung and colon cancers served as the benchmark. Results showed that the optimized MobileNetV1 reduced model size from 13.1 MB to 3.23 MB while achieving an F1 score of 0.99, and MobileNetV2 reached 2.82 MB with a 13 ms inference time. Both models significantly outperformed heavier architectures like VGG16 and ResNet50 in terms of efficiency.

e: BREAST CANCER

The authors in [86] proposed a computer-aided diagnosis (CAD) system leveraging deep learning to classify breast cancer histopathology images on embedded devices, addressing challenges like limited computational resources and the need for real-time analysis in low-resource settings. The methodology employed CNNs, optimized for edge computing, to distinguish between benign and malignant tumors using features extracted from histopathology images. The study utilized datasets like PCam and focused on deploying models on Raspberry Pi devices (3 B+ and 4 B) to evaluate performance under hardware constraints. Results demonstrated the feasibility of achieving high diagnostic accuracy with minimal latency, enabling early cancer detection in clinical environments with limited infrastructure. The system's portability and efficiency highlight its potential to democratize access to advanced diagnostic tools, particularly in underserved regions, while maintaining compliance with privacy and data security requirements through localized processing.

f: BRAIN TUMOR

The authors in [87] proposed an AI-based system to detect brain tumors in cranial MRI images using Edge Impulse®and Tiny ML techniques. The methodology involved training an Object Detection AI (ODAI) with a dataset of 541 superior-view MRI images, sourced from public databases like Kaggle®, which were labeled for tumors and background. The model utilized a Leaky-ReLU activation function, grayscale color depth, and an 80/20 training-testing split, achieving an F1-score of 94.5% in its unoptimized version and 94.4% in the quantized version.

6) MENTAL HEALTH AND STRESS DETECTION

TinyML-based systems are increasingly being explored for real-time mental health monitoring, particularly stress detection, on resource-constrained microcontrollers. In [64], the authors developed a stress detection model based on LSTM networks trained on raw PPG signals from the WESAD dataset. To reflect real-world scenarios, the data was used without preprocessing. The model underwent significant compression using TensorFlow techniques, including 80% pruning, quantization-aware training (QAT), and full integer quantization, reducing the model size from 10.2 MB to 0.86 MB and memory usage to 170 KB. Deployed on STM32H7xx MCUs, the system achieved 87.76% classification accuracy while maintaining cost-efficiency via external flash memory integration.

Expanding on multi-modal integration, [65] introduced a dual-CNN pipeline for context-aware stress detection. A 1D CNN first performed human activity recognition (HAR) using accelerometer data (WISDM dataset), and only during sedentary periods was a second 1D CNN activated for stress classification using heart rate (HR) and electrodermal activity (EDA) data. Both models were optimized through quantization (float16 for HAR and int8 for stress) to fit within the memory constraints of the Arduino Nano 33 BLE Sense. The system achieved 98% accuracy for HAR and 88% for stress detection, with latency of 26 ms and 3.6 s

respectively. A more focused physiological approach was presented in [66], where emotion recognition was achieved using only Galvanic Skin Response (GSR) signals collected from 21 participants exposed to auditory stimuli inducing relaxation, nervousness, and anxiety. Signals were processed via moving averages, logarithmic transforms, and difference metrics. Several models Random Forest, SVM, and LSTM were trained on the extracted features. While classification performance was promising in desktop simulations, the study emphasized the challenge of deploying such models on STM32 microcontrollers under TinyML constraints, particularly regarding model size and computation time.

Additionally, [183] introduced a real-time stress monitoring framework targeting wearable applications for frontline healthcare workers. Using physiological and inertial data collected during the COVID-19 pandemic, the dataset was heavily imbalanced and addressed using Near Miss undersampling. Among six machine learning models evaluated, XGBoost demonstrated the best performance with 86.0% accuracy after hyperparameter tuning. The final model fit within the 2 MB memory limits of the Raspberry Pi RP2040 microcontroller. The authors in [88] proposed a TinyML-based system for real-time stress and sleep monitoring using heart rate variability (HRV) data, deployed on a Raspberry Pi Pico microcontroller. The methodology involved collecting HRV metrics from the SWELL and ISRUC datasets, preprocessing the data, and training a custom feedforward neural network optimized via quantization for edge deployment. The system achieved high accuracy (90% for younger individuals, 82% for older adults) while maintaining low power consumption (99 mW during inference). Results highlighted demographic variations in performance, with males showing higher accuracy (89%) than females (86%).

7) SLEEP MONITORING AND APNEA

Mallick et al. [162] introduced a cost-effective, self-contained system for sleep apnea detection using ECG signals and TinyML, tailored for deployment on resource-constrained edge devices. The system incorporates the AD8232 ECG sensor for signal acquisition, followed by filtering and RR interval-based feature extraction. A neural network classifier, trained on 1.5-minute ECG segments from the MIT-Physionet dataset, was deployed using TensorFlow Lite on devices such as the ESP32, Raspberry Pi Pico, and Arduino Nano 33 BLE Sense. The model achieved an accuracy of 84.21%, with recall and precision exceeding 0.8 for both apneatic and non-apneatic classifications. The final implementation included on-device inference, OLED-based output display, and optional microSD logging, offering a real-time, privacy-preserving solution with a total system cost below $30.

In a more hardware-efficient approach, [163] proposed SABiNN, a binarized neural network architecture tailored for FPGA-based sleep apnea detection. By employing binary weights and shift-accumulate operations, the model drastically reduced computational complexity and memory usage. Trained on RR interval features extracted from 1-minute windows in the MIT-BIH Apnea-ECG database, SABiNN achieved 91.8% accuracy and an F1-score of 0.89. When implemented on a Xilinx Artix-7 FPGA, the model consumed just 22% of logic resources and operated at a clock frequency of 125 MHz, outperforming traditional CNN baselines in both efficiency and performance. Further extending edge deployment options, [164] presented a real-time sleep apnea detection framework using TinyML on microcontrollers including the ESP32 and Raspberry Pi Pico. Two lightweight neural network models were developed: one using raw RR intervals and another based on statistical heart rate features. Both were trained on the MIT-Physionet Apnea-ECG dataset and optimized using quantization via Edge Impulse and TensorFlow Lite. The best-performing model achieved 85.74% accuracy, with F1-scores of 0.85 and 0.86 for apneatic and non-apneatic classes, respectively, while maintaining inference latencies below 135 ms.

8) REHABILITATION AND MOBILITY

a: PHYSIOTHERAPY/PROSTHETICS

In [169], the authors introduced a wearable physiotherapy monitoring system utilizing dual MPU-6050 IMU sensors connected to an ESP32 microcontroller to classify wrist flexion and extension quality in real time. Accelerometer and gyroscope data were collected across six motion categories, ranging from incorrect to optimal form, and a decision tree classifier was trained using the Edge Impulse platform. The final system enabled hybrid on-device inference and mobile app-based cloud feedback, achieving a classification accuracy of 97.43% with low latency and a resource footprint suitable for microcontroller deployment.

Expanding the scope to prosthetic control, [170] proposed a real-time embedded platform using surface electromyography (sEMG) signals processed on an Arduino Nano 33 BLE Sense. A multilayer perceptron (MLP) classifier was trained on five hand gestures using features such as mean absolute value and waveform length extracted through a recursive windowing method to reduce computational load. Confidence-based rejection thresholds were applied to increase classification robustness. TinyML versions of the model quantized to Float-32 and Float-16 achieved accuracy comparable to full TensorFlow models while reducing model size by up to 24× and inference time by two-thirds, supporting low-power, real-time prosthetic control validated in experiments with two participants. In the context of post-stroke rehabilitation, [171] presented a wearable gait monitoring system designed to detect abnormal gait in stroke patients. Motion data from a 9-axis IMU embedded in the Arduino Nano 33 BLE Sense were processed using a 1D CNN trained on time-series features including acceleration, gyroscope, and orientation data. The model was quantized and deployed using TensorFlow Lite, achieving 98.2% accuracy, sub-100 ms inference latency,

and a compact model size under 100 KB. The system was validated in real-time trials with stroke patients.

Further addressing stroke rehabilitation, [89] proposed a wearable sensor system employing TinyML to classify eight upper limb movements in stroke patients. Accelerometer data collected from 10 healthy subjects using an Arduino Nano 33 BLE Sense Rev2 were preprocessed via spectral analysis. A hybrid model combining an MLP and k-means clustering was trained using Edge Impulse, achieving 96.1% training accuracy, 95.09% testing accuracy, and 88.01% deployment accuracy, with an F1-score of 0.961. Quantization reduced the model size by 51.53% (from 52.2 KB to 25.3 KB), while BLE-enabled operation consumed only 54.05 mW, highlighting its efficiency for wearable deployment.

Additionally, [91] introduced a portable, wirelessly controlled TinyML system for assisting dysarthria patients post-stroke through speech command recognition. The approach involved a lightweight deep convolutional neural network (DCNN) paired with Short-Time Fourier Transform (STFT) for feature extraction and real-time inference on an ESP32 microcontroller. The dataset included 106,035 speech recordings comprising dysarthric vowel sounds in Bahasa Indonesia and micro-speech samples, with a focus on ''eee'' and ''iii'' sounds to minimize false positives. The system achieved over 90% accuracy in quiet environments (24 dB), 85% in moderate noise (42 dB), and 50% in high noise (62 dB), demonstrating its utility in speech-based prosthetic control under varying environmental conditions.

b: FALL DETECTION

In [172], the authors developed a wearable fall detection system leveraging TinyML to predict falls and activate a protective airbag belt prior to impact. The system utilized an Arduino Nano 33 BLE Sense to capture inertial data via its onboard IMU, which was analyzed in real time by a lightweight neural network optimized with TFLM. The model was trained on annotated motion sequences representing walking, bending, and various types of falls, focusing on pre-impact patterns. It achieved a classification accuracy of 98.7%, with an inference latency below 90 ms and memory usage under 60 KB. Upon fall detection, the system automatically triggered an airbag to protect the femoral region an innovation targeting a leading cause of morbidity among the elderly.

Building on the objective of senior safety, [92] proposed an advanced TinyML-based fall detection system combining wearable sensors and computer vision for both indoor and outdoor monitoring. Inertial data from accelerometers and gyroscopes were processed locally on low-power ESP32 microcontrollers, while outdoor positioning and physiological signals were tracked using GPS and health sensors. For indoor settings, a vision-based module utilized YOLOv7 and pose estimation to identify fall-related postures. The dataset comprised labeled accelerometry data for activities of daily living (ADL) and fall events, as well as video inputs for pose model training. The integrated system achieved 98.77% classification accuracy, significantly reduced false alarms, and delivered timely alerts to caregivers via SMS and messaging platforms.

9) ASSISTANCE TO DISABLED PEOPLE

In [69], the authors introduced SmartCall, a real-time sign language emergency communicator aimed at supporting speech-impaired individuals in critical healthcare situations. The system utilized an Arduino Nano 33 BLE worn on the wrist to collect inertial data corresponding to five American Sign Language (ASL) medical terms (e.g., ''help,'' ''hospital''). A 1D-CNN was trained offline on gesture data and subsequently optimized for deployment on the embedded device. The model achieved 91.2% offline accuracy and 92% online accuracy post-deployment, with a low-latency response of 2 ms. The system demonstrated robust inter-subject generalizability and offered a privacy-preserving, low-cost solution for emergency communication by individuals with speech disabilities.

Similarly, in [70], the authors proposed a TinyML-based IoT system for translating Indian Sign Language (ISL) into spoken and written formats using a wearable device equipped with accelerometers and gyroscopes. The collected time-series motion data were preprocessed and used to train a Time Series Deep Neural Network (TSDNN). To overcome data limitations, the authors employed deep transfer learning, enabling knowledge transfer from high-end sensor datasets to the target low-cost hardware. The model achieved an average accuracy of 87.18% with as few as four samples per sign and was deployed on an ESP8266 microcontroller. Recognized gestures were transmitted to a cloud platform and vocalized or displayed via the SignTalk mobile application.

10) GENERAL MEDICAL SURVEILLANCE SYSTEMS

a: MULTI-PARAMETRIC MONITORING/MEDICAL IoT

In [173], the authors introduced an integrated TinyML framework for edge-based health monitoring in Medical Internet of Things (MIoT) environments, targeting real-time ECG signal classification. The system utilized an ESP32 microcontroller paired with a low-power ECG sensor, processing signals through a CNN trained on the MIT-BIH Arrhythmia dataset. The model, quantized and optimized with TensorFlow Lite, achieved a classification accuracy of 97.14% while maintaining a compact footprint occupying less than 60 KB of memory and consuming only 87 μJ per inference. The device enabled on-device decision-making and secure MQTT-based cloud synchronization for remote health monitoring. Extending edge intelligence to visual monitoring, [174] proposed a TinyML-powered patient monitoring system using an ESP32-CAM microcontroller. A MobileNetV2 model, trained via Edge Impulse, was deployed to classify patient states such as 'asleep' and 'critical' (e.g., seizure or unconsciousness), based on posture

and motion patterns. The model was quantized and evaluated in two FOMO-based versions, with version 0.35 selected for its optimal trade-off between memory usage and inference performance. Achieving 97.5% accuracy and F1-score with low latency and flash memory consumption, the system transmitted alerts via Twilio SMS, enabling real-time physician intervention in emergency scenarios.

In a broader architecture, [175] introduced a hybrid fog-TinyML framework integrating lightweight ensemble models (Random Forest, LightGBM, and XGBoost) with explainable AI mechanisms for early disease prediction. Trained on UCI datasets for heart disease and diabetes, the models were quantized and compressed for deployment on ESP32, Raspberry Pi 3, and Jetson Nano platforms. The system achieved up to 94.2% and 96.5% accuracy for diabetes and heart disease, respectively, with inference latency as low as 0.018 s on ESP32. To improve transparency, SHAP values and fuzzy rule-based modules were used to provide interpretable insights for medical professionals. In a different application, [176] presented a wearable, skin-conformal pressure sensing system combined with TinyML for real-time classification of physiological signals such as tapping, wrist movements, and swallowing. The device incorporated ultrathin nanocomposite pressure sensors (carbon black/PDMS) into a flexible substrate interfaced with an Arduino Nano 33 BLE Sense. A fully connected neural network, trained on time-series data, was quantized and deployed via TFLM. The system demonstrated high accuracy (98.9%) across multiple users and sensing locations, with memory usage below 80 KB and inference latency under 100 ms.

Lastly, [180] proposed a TinyML-based ECG anomaly detection system for edge deployment in IoMT settings. A 1D-CNN was trained on the MIT-BIH Arrhythmia dataset to classify four heartbeat types, including normal and abnormal events. The quantized model, deployed on Arduino Nano 33 BLE Sense and Raspberry Pi Pico, achieved 98.37% accuracy and an F1-score of 0.983. With a model size under 70 KB and inference latency below 120 ms, the system supported real-time anomaly detection and MQTT-enabled remote diagnostics.

b: CHATBOTS AND LIGHT DIAGNOSTICS

The authors in [181] proposed a resource-efficient AI chatbot system that integrates TinyML for real-time health consultation and symptom assessment on mobile and embedded platforms. The chatbot employs a compressed NLP model trained using BERT embeddings and shallow neural architectures, optimized via quantization and pruning for deployment on devices such as Arduino Nano 33 BLE Sense and ESP32. The model processes text-based symptom inputs and provides medical recommendations while maintaining low latency and memory usage. It achieved an intent classification accuracy of 95.3% with an inference time under 150 ms and a memory footprint below 80 KB. The chatbot was further integrated into a mobile app with secure data transmission for remote healthcare access.

c: OPTIMIZATION OF RESOURCE-CONSTRAINED MEDICAL DEVICES

In [182], the authors proposed a hybrid framework integrating TinyML and federated learning (FL) to enable secure, efficient health data processing on resource-constrained medical devices. The system performs on-device inference for applications such as cough detection, gesture recognition, and elderly monitoring using lightweight CNN and LSTM models deployed on platforms like Xilinx Artix-7 FPGAs and Jetson NX. Federated learning facilitates decentralized training across distributed healthcare nodes without exposing sensitive patient data, employing encryption and secure aggregation protocols to ensure privacy. The framework demonstrated high real-time accuracy with minimal memory and power consumption, validating its suitability for secure, collaborative medical AI at the edge.

In [183], the authors introduced ED-TSC (Energy-efficient Dynamic Time Series Classification), an adaptive TinyML system for classifying multivariate time-series sensor data using early-exit strategies. The framework utilizes a shared encoder and a cascade of classifiers with confidence thresholds, dynamically determining whether to exit early or continue inference to deeper layers. This approach reduces energy consumption while maintaining high accuracy. Evaluated on Epilepsy, PAMAP2, and UCI-HAR datasets, ED-TSC achieved 95.1% accuracy on the Epilepsy dataset with 49.3% energy savings compared to full-depth inference. Deployed on an STM32F746 microcontroller using CMSIS-NN, the system achieved up to 66% energy reduction, demonstrating the effectiveness of early-exit strategies for energy-efficient TinyML deployment. In [184], a hybrid cloud–edge architecture was proposed to support TinyML-based Prognostics and Health Management (PHM) systems. The reference architecture combines local inference on microcontrollers with cloud-assisted training, updates, and feedback. Three architectural patterns were defined: (1) local analysis with cloud fallback, (2) cloud-optimized feedback, and (3) bidirectional model refinement. A prototype system, using Arduino Nano 33 BLE Sense devices and Edge Impulse for training, was evaluated on vibration-based anomaly detection tasks. The deployed models achieved over 95% classification accuracy with inference latency below 150 ms and low energy consumption.

d: BIOMARKER MONITORING

The authors in [185] proposed an AI-augmented electrochemical biosensing system for accurate dopamine detection in the presence of strong interfering agents such as ascorbic acid and uric acid. The system employs a carbon-based screen-printed electrode integrated with an ultra-low-power microcontroller running a lightweight TinyML model trained on voltammetric response data. The authors implemented both a principal component analysis (PCA) and a neural

network classifier to extract and discriminate complex signal patterns. The model achieved over 96.3% classification accuracy across multiple interfering species in synthetic and biological samples, including urine and serum. The optimized TinyML model was deployed on an STM32-based embedded platform with less than 100 KB of flash usage and sub-200 ms inference latency.

11) OPHTHALMOLOGY

a: OCULAR DISEASES

The work in [71] presents a TinyML-based solution for ocular disease recognition using lightweight CNN models optimized for Android-based edge devices. Three models V-CNN, EfficientNet B0, and ResNet20 V2 were trained on the ODIR dataset, which includes fundus images representing eight ocular disease classes. The authors resized and augmented images to 96×96 and 128×128 resolutions to evaluate performance trade-offs. Among the tested models, V-CNN achieved the highest efficiency, reaching 98.83% accuracy on a reduced 6-class dataset, with only 0.35 million parameters and a latency of 6.4 ms. Real-time deployment on a Samsung Galaxy A41 confirmed the model's practicality for low-resource settings. To address real-time diagnosis of ophthalmic diseases on edge platforms, [93] proposed efficient deep learning models for classifying diabetic retinopathy, cataracts, and macular degeneration. A ResNet18 model was compressed using pruning, quantization, and knowledge distillation. Post-training quantization yielded the best trade-off, reducing model size to 11.22 MB while maintaining a validation accuracy of 97.39%. Further improvements were achieved through hybrid techniques. The resulting models were suitable for deployment on devices like the Jetson Nano, supporting low-power, portable diagnostic tools.

ElectraSight, introduced in [94], is a smart glasses system designed for non-invasive, on-device eye tracking. The authors employed a hybrid contact and contactless electrooculography (EOG) setup using QVar sensors to detect corneo-retinal potentials, with a TinyML classifier distinguishing ten eye movement classes. Data were collected from 20 participants, and the system achieved 81% classification accuracy across 10 classes and 92% for a reduced 6-class set. With a response latency under 60 ms and ultra-low energy consumption (7.75 mW sensing, 46 μJ per inference), the system supports continuous usage for over three days on a compact battery. A transfer learning-based approach is detailed in [95] for the detection of retinal diseases such as Choroidal Neovascularization (CNV), Diabetic Macular Edema (DME), and Drusen from Optical Coherence Tomography (OCT) images. The method utilizes a VGG-16 convolutional neural network enhanced with an attention mechanism to boost classification performance. Trained on a public dataset containing 84,484 OCT images, the model achieved 97.79% training accuracy and 95.6% testing accuracy, outperforming MobileNet, InceptionV3, and ResNet. The system demonstrates strong potential for integration into embedded diagnostic tools.

b: DIABETIC RETINOPATHY

The authors in [96] proposed a mobile-based solution for the classification and detection of diabetic retinopathy (DR) using TinyML techniques. The methodology involved evaluating several deep learning models, including ResNet variants (18, 34, 50, 101), RegNet, and WideResNet, implemented in PyTorch and trained on the APTOS 2019 dataset, which contains 3,662 fundus images labeled into five DR severity levels. The models were optimized for mobile deployment using TinyML techniques such as quantization, pruning, and model distillation to reduce computational requirements. The results showed test accuracies of 79.8% for ResNet-18, 81.47% for ResNet-50, and 82.37% for ResNet-101, demonstrating the effectiveness of residual networks in DR classification.

Table 8 presents a comparative summary of recent TinyML applications in the Healthcare sector, highlighting key aspects including implemented algorithms, hardware platforms, model performance metrics, size optimizations, and inference times.

B. OBSERVATIONS AND DISCUSSION

TinyML technologies have demonstrated considerable promise in healthcare, particularly in resource-constrained environments where full-scale cloud or edge solutions may be impractical. However, a critical review of current literature reveals several patterns, limitations, and underexplored areas that must be addressed to move from proof-of-concept to sustainable, deployable healthcare solutions.

1) A majority of studies emphasize technical metrics such as accuracy, latency, and model size. While these are vital for optimization, they often overshadow more critical aspects such as real-world clinical applicability, user acceptance, and integration into existing healthcare workflows. For instance, a model with 95% accuracy on a curated dataset may underperform in noisy, real-life conditions where variability in symptoms, demographics, or sensor conditions could drastically impact outcomes. Moreover, few studies evaluate usability from a clinician or caregiver's standpoint, or consider interoperability with hospital information systems and medical record protocols.
2) Dataset quality and diversity remain significant limitations. Several works utilize public datasets with limited sample sizes or synthetically generated data, which may not represent the target population accurately. This limitation poses risks for generalization, particularly in applications involving critical diagnoses or patient safety. For example, an asthma detection model trained on controlled environments may misclassify background noises in crowded clinics, while ocular disease classifiers developed using homogeneous retinal images may not perform equally across ethnic groups or camera types.
3) Application focus tends to cluster around a few domains such as respiratory diseases, stroke detection, fall

TABLE 8. Summary of research work on healthcare using TinyML.

Task		Year	Ref.	Algorithms	Performance	Model Size	Inference Time	Hardware
Cardiology and Cardiac Monitoring	ECG / Arrhythmias / Atrial Fibrillation	2023	[14]	CNN	Acc: 97%	27 KB	×	Arduino Nano 33 BLE Sense
		2024	[15]		Acc: 98.8% (ECG), 100% (PPG)	19.7 K params	×	Arduino Nano 33 BLE Sense
		2024	[16]		F1-score: 98.33%	17.33 KiB	×	STM32F303K8 (ARM Cortex-M4)
		2022	[17]		Acc: 99.33%	69.4 KB	10 ms	ESP32
		2022	[20]		Acc: 97%, Loss: 0.11%	2.69 MB	11 ms	Raspberry Pi 3B+
		2023	[22]		Fβ-score: 97.8 (1D), 95.15 (2D)	3.05-5.99 KiB	11-26.2 ms	STM32 Nucleo-32
		2024	[23]		Fβ-score: 0.99265, G: 0.9375	24.332 KiB	2.593 ms	NUCLEO-F303K8
		2025	[18]		Sens: 99.63% , Spec: 99.85%	7.2 kB	$\leq$20 ms	Cortex-M4 MCU
		2025	[19]	FusionGCNN (SigNet + DualGCNN)	Acc: 96.41%, F1-score: 96.45%	45-180 KB	105 ms	Raspberry Pi 4, ESP32
		2023	[21]	Multilayer neural network	F1-score: 0.98	32.1 KB	6-8 ms	Arduino Nano 33 BLE Sense
		2024	[24]	DNN (spectrogram features)	Acc: 97.6%	223.6-841.9 KiB	241-377 ms	ESP-EYE (ESP32)
		2023	[25]	NAS-optimized DNN	Acc: 94.2%	< 256 KB	12.8 ms	Arduino Nano 33 BLE
		2021	[26]	Knowledge-distilled DNN	Acc: 93.7%	62.3 KB	×	STM32 MCU
		2022	[27]	RNN	Acc: 75%	×	×	ARM MCU
		2023	[28]	CNN, Decision Tree, HDC	Fβ-score: 0.978	11.18-256 KiB	0.22-200 ms	NUCLEO-L432KC
		2024	[29]	CNN with large kernels	Acc: 97.34%, F1-score: 0.96	143 KB	68 ms	ESP32-S3
		2022	[30]	Light-weight ResNet with attention	F1-score: 0.885 (overall)	144 KB	243 ms	Arduino Nano 33 BLE
	Blood Pressure	2023	[31]	AlexNet, LeNet, SqueezeNet, ResNet, MobileNet	MAE: 5.95 (DBP), 11.27 (SBP)	0.3-1.04 MB	27ms-60.82s	ESP32, RPi Pico, Arduino Nano
		2022	[32]	Linear Regression, SVM, Decision Tree, Random Forest	MAE: 6.85 (DBP), 14.08 (SBP)	2KB-320KB	1.53s-60.82s	Arduino Uno, ESP32, Py-Badge
		2023	[33]	Convolutional Autoencoder	MAE: 2.25 (SBP), 5.01 (DBP)	256KB	64MHz clock	Arduino Nano 33 BLE Sense
		2025	[72]	Temporal Convolutional Network	MAE: 1.7 (SBP), 0.6 (DBP)	< 100 kB	3 readings/sec	STM32WB55RG (Cortex-M4)
Neurology and Detection of Neurological Events	Epilepsy	2023	[34]	LSTM RNN	Acc: 99%	256 KB	0.000015 s	Raspberry Pi 4
		2025	[35]	CNN	Acc: 98-99%	17.3 KB	4 ms	nRF52840 microcontroller
		2025	[36]	Neural Network	High accuracy	×	×	Nvidia Jetson Nano
		2023	[37]	Spectral Analysis + DNN	Acc: 96.3%	×	×	Arduino Nano 33 BLE
		2022	[38]	LightSeizureNet (1D CNN with dilated conv, GAP, pruning)	Acc: 99.77%	113.8 k params	×	Ultra-low power wearable device
	Alzheimer / Cognitive Frailty	2017	[39]	Neural Networks	High accuracy	×	×	Wearables, Microcontrollers
		2024	[73]	LDA, GNB, SVM	Acc: 70.9%	Optimized for Android	×	Smartphones, LOC devices
	EEG / Brain-Machine Interfaces	2023	[40]	CNN, NAS	Similar to SOTA	< 256 KB SRAM	1.7x speedup	Wearables, Microcontrollers
		2023	[41]	CNN	Acc: 99.3%	128 KB SRAM	0.82 s	STM32 MCU, Wearables
		2025	[74]	CAE, DS-CNN	22.6-27.4 dB SNDR	< 256 KB SRAM	45.47 ms	FPGA, MCUs
	Stroke Detection	2023	[42]	CNN	Acc: 93% , F1-score: 0.929	13.12 MB	6 minutes	Raspberry Pi, PocketVNA
		2023	[75]	Neural Network, FIR/SG Filters	Acc: 85.7%	45 KB	×	Arduino Nano 33 BLE, MPU-6050 IMU
Respiratory Surveillance and Pulmonary Diseases	Respiratory Diseases	2024	[43]	CNN, MFCC	Acc: 96%, F1-score: 97%	249.6 KB	127 ms	Arduino Nano 33 BLE
		2023	[44]	Neural Network, VOC Analysis	Acc: 95.4%	14.5 KB	6 ms	Arduino Nano 33 BLE, VOC Sensors
		2021	[45]	Neural Network	Acc: 97.78% (Real Data), 93.33% (Synthetic)	92.9 KB	12 ms	STM32, VOC Sensors
		2021	[46]	Neural Network, K-means Anomaly Detection	Acc: 95.3%	15.9 KB	1 ms	STM32, VOC Sensors
		2024	[47]	XGBoost	Prec: 0.89-0.93, Rec: 0.90-0.94, F1-score: 0.88-0.93, AUC: 0.82-0.87	×	×	ESP32, MAX30102, NTC thermistor
	Cough Detection	2022	[48]	CNN	Acc: 96.9%, F1-score: 0.92	17 KB RAM	217 ms	Arduino Nano 33 BLE Sense
		2022	[49]	CNN	Acc: 99.5%	×	×	Arduino Nano 33 BLE Sense
		2021	[50]	Neural Network	Acc: 80.83%	983 KB	×	Arduino Nano 33 BLE Sense
		2023	[51]	CNN	Acc: 98%, Prec: 94.3%, Rec: 96%, F1-score: 98.6%	×	×	Arduino Nano 33 BLE Sense
		2024	[52]	Neural Network	Acc: 94.38%, Sens: 93.92%, Spec: 94.84%, F1-score: 94.35%	14.8 KB	1 ms	Nordic Thingy:53 (nRF5340 SoC)

TABLE 8. *(Continued.)* **Summary of research work on healthcare using TinyML.**

Task		Year	Ref.	Algorithms	Performance	Model Size	Inference Time	Hardware
Respiratory Surveillance and Pulmonary Diseases	Cough Detection	2022	[53]	Tiny-COUNET (1D CNN)	Acc: 90.0%, F1-score: 89.5%	480 KB	5.2 ms	BES2300YP (ARM Cortex-M4F)
		2023	[186]	CNN	ObjDet: 92%, Cough: 68%	32.2 KB	34 ms (Obj), 1 ms (Cough)	Arduino Nano 33 BLE Sense (nRF52840)
		2023	[76]	Neural Networks	Sens: 100%, F1-score: 0.93	13.7 KB	1 ms	Nordic Thingy:53 (nRF5340)
		2023	[77]	Neural Networks	×	29.6 KB	×	Arduino Nano 33 BLE (nRF52840)
		2024	[78]	DNN	Acc: 99.52% (Vital Signs)	×	×	Custom Wearable (ESP32 + MAX30102)
		2023	[79]	Neural Networks	Sens: 100%, F1-score: 0.95	13.7 KB	1 ms	Nordic Thingy:53
Diabetes and Glucose Monitoring		2024	[54]	XGBoost, DNN, CNN	Acc: 95%,	13.7 KB	0.0049 s	Arduino Nano 33 BLE Sense
		2024	[55]	LightGBM, Random Forest, DNN	Acc: 94.14%	13.7 KB	0.09 s	Arduino Nano 33 BLE Sense
		2025	[56]	ResNet34	RMSE: 19.7 mg/dL, 76.6% in Zone A	7 MB	6.4 sec	STM32H743IIT6
		2024	[57]	Random Forest, SVR	MAE: 16.13 mg/dL	24 KiB	0.192 ms	STM32U5855I
		2024	[58]	CNN	Acc: 91.4% (16 classes)	598 kB	×	60 GHz Radar + MCU
		2024	[59]	Convolutional Autoencoder	MAE: 5.16 mg/dL, 100% in Clarke Zone A	279.85 KB	×	Arduino Nano 33 BLE Sense
Oncology and Cancer Detection	Lung / bronchial cancer	2024	[60]	M-SVM with Feature Fusion (CCA/NCA)	Acc: 95.95%, Sens: 95.99%, Spec: 96.00%	×	×	TinyML Edge Devices (Federated Learning)
	Skin Cancer	2024	[61]	MUCM-Net (Mamba-powered UCM-Net)	DSC: 0.9185, Sens: 0.9014, Spec: 0.9857, ACC: 0.9697	0.071 M, 0.055 GFLOPS	×	Mobile Devices
		2024	[62]	VRES-CNN	Acc: 97.99%	0.12 M , 0.12 GFLOPS	0.60 ms	Android
		2024	[80]	Double-Condensing Attention Condenser	AUROC: 0.90	1.6 M	0.32 GFLOPs	Edge Devices
		2025	[81]	Lightweight CNN	Acc: 98.25%, F1-score: 0.98	976 KB	0.01s (RPi5)	Raspberry Pi 5 / Jetson Nano
		2024	[82]	UCM-NetV2, BNN-UCM-NetV2	DSC: 0.9154, ACC: 0.9682	0.046 M , 0.037 GFLOPs	×	NVIDIA RTX A6000
		2024	[83]	UCM-Net	mIoU: 89.62%, mDice: 94.42%	49.9 K, 0.0465 GFLOPs	×	NVIDIA RTX A6000
	Tongue Cancer	2022	[63]	MobileNetV2	Acc: 98.69%, F1-score: 0.987	1.2 MB	1.1 ms	OpenMV Cam H7 Plus
	Invasive Ductal Carcinoma	2023	[84]	CNN	Acc: 89.3%, F1-score: 0.89	52.9 KB	15 ms	Raspberry Pi 3 B+
		2025	[85]	MobileNetV1/V2	F1-score: 0.98-0.99	2.82-3.23 MB	13-14 ms	Mobile/Edge Devices
	Breast cancer	2024	[86]	CNN	High accuracy	×	×	Raspberry Pi 3B+/4B
	Brain Tumor	2023	[87]	Leaky-ReLU ANN	F1-score: 94.4%	×	×	Edge Devices
Mental Health and Stress Detection		2024	[64]	LSTM	Acc: 87.76%	170 KB RAM	6.23 s	STM32H750VB76 + W25016JV Flash
		2023	[65]	Dual CNN	Acc: 88% (stress), 98% (HAR)	1.1 MB (stress), 152 KB (HAR)	3.642 s (stress), 26 ms (HAR)	Arduino Nano 33 Sense
		2024	[66]	Random Forest, SVM, RNN/LSTM with GSR features	×	×	I×	STM32F401-RE with Grove GSR sensor
		2025	[67]	XGBoost	Acc: 86.0%	1.12 MB	×	Raspberry Pi Pico RP2040
		2024	[88]	Custom FNN	Acc: 90% (young adults)	×	99 mW	Raspberry Pi Pico
Sleep Monitoring and Apnea		2023	[162]	ANN	Acc: 84.21%	124.4 KB	×	Raspberry Pi Pico/ESP32/Arduino Nano 33 BLE
		2023	[163]	SABiNN (Binarized NN)	Acc: 88%	×	10 μW	ASIC (130 nm CMOS)
		2023	[164]	ANN	Acc:82.57% (Model B)	79.9 KB (Model B)	< 3 minutes	ESP32, Raspberry Pi Pico
Rehabilitation and Mobility	Physiotherapy / Prosthetics	2024	[169]	Decision Tree	Acc: 97.43%	< 1 MB	< 5s per movement	ESP32, MPU-6050 IMU
		2023	[170]	MLP	Acc: 92-95%	17-24x smaller than original	×	Arduino Nano 33 BLE, EMG sensors
		2023	[171]	CNN-LSTM	Acc: 88-91%	224.5-1700 KB	< 0.2 s per inference	Raspberry Pi, IMU sensors
		2025	[89]	MLP + K-means	Acc: 95.09%	20.6 KB	15 ms	Arduino Nano 33 BLE Sense Rev2
		2023	[91]	DCNN + STFT	Acc: 93%	20.6 KB	15 ms	ESP32
	Fall Detection	2023	[172]	ConvLSTM	Sens: 99.32%, Spec: 99.01%	×	403 ± 163 ms	Embedded system (belt with IMU)
		2024	[92]	Neural Networks, YOLOv7	Acc: 98.77%, Prec: 0.9875	×	×	ESP32, Raspberry Pi
Assistance to Disabled People		2022	[69]	CNN	Acc: 91.2% (offline), 92% (online)	28.6 KB	2 ms	Arduino Nano 33 BLE
		2024	[70]	TSDNN	Acc: 87.18% (with transfer learning)	28.6 KB	836 ms (end-to-end)	ESP8266, MPU6050

TABLE 8. *(Continued.)* **Summary of research work on healthcare using TinyML.**

Task		Year	Ref.	Algorithms	Performance	Model Size	Inference Time	Hardware
General Medical Surveillance Systems	Multi-Parametric Monitoring / Medical IoT	2023	[173]	Neural Networks	High accuracy	2976 bytes	×	Arduino/Raspberry Pi
		2024	[174]	MobileNetV2 (FOMO v0.35)	Acc: 97.5%	76.5 KB	1354 ms	ESP32 CAM
		2024	[175]	Random Forest	F1-score: 0.93, MCC: 0.84	18.4 KB	27 ms	Raspberry Pi 3 + Arduino Uno
		2025	[176]	Decision Tree, kNN, MLP	Acc: 96.78%-98.43%	< 1 MB	24 ms	Arduino Nano 33 BLE Sense
		2025	[180]	CNN	Acc: 92.89%-99%	193 KB	0.2 ms	Raspberry Pi 4
	Chatbots and Light Diagnostics	2024	[181]	CNN	Acc: 95.3%	50.3 KB	4 ms	Arduino Nano 33 BLE Sense
	Optimization of Resource-Constrained Medical Devices	2025	[182]	CNN, LSTM	Acc: 75-95%	×	×	Wearables, FPGA
		2024	[183]	Temporal CNN (OTCD)	Acc: +2.77-14.74% vs SOTA, F1-score: 0.61	1 KB	44.40 ms	ESP32-based Edge Devices
		2021	[184]	MobileNet (TinyML), U-Net (Cloud)	Acc: 99.5% (MCU), mIOU: 0.5075 (Cloud)	584.9 KB (MCU), 40.8 MB (Cloud)	479 ms (MCU), 7.3 ms (Cloud)	ESP32-Cam (MCU), NVIDIA P5000 (Cloud)
	Biomarker Monitoring	2023	[185]	Dense Neural Network	Acc: 96.01%	93.3 kB	33 ms	ESP32 MCU, Custom potentiostat
Ophthalmology	Ocular Diseases	2023	[71]	V-CNN, EfficientNet B0, ResNet20 V2	Acc: 98.83% (V-CNN)	1.35 MB (V-CNN)	4.58-9.85 ms	Android (Samsung Galaxy A41)
		2025	[93]	ResNet18	Acc: 97.39%	11.22 MB	0.63 s	Jetson Nano
		2024	[94]	Hybrid EOG + 4-bit CNN	Acc: 92% (6 classes)	79 kB	301 μs	Custom Smart Glasses
		2022	[95]	VGG16 + Attention Mechanism	Acc: 97.79%	592 kB	×	×
	Diabetic Retinopathy	2023	[96]	ResNet-18, ResNet-34, ResNet-50, ResNet-101, RegNet, WideResNet	Acc: 82.37% (ResNet-101)	×	×	Smartphone

detection, and prosthetic control. These are undoubtedly important, yet the potential of TinyML is far broader. Areas like chronic disease self-management, pediatric monitoring, maternal health, and real-time mental health support remain relatively unexplored. The lack of diversity in application areas could reflect publication bias or limitations in available datasets, but it also suggests opportunities for more inclusive and wide-reaching innovations.

4) Energy efficiency and memory usage are frequently reported, yet often without sufficient context. While it is encouraging to see model sizes under 100KB and inference latencies below 100ms, these figures lack deployment realism without discussions on battery life under real use, firmware update protocols, or sensor-node integration. For example, an ultra-efficient fall detector must also manage continuous data logging, alert signaling, and potential cloud syncing, which together influence system lifespan and user adoption.
5) Ethical considerations are inconsistently addressed. With increasing use of biometric data, emotion recognition, and real-time monitoring in TinyML systems, there is a pressing need to discuss privacy, data security, algorithmic fairness, and consent. While a few studies make passing references to on-device processing as privacy-preserving, they often lack formal ethical reviews or user feedback. This is especially concerning for systems targeting vulnerable populations such as the elderly, children, or patients with cognitive impairments.
6) Long-term deployment considerations are largely absent. Most systems are validated over short trials or synthetic datasets, without discussion of how these models handle device drift, changing patient behavior, sensor calibration issues, or clinical handoff scenarios. Additionally, no standardized framework exists for validating TinyML healthcare devices across settings, raising questions about reproducibility and longitudinal robustness.

C. LESSONS LEARNED AND RECOMMENDATIONS

Based on the collective insights from the reviewed studies and trends in TinyML applications within healthcare, several important lessons have emerged that can guide future research, development, and deployment efforts. These are presented as detailed recommendations:

1) Prioritize clinically relevant validation and real-world integration. While high accuracy, low latency, and compact model size are essential performance indicators, they do not alone guarantee clinical success. Future work must go beyond technical benchmarks and rigorously validate models in real-world clinical environments, ideally through collaboration with healthcare institutions. This includes assessing patient safety, caregiver usability, environmental variability (e.g., noise, lighting), and workflow integration. Clinical trials, user-centered design approaches, and Health Technology Assessments (HTAs) should become standard components of TinyML healthcare innovation pipelines. Without such validation, even highly optimized models risk remaining academic exercises.
2) Expand dataset diversity and promote inclusive benchmarking. A recurring limitation is the dependence on small, homogeneous, or synthetic datasets, which undermines the generalizability and ethical robustness of many TinyML solutions. It is crucial to prioritize diverse, representative datasets that include various ages, ethnicities,

comorbidities, and device types. Future research should also include cross-device and cross-region validation to assess robustness under different usage conditions. Initiatives to share anonymized, labeled, and ethically curated datasets especially from underrepresented settings would dramatically improve reproducibility and fairness in healthcare TinyML models.

3) Broaden the scope of TinyML applications in healthcare. The current landscape is skewed toward a few high-frequency use cases (e.g., fall detection, stroke, respiratory illness). Researchers and developers should consider expanding into overlooked areas such as maternal and neonatal care, chronic disease adherence monitoring, pediatric growth tracking, pain assessment, and digital therapeutics for mental health. TinyML's low-cost, offline capabilities make it particularly suitable for low-resource and rural contexts, as well as for long-term outpatient care. Diversifying application domains will not only address unmet needs but also reveal the full potential of embedded intelligence in public health.
4) Adopt hybrid evaluation frameworks that incorporate both technical and healthcare-centric indicators. While TinyML researchers often report metrics like model size and inference time, these should be complemented by health-related indicators such as diagnostic value, time-to-intervention, false alarm rate, and end-user satisfaction. Frameworks that fuse clinical relevance with technical efficiency are needed to guide the development of practical, trusted solutions. For example, a 98
5) Address privacy, ethics, and regulatory readiness early in the design cycle. As TinyML systems increasingly handle sensitive biosignals and behavioral data, ethical issues must not be an afterthought. Developers should implement privacy-by-design principles, on-device anonymization, and mechanisms for user consent and opt-out. Bias mitigation strategies should be included from dataset selection through model deployment. Regulatory bodies are beginning to formalize guidelines for AI in healthcare, and compliance with standards like GDPR, HIPAA, or MDR (EU) should be treated as integral to development not as post-hoc add-ons.
6) Plan for long-term usability, maintenance, and support. TinyML healthcare devices are often validated in short trials but lack provisions for long-term use, such as firmware updates, hardware degradation, sensor recalibration, and battery lifespan. Solutions should account for edge cases such as loss of internet access, caregiver training, multilingual support, and sustainability of usage over months or years. Community feedback mechanisms and version tracking can improve model robustness over time. Moreover, deployment pipelines should support continuous learning or personalization where clinically safe.
7) Foster interdisciplinary collaboration and user co-design. Successful healthcare innovation requires integration across AI, embedded systems, medicine, and human factors. Teams should include clinicians, biomedical engineers, human-computer interaction specialists, and public health experts from the outset. Co-designing with end-users patients, nurses, caregivers ensures the system fits real-world workflows, user needs, and cultural contexts. Workshops, pilot studies, and participatory prototyping can uncover hidden barriers and improve adoption rates.
8) Encourage open, low-cost toolkits and reproducibility in research. To democratize TinyML healthcare innovation, researchers should publish reproducible code, model architectures, datasets, and deployment scripts. Platforms like Edge Impulse, TFLM, and Arduino can be paired with clear documentation to support community engagement. Creating open-source reference designs for common use cases (e.g., pulse oximetry, motion detection, anomaly alerting) can accelerate innovation, especially in low-income regions or educational contexts.

V. TinyML IN EDUCATION

A. APPLICATIONS

Recent studies have demonstrated the growing role of TinyML in enhancing educational outcomes through hands-on, low-power embedded AI systems (see Figure 7).

One such initiative introduced a TinyML-based workshop targeting middle school students, utilizing the Arduino Nano 33 BLE Sense to develop a proximity sensor classification project [2]. Spanning five sessions totaling 20 hours, the curriculum integrated Python programming, TensorFlow Lite, and core AI principles. To assess the workshop's educational impact, the bASES21 questionnaire was administered before and after the intervention, measuring four skill areas: Learning and Teamwork (LTE), Citizenship and Social Responsibility (CSR), ICT Proficiency, and Communication. Although statistical analysis revealed no significant differences across these domains e.g., LTE (pre: 3.18, post: 3.26, $p = 0.946$) certain individual items like ''I can explain my opinions and decisions'' exhibited a 27.08% improvement. The authors concluded that while TinyML holds strong pedagogical potential, extended duration and improved hardware access may be necessary to achieve quantifiable gains.

In another study, a cost-effective and real-time facial recognition-based attendance system was developed using the ESP32-CAM microcontroller and TinyML [3]. The proposed solution integrated a Flutter-based Android app to manage facial image acquisition, real-time attendance tracking, and user interaction. The core recognition model, based on MobileNetV2 with an input resolution of 96×96 and a width multiplier of 0.35, was trained on a custom dataset captured through the app. Achieving a validation accuracy of 87.12%, the model demonstrated individual recognition accuracies ranging from 91.34% to 97.78%. Following training and quantization via TensorFlow Lite, the model was deployed directly on the ESP32-CAM, enabling

FIGURE 7. State-of-the-art TinyML applications in education.

offline inference and automated data logging to Google Sheets. The app additionally featured real-time messaging, study materials, and notifications providing a comprehensive educational tool.

Expanding the scope of emotion-driven learning analytics, the authors in [4] proposed a lightweight emotion recognition system using a spatiotemporal graph convolutional network (ST-GCN) augmented with a multi-scale semantic fusion mechanism. Designed for deployment on edge devices, the system leverages 2D facial landmark sequences (68 keypoints) to form region-connected graph data, enabling the ST-GCN to extract spatial and temporal emotion cues. The model was evaluated on the publicly available CK+ dataset and a custom OSCASA (Online Student Community Assessment Sentiment Analysis) dataset, achieving accuracy scores of 98.2% and 98.8%, respectively. This performance significantly outperformed classical methods like Optical Flow+SVM and LBP+CNN, while maintaining low computational demands suitable for TinyML integration in emotion-aware educational platforms.

Subsequently, an inquiry-driven educational framework was introduced that combines the Raspberry Pi 3 with a compressed YOLOv3 model to perform real-time object detection and contextual learning [5]. Trained on the COCO dataset encompassing 80 object categories, the model supports educational engagement by enabling students to capture physical object images and receive corresponding textual and visual content via automated Wikipedia and Google image scraping modules. Despite accuracy trade-offs due to model compression e.g., person (96%), car (70%), bicycle (57%) the system delivers a complete image-to-knowledge workflow within 5–7 minutes per sample. This setup showcases how TinyML-powered computer vision can facilitate active learning through embedded inquiry-based exploration.

In parallel, a context-aware TinyML model for Korean handwriting recognition designed for online assisted learning [6]. The methodology involved integrating a Mamba layer into a CRNN to improve global context modeling with linear complexity and employing a multi-layer joint distillation mechanism from a teacher network to compress the model. The model was trained and evaluated on two publicly available Korean character datasets, AIHub and MLT-h. The results demonstrated that their distilled student model achieved competitive word-level accuracies of 94.2% on AIHub and 86.8% on MLT-h, making it suitable for deployment on resource-constrained devices while maintaining high performance.

Concurrently, a study introduced a low-cost, open-source hardware prototype for bearing failure detection using TinyML and vibration analysis [7]. The methodology involved constructing a physical test bench with a DC motor, bearings, and inertial sensors (such as MPU6050 or Arduino BLE Sense boards) to collect vibration data, which was processed using the Edge Impulse platform to train a MLP neural network with spectral features extracted via Fast Fourier Transform (FFT). The prototype was validated using a custom-built dataset collected from both healthy and defective bearings. The results demonstrated that the system could successfully classify bearing conditions in real-time on embedded devices, providing an accessible educational tool for teaching predictive maintenance, AI, and TinyML concepts in academic settings.

Finally, a study designed a TinyML-based edge intelligent system for assessing English classroom quality, addressing the limitations of traditional subjective and inefficient evaluation methods [8]. The methodology integrated an ''end-edge-cloud'' architecture using multimodal sensors (microphones and cameras) to collect real-time speech and visual data, which were processed on edge devices (e.g., Rockchip RK3588, ESP32-S3) via lightweight deep learning models optimized with quantization and pruning. The system employed advanced feature extraction techniques, including enhanced MFCC and Gammatone filters for speech emotion recognition, and ellipse-fitting algorithms for visual attention analysis. Experiments involved 15 classes and over 600 students, showing that the system achieved 91.3% emotion recognition accuracy, reduced power consumption to 3.2W, and decreased end-to-end delay to 120ms. Most significantly, students in the experimental group saw an average score increase of 12.3 points, demonstrating the system's effectiveness in enhancing teaching quality and student engagement.

Table 9 presents a comparative summary of recent TinyML applications in the education sector, highlighting key aspects including implemented algorithms, hardware platforms, model performance metrics, size optimizations, and inference times.

TABLE 9. Summary of research work on education using TinyML.

Task	Year	Ref.	Algorithms	Performance	Model Size	Inference Time	Hardware
Integrating AI and Machine Learning in Education (K-12)	2023	[2]	CNN	Improved student engagement (qualitative)	×	×	Arduino Nano 33 BLE Sense
Automated Attendance and Management	2023	[3]	MobileNetV2, Transfer Learning	Acc: 87.5% , Loss: 0.50	×	×	ESP32-CAM, I2C OLED
Analysis of Student Emotions and Engagement	2025	[4]	ST-GCN, Multi-scale Semantic Fusion	Acc: 98.2% (CK+), Acc: 98.8% (OSCASA)	×	×	Edge devices (cameras)
Practical Learning and Physical Computing	2022	[5]	YOLOv3, Darknet-53	Acc: 66% (bus class)	×	5–7 min	Raspberry Pi 3 + Pi Camera
Korean Handwriting Recognition Under Online Assisted Learning Scenes	2025	[6]	CRNN + Mamba, Knowledge Distillation	Acc: 94.2% (AIHub), 86.8% (MLT-h)	×	×	Wearable devices
Bearing Failure Detection	2025	[7]	MLP (FFT Spectral Features)	Acc: 100.0% (Edge Impulse)	×	×	ESP32, Arduino BLE Sense, MPU6050
English Classroom Quality Assessment	2025	[8]	MobileNetV3 (GhostModule), Causal Conv1D, Gaze/Head Pose Estimation	Acc: 91.3% (Speech Emotion), 3.2° (Gaze Error)	×	120 ms	Rockchip RK3588, ESP32-S3, NVIDIA Jetson Orin Nano

B. OBSERVATIONS AND DISCUSSION

Despite encouraging early results, the application of TinyML in education remains a developing area with several gaps and limitations. A critical review of the selected studies reveals recurring challenges that constrain the broader adoption and pedagogical effectiveness of TinyML solutions in academic settings.

1) Current research on TinyML in education is limited in scope and often restricted to short-term, small-scale pilot studies, with little longitudinal analysis to evaluate sustained educational impact.
2) Most studies focus on technical feasibility such as model accuracy, memory footprint, and latency while overlooking pedagogical metrics like knowledge retention, conceptual understanding, or cognitive skill development.
3) There is a lack of demographic and contextual diversity; experiments are frequently conducted in controlled environments with homogeneous student groups, limiting their applicability to real-world classroom settings, particularly in low-resource regions.
4) Emotion recognition and other affective computing applications, though technically advanced, introduce significant ethical concerns. These include student privacy, data security, informed consent, and potential algorithmic bias issues largely unaddressed in the literature.
5) Comparative analysis across studies is hindered by the absence of standardized benchmarks for educational TinyML applications, making it difficult to assess reproducibility or cross-context generalization.

C. LESSONS LEARNED AND RECOMMENDATIONS

Building on the preceding observations, several key lessons and actionable recommendations can help guide future development and deployment of TinyML in education. These points aim not only to address current limitations but also to propose concrete directions for scalable, inclusive, and pedagogically meaningful applications.

1) Expand future research to include diverse learning environments particularly rural, low-income, and underserved schools by piloting TinyML projects in these contexts. This would help assess the scalability, cost-effectiveness, infrastructure needs (e.g., power, internet), and overall real-world viability of TinyML-enabled educational interventions in non-ideal conditions.
2) Shift the focus from system-centric metrics (e.g., deployment success, technical feasibility) to learner-centric perspectives by co-designing TinyML tools with teachers and students. Emphasize alignment with educational goals such as conceptual understanding, collaborative problem solving, inquiry-based exploration, and differentiated instruction tailored to diverse learning styles.
3) Conduct longitudinal and mixed-method studies that go beyond immediate engagement or satisfaction. These should track long-term impacts on cognitive development (e.g., knowledge retention, critical thinking), behavioral changes (e.g., increased STEM interest), and skill acquisition (e.g., computational thinking, hands-on prototyping), using pre-post assessments, interviews, and classroom observation over extended periods.
4) Ensure ethical and transparent deployment of TinyML systems in educational settings, especially when involving sensitive modalities such as facial recognition, emotion tracking, or behavioral monitoring. Address issues of data privacy, informed consent (from both students and guardians), algorithmic bias, and inclusivity, by adhering to established educational technology ethics guidelines.
5) Develop robust and multidimensional evaluation frameworks that integrate both technical performance metrics (e.g., model accuracy, inference speed, energy efficiency) and pedagogical effectiveness indicators (e.g., learner engagement, motivation, time-on-task, and assessment

performance). Such hybrid frameworks would support more holistic assessments of TinyML tools in real classrooms.

6) Encourage interdisciplinary collaboration by forming teams that include AI engineers, teachers, learning scientists, curriculum developers, and educational psychologists. This collaboration ensures that TinyML solutions are not only technically sound but also pedagogically meaningful and developmentally appropriate for targeted age groups.
7) Promote the use of open-source TinyML platforms (e.g., Edge Impulse, TFLMs, Arduino IDE) and provide accessible learning resources. Supporting DIY and maker-based learning environments will empower both teachers and students to build, modify, and understand TinyML systems, fostering a culture of creativity, agency, and technological fluency.
8) Broaden the scope of TinyML applications in education by exploring diverse and practical use cases, such as real-time feedback systems for language learning (e.g., pronunciation or public speaking analysis), portable sensor-based lab kits for hands-on science experiments, emotion and attention monitoring tools to support adaptive tutoring, handwriting and sketch recognition for literacy and arts education, TinyML-driven learning games that dynamically adjust difficulty, low-cost voice assistants for inclusive learning in multilingual or special education contexts, embedded activity monitors for physical education and biomechanics instruction, interactive storytelling devices responsive to gestures or expressions, DIY kits that empower students to build and program their own TinyML projects, and environmental monitoring tools that teach air quality, noise, or climate literacy through real-time data.

VI. TinyML IN TRANSPORTATION

A. APPLICATIONS

TinyML has emerged as a key enabler of intelligent transportation solutions by allowing real-time processing on low-power edge devices. It supports a wide range of applications such as autonomous navigation, traffic monitoring, and in-vehicle diagnostics. Figure 8 presents an overview of current state-of-the-art applications in this domain.

1) INTELLIGENT VEHICLES AND DRIVER MONITORING

a: DRIVER BEHAVIOR ANALYSIS

In [139], the authors introduced a hierarchical architecture for real-time driver behavior classification that integrates IMU data processing, edge intelligence, and embedded deep learning. The system comprises two key layers: the first layer extracts time-domain statistical features from accelerometer and gyroscope signals to identify road types (e.g., straight, curved, hilly), while the second layer uses a lightweight 1D-CNN model to classify driver behavior into categories such as aggressive, normal, and drowsy. Data were collected from real-world driving sessions using a smartphone mounted on a vehicle's dashboard. After segmentation and feature extraction, the model was trained on 70% of the dataset and tested on the remaining 30%, achieving an overall classification accuracy of 96.75%. The final model was quantized and deployed on an Arduino Nano 33 BLE Sense, achieving an average inference time of 97 ms and consuming less than 50 KB of flash memory.

The study in [140] proposed a novel k-fix AutoCloud algorithm designed for TinyML-based, real-time driver behavior analysis. This unsupervised online algorithm leverages the principles of typicity and eccentricity to identify behavioral patterns from streaming vehicular data without the need for prior training, making it ideal for low-power microcontrollers. A four-day real-world case study was conducted using a Nissan Kicks vehicle equipped with an OBD-II sensor and the Torque Pro application to collect speed and throttle position data. The proposed k-fix AutoCloud algorithm formed 4–5 data clusters per day and demonstrated performance comparable to the standard AutoCloud model. Clustering effectiveness was validated using Dunn and Davies-Bouldin indices (e.g., Dunn index: 0.028–0.029; DB index: 0.521–0.688), showing that the new approach simplifies cluster size estimation and reduces computational overhead without compromising analytical precision—highlighting its suitability for TinyML deployment in intelligent transportation systems.

In [141], the authors developed a sparse representation-based learning model to classify driver behavior using in-vehicle Controller Area Network (CAN) bus data, optimized for TinyML environments. The dataset was collected from real-world driving sessions involving 16 drivers, capturing 15 sensor signals—including brake pressure, throttle position, and engine RPM—across aggressive, moderate, and mild driving styles. A two-layer sparse encoder coupled with K-means clustering was trained on 80% of the data and tested on the remaining 20%, yielding a classification accuracy of 95.3%. The model maintained strong performance even when reduced to only 5 input signals, still achieving an 88.7% accuracy, thereby demonstrating its efficiency in signal compression. Designed for deployment on low-power platforms such as the Raspberry Pi Zero or similar microcontrollers, the model had a memory footprint of less than 100 KB and supported real-time inference capabilities.

Finally, in [142], the authors proposed an embedded driver monitoring system to detect eye closure and head position using deep learning models executed on a Raspberry Pi 4. The goal was to identify early indicators of fatigue and distraction. A Pi camera was used to capture real-time video frames, which were processed by two CNN-based models: one for eye state classification (open/closed) and another for head pose estimation. Training was conducted using publicly available datasets—CEW (Closed Eyes in the Wild) for eye detection and BIWI Head Pose for head orientation. The eye detection model achieved an accuracy of 98.3%, while the head pose estimator reported a mean absolute error of 3.4°. Real-time

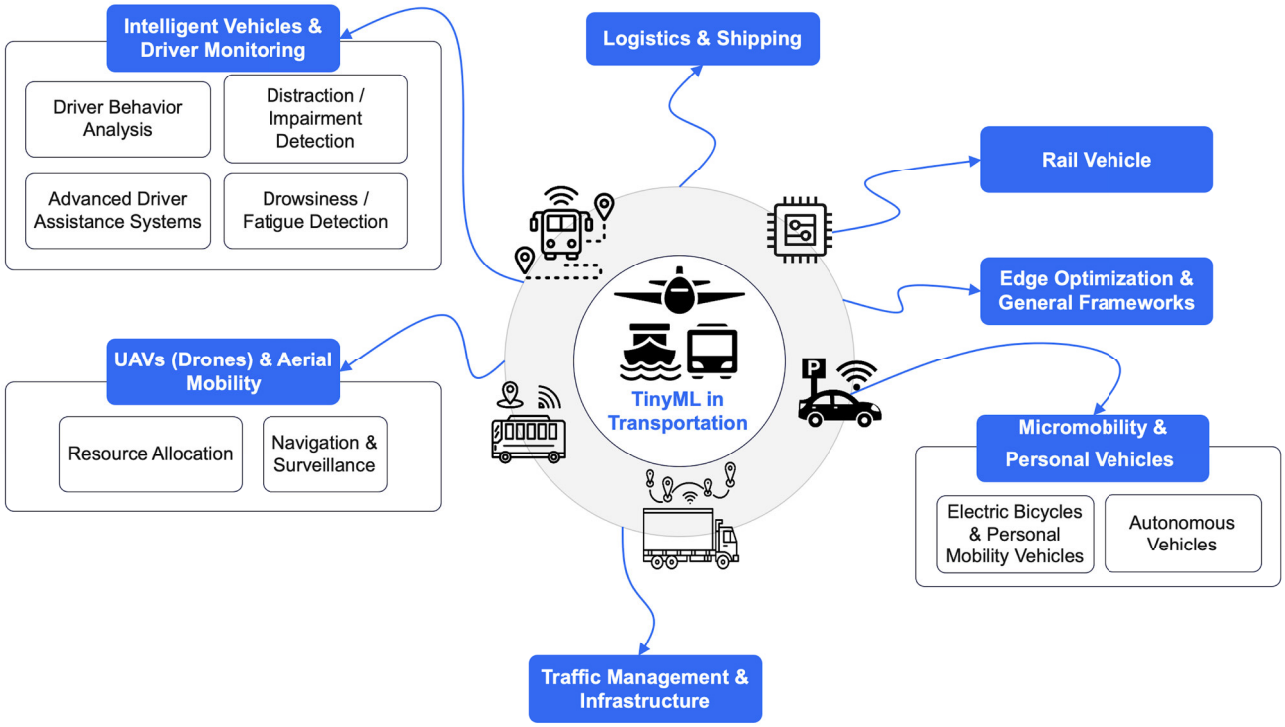

FIGURE 8. State-of-the-art TinyML applications in transportation.

performance evaluation revealed an average inference time of 75 ms per frame, confirming the system's feasibility for in-vehicle applications. per frame, suitable for in-vehicle deployment.

b: DISTRACTION / IMPAIRMENT DETECTION

To address the problem of distracted driving, [130] proposed an optimized convolutional neural network deployed on the NVIDIA Jetson Nano platform. Their solution focused on recognizing ten distinct driver postures using the State Farm Distracted Driver Detection dataset, which includes behaviors like texting, drinking, and talking. A modified MobileNetV2 architecture was trained using transfer learning and enhanced through data augmentation to improve generalization. The final model, quantized and optimized with TensorRT, achieved 92.5% accuracy and delivered real-time inference at 15.38 ms per frame. The complete system integrated a camera module and live dashboard for continuous monitoring, demonstrating its suitability for in-vehicle deployment.

Focusing on alcohol impairment detection, [131] developed a fully offline, embedded system using a Wio Terminal microcontroller and a Grove Multichannel Gas Sensor. Data were collected over a month, covering four classification categories: *Alcohol 1*, *Alcohol 2*, *Clean*, and *Coffee*. A Keras-based model was trained and quantized via Edge Impulse, achieving classification accuracies between 96% and 99%, as validated through confusion matrix analysis. The device featured an integrated LCD and buzzer for immediate feedback, eliminating reliance on cloud infrastructure and emphasizing privacy and low-latency processing key advantages of TinyML in vehicular contexts.

Expanding on real-time driver monitoring, [132] introduced a distraction detection system built around the Arduino Portenta H7 microcontroller. Using Edge Impulse Studio, a CNN model was trained on a subset of the State Farm dataset, encompassing 22,424 images across five driver activity classes. The quantized int8 model achieved 99.3% accuracy with a 72 ms inference latency, while drastically reducing memory usage requiring only 164 KB RAM and 52.7 KB Flash. Compared to its float32 counterpart, this represented a 74% reduction in memory and a 45% improvement in execution speed. Integration with a Vision Shield enabled real-time visual input, and evaluation showed F1-scores ranging from 0.99 to 1.00.

c: ADVANCED DRIVER ASSISTANCE SYSTEMS (ADAS)

To deliver real-time, modular driving assistance on edge hardware, [133] introduced a comprehensive framework that combines lightweight deep learning models for lane detection, traffic sign recognition, and object detection. Deployed on a Raspberry Pi 4B, the system utilized a pruned version of LaneNet trained on the TuSimple dataset (95.45% accuracy), a MobileNetV2-based classifier trained

on GTSRB (98.4% accuracy), and YOLOv3-Tiny trained on Pascal VOC (89.5% mAP). All models were quantized and optimized to meet the performance constraints of embedded platforms. The framework demonstrated real-time inference capabilities with latencies of 19 ms (traffic signs), 31 ms (lane detection), and 45 ms (object detection), validating its suitability for low-cost, energy-efficient ADAS deployments.

Expanding on vehicle event recognition, [134] presented a sensor-driven TinyML methodology leveraging MEMS and IMU sensors to classify vehicular handling and misuse behaviors. Data were collected from multiple car locations during scenarios such as door knocking, pulling, or closing with varied intensity. Preprocessing included mean-centering, auto-scaling, PCA-based dimensionality reduction, and SMOTE resampling to handle class imbalance. A Random Forest model trained on this augmented dataset achieved a significant accuracy boost from 58.4% to 93.8% with F1-scores between 0.92 and 0.99. The framework showed strong performance in distinguishing event types, though detection of subtle actions remained a challenge.

As a complementary application within ADAS, [135] introduced a compact CO_2 emission estimator using a Multilayer Perceptron (MLP) model embedded in a Macchina A0 OBD-II scanner. Trained on real-world data from four vehicles, the system used features like intake manifold pressure, air temperature, and vehicle speed to infer emissions based on physical ground-truth calculations. Various MLP architectures were evaluated using different activation functions (ReLU, Sigmoid), with TensorFlow Lite quantization techniques applied for optimization. The best-performing model reduced mean absolute percentage error (MAPE) by 50% and improved inference efficiency by 78.3%.

d: DROWSINESS / FATIGUE DETECTION

To support real-time detection of driver drowsiness on embedded platforms, [136] developed a lightweight CNN-based system deployed on the Arduino Nano 33 BLE Sense. Utilizing grayscale images from a camera module, the system classifies eye states (open vs. closed) to infer alertness levels. The model was trained on a combination of the public YawDD dataset and custom images collected under varying lighting and orientation conditions. Quantized with TFLM, the final model achieved 95% accuracy and sub-100 ms inference latency. Alerts were delivered via LED or buzzer upon prolonged eye closure, offering a compact and energy-efficient ADAS solution. A complementary approach by [137] employed behind-the-ear EEG signals for drowsiness detection, using flexible printed electrodes and a fully connected neural network (FCNN) deployed on the same microcontroller. EEG signals were filtered, segmented, and fed into a 1D neural network trained via Edge Impulse. The system was evaluated using fatigue-inducing driving simulations with 10 participants, achieving 91.4% classification accuracy while consuming only 17 KB of RAM and 48 KB of Flash. Real-time inference latency remained under 10 ms, confirming its reliability in diverse user and environmental contexts.

Focusing on facial cues, [138] introduced a camera-based wearable system that detects fatigue through micro-expression analysis. A CNN was trained on a custom dataset of facial recordings from 10 participants, targeting blink rate, yawn frequency, and head posture as key indicators. The architecture consisted of two convolutional layers with ReLU activation and was optimized through quantization for deployment on the Arduino Nano 33 BLE Sense. Final performance metrics included 92% classification accuracy, inference latency under 120 ms, and a compact memory footprint suitable for microcontroller execution. This work demonstrates a non-intrusive, real-time fatigue monitoring solution tailored for intelligent transportation systems.

2) AAVs (DRONES) AND AERIAL MOBILITY

a: RESOURCE ALLOCATION

To improve driver safety and network efficiency in Cellular Vehicle-to-Everything (CV2X) environments, [143] proposed a unified framework integrating AAV-assisted communication, cooperative NOMA (Non-Orthogonal Multiple Access), and TinyML. The resource allocation strategy involves partial cooperation between AAVs and ground vehicles, aiming to reduce communication latency and enhance throughput. As a practical use case, a drowsiness detection system was developed using the Arduino Nano 33 BLE Sense, where a TinyML model trained via Edge Impulse on time-series IMU data classified driver states as alert or drowsy with 96.7% accuracy. The final model occupied less than 30 KB of memory and had inference latency under 100 ms, illustrating the effectiveness of lightweight AI for real-time vehicular safety applications within a broader intelligent network infrastructure.

Extending the role of TinyML in vehicular networks, [144] introduced a dual-objective optimization framework that balances AAV energy efficiency with vehicular risk minimization. A hybrid metaheuristic algorithm, combining Artificial Bee Colony (ABC) and Cuckoo Search (CS) methods, was used to dynamically adapt AAV operations based on road and traffic conditions. A TinyML classifier deployed on the Arduino Nano 33 BLE Sense was trained to identify risky driving behaviors such as sharp turns and abnormal acceleration from IMU data. The embedded model achieved 97.2% accuracy, with inference latency under 80 ms and memory usage below 40 KB, making it suitable for continuous deployment in energy-constrained AAV systems.

Focusing on AAV cybersecurity, [145] presented Semantic TinyIDS, a compact intrusion detection system designed to detect container escape attacks on embedded platforms. This approach employs semantic encoding of system calls using n-grams and Word2Vec embeddings to capture behavioral patterns without inflating model complexity. Evaluated on the ADFA-LD dataset, the model achieved 97.8% accuracy, required only 27.5 KB of memory, and maintained inference

times below 75 ms on the Arduino Nano 33 BLE Sense. Compared to traditional n-gram-based IDS methods, Semantic TinyIDS improved the true positive rate by 12.6% and reduced false alarms.

b: NAVIGATION AND SURVEILLANCE

A grid-partitioned TinyML architecture was introduced in [146] to support real-time AAV tracking and environmental data analysis in hydraulic surveying systems. The monitored terrain is divided into spatial grids, each associated with a TinyML-enabled drone node performing localized inference. A 1D-CNN model was trained on multi-modal sensor data comprising wind force, vibration, and GPS coordinates and deployed on an Arduino Nano 33 BLE Sense using quantization. The system achieved 94.6% classification accuracy for AAV tracking anomalies and hydro-environmental patterns, with a model size under 32 KB and inference latency below 85 ms. This grid-aware strategy effectively reduced communication overhead and energy consumption, enabling longer AAV mission endurance and scalable coverage.

To address urban surveillance needs, [147] proposed a four-layer TinyML-driven AAV framework for smart city traffic monitoring, prioritizing energy efficiency and cybersecurity. The architecture consists of: (1) a Traffic Layer for real-time detection of vehicles and pedestrians, (2) a AAV Layer for sensing and data relay, (3) a Security Layer leveraging lightweight, quantized ML models including Decision Tree, ANN, and SVM for detecting DoS attacks via packet-level telemetry, and (4) a Controller Layer that executes responsive actions based on inferences. Quantization using TensorFlow Lite and ONNX significantly reduced model sizes and improved runtime performance. The ANN model achieved 95.5% accuracy with a 91% size reduction and a 3× speedup, while the SVM and Decision Tree models retained over 91% accuracy despite mixed changes in inference latency.

A comparative deployment study in [148] evaluated TinyML model performance across three embedded platforms Arduino Nano 33 BLE Sense, Raspberry Pi Pico, and Raspberry Pi Zero 2 W for AAV-based inference tasks. Two use cases were considered: audio classification (ESC-50 dataset) and image recognition (CIFAR-10). Quantized MobileNet models, compiled via TVM and TensorFlow Lite, were used for deployment. The Arduino Nano 33 BLE Sense provided 92.6% accuracy for audio and 84.1% for image classification, with energy usage averaging 56.9 mJ per inference. Raspberry Pi Zero 2 W offered the fastest inference (9.4 ms) at the cost of higher energy (>400 mJ/inference), while Raspberry Pi Pico delivered a favorable trade-off between accuracy and energy efficiency (<100 mJ/inference).

3) TRAFFIC MANAGEMENT AND INFRASTRUCTURE

A real-time adaptive traffic light control system was developed in [149] using TinyML to dynamically manage intersection flow and reduce vehicle wait times. The solution employs an Arduino Nano 33 BLE Sense paired with ultrasonic sensors to collect traffic density data. A decision tree model, trained via Edge Impulse on simulated traffic scenarios (low, medium, and high density), was quantized and deployed for on-device inference. The resulting model reached an accuracy of 97.3%, with a compact footprint of 18.2 KB and sub-85 ms inference latency. Field simulations showed a 32% decrease in average vehicle waiting time, confirming the model's effectiveness under dynamic traffic conditions. To address road surface quality monitoring, [150] presented a low-power edge system for anomaly detection using unsupervised learning. Vibration data captured by an IMU on an Arduino Nano 33 BLE Sense during vehicle operation was processed using an autoencoder trained on normal driving behavior. High reconstruction error flagged events such as potholes and bumps. Quantized via TensorFlow Lite, the model achieved 94.8% reconstruction accuracy, a size under 40 KB, and an inference latency below 60 ms. The fully embedded system operated offline, detecting and logging anomalies without cloud connectivity or heavy computation, making it ideal for scalable infrastructure monitoring.

An alternative approach using Binary Neural Networks (BNNs) was proposed in [152] for ultra-efficient road anomaly detection on constrained edge platforms. Accelerometer data collected from in-vehicle sensors under normal and abnormal driving conditions was used to train a BNN with 1-bit weights and activations. Leveraging the Larq library and TensorFlow Lite for quantization and deployment, the model achieved 93.5% detection accuracy, required under 20 KB of memory, and delivered inferences within 42 ms. Compared to full-precision counterparts, the BNN offered over 85% reductions in memory and compute load while maintaining robust performance.

4) RAIL VEHICLE

The authors in [153] proposed a low-power, on-board monitoring system that uses TinyML to classify rail vehicle operating states such as acceleration, braking, and cruising. The system leverages an Arduino Nano 33 BLE Sense equipped with an IMU to collect real-time motion data, which is processed by a 1D-CNN trained via Edge Impulse. The model was trained on time-series data from actual train runs, and after quantization, it achieved classification accuracy of 98.7%, with inference latency under 80 ms and a memory footprint of 38 KB. The deployed model enables onboard edge intelligence, reducing the need for continuous communication with central systems and enhancing the responsiveness and scalability of smart railway infrastructures.

5) MICROMOBILITY AND PERSONAL VEHICLES

a: ELECTRIC BICYCLES AND PMVs (PERSONAL MOBILITY VEHICLES)

A real-time, low-power battery health monitoring system for electric bicycles (e-bikes) was introduced in [154], leveraging

embedded neural networks to estimate the State of Health (SoH) of lithium-ion batteries. The authors trained four fully connected neural network architectures ranging from one to four hidden layers on battery discharge datasets, then deployed the quantized int8 models on microcontrollers such as the Arduino Nano 33 BLE Sense, ESP32, and Portenta H7. The optimal model configuration {128, 64, 32, 16} yielded a mean absolute error (MAE) of 0.0003, mean squared error (MSE) of 1.0×10^{-5}, and a coefficient of determination (R^2) of 0.99. With a latency of just 109.3 μ s and a memory footprint below 80 KB, the system demonstrated excellent performance in both energy efficiency and embedded suitability, offering a practical solution for micromobility systems.

In [155], a lightweight TinyML-based system was proposed for fall detection in electric scooter users, targeting real-time, on-device operation. The solution employed an Arduino Nano 33 BLE Sense to collect inertial data via accelerometers and gyroscopes, which was processed using a 1D CNN trained on Edge Impulse. The dataset included both normal riding and simulated fall scenarios. After quantization with TFLM, the final model achieved 97.3% accuracy and an F1-score of 0.96, with an inference latency of 81 ms and memory consumption under 40 KB. Upon detecting a fall, the device communicated with a paired smartphone via Bluetooth to initiate alerts and transmit geolocation data.

b: AUTONOMOUS VEHICLES

To enhance the reliability of neural networks in resource-constrained autonomous mini-vehicles, the authors in [156] proposed a framework that integrates steering control with out-of-distribution (OOD) detection. A CNN was trained on image data for trajectory prediction and quantized using TensorFlow Lite for deployment on microcontrollers such as the Arduino Nano 33 BLE Sense. A signal-level monitoring mechanism, based on autoencoder reconstruction errors, was introduced to detect anomalous inputs and trigger predefined safe behaviors. The CNN achieved 96.2% accuracy on a benchmark indoor track, with an inference latency of 87 ms and a memory footprint below 45 KB. In [157], an energy-efficient navigation system was presented for real-time autonomous driving using a TinyML-based CNN on the ultra-low-power GAP8 microcontroller. The model was trained via a closed-loop imitation learning approach, replicating a conventional computer vision algorithm under optimal lighting conditions. Once deployed, the system maintained accurate lane following even under poor lighting, improving navigation success rates by 20% in fast acquisition scenarios. The inference consumed only 3.9 μ J and achieved a 21× speedup compared to conventional microcontrollers such as STM32L476 and NXP K64f.

For in-vehicle cybersecurity, [158] introduced a lightweight intrusion detection system (IDS) capable of real-time monitoring of Controller Area Network (CAN) traffic using TinyML. An LSTM-based model was trained to detect anomalies including Denial-of-Service (DoS) and spoofing attacks, based on labeled CAN traffic data. After quantization and deployment on the Arduino Nano 33 BLE Sense via TFLM, the IDS achieved a detection accuracy of 96.5%, a false positive rate below 2.1%, and an inference latency under 90 ms, while maintaining a memory usage under 50 KB.

6) EDGE OPTIMIZATION AND GENERAL FRAMEWORKS

The authors in [129] proposed the Tiny Anomaly Compress (TAC) algorithm, a lightweight, unsupervised data compression method designed for real-time execution on constrained IoT microcontrollers. The solution was deployed on an Arduino Nano 33 BLE Sense board paired with a GPS NEO-6M to monitor vehicle telemetry in a real-world driving environment. The TAC algorithm operates by identifying anomalies within data streams using concepts of eccentricity and typicality, storing only significant deviations via a dynamic "anomaly window." The study evaluated three scenarios: no compression (control), TAC with fixed hyperparameters, and AutoTAC with automatic parameter tuning. Results from a controlled 5 km test route showed that TAC achieved an 85.71% compression ratio (CR) with a mean absolute error (MAE) of 0.3544, while AutoTAC yielded a slightly lower CR of 77.77% but with improved accuracy (MAE = 0.2026). Crucially, the analysis of acquisition times from both Arduino and external scripts confirmed that the embedded compression had negligible impact on system latency.

7) LOGISTICS AND SHIPPING

The authors in [159] proposed an IoT prototype that leverages TinyML and NB-IoT technologies for real-time event classification and transmission during cargo shipping. The system integrates an OpenMV Cam H7 Plus and an Actinius Icarus IoT board, alongside an LSM6DSOX accelerometer featuring an embedded ML core to classify package movement events (e.g., drop, shake, rest, wrong rest) using a J48 decision tree algorithm. The model processes features like variance and peak-to-peak values and performs inference directly on the sensor. Environmental data and classification results are transmitted using MQTT over NB-IoT. The dataset comprised 30 samples per class over five categories, collected at 104 Hz. Evaluation yielded an average accuracy of 99.55%, with precision and recall scores exceeding 99% across all classes.

Table **??** presents a comparative summary of recent TinyML applications in the transportation sector, highlighting key aspects including implemented algorithms, hardware platforms, model performance metrics, size optimizations, and inference times.

B. OBSERVATIONS AND DISCUSSION

TinyML applications in transportation have rapidly expanded in recent years, offering innovative solutions to challenges

TABLE 10. Summary of research work on transportation using TinyML.

Task		Year	Ref.	Algorithms	Performance	Model Size	Inference Time	Hardware
Intelligent Vehicles and Driver Monitoring	Driver Behavior Analysis	2024	[139]	TEDA, AutoCloud k-fix (incremental clustering)	Silhouette: 0.51-0.63, Davies-Bouldin: 0.72-0.89	2.5MB (approx.)	Real-time (1s intervals)	Freematics One+ OBD-II (ESP32)
		2023	[140]	AutoCloud k-fix (optimized)	Dunn Index: 0.85-1.12, DB Index: 0.91-1.15	< 1MB	Real-time stream	OBD-II + Torque Pro App
		2020	[141]	DepthConv-LSTM/GRU	Acc: 98.72%, FLOPs: 0.232M	1.69MB	182-227μs	Jetson Xavier/TX2/Nano
		2025	[142]	YOLOv8 with GhostNet + Head Pose Estimation (OpenCL)	mAP50: > 90%, Acc: > 90%	×	10 FPS	Raspberry Pi Zero 2 W
	Drowsiness / Fatigue Detection	2023	[136]	SqueezeNet, AlexNet, CNN, MobileNet-V2, MobileNet-V3	Acc: 99.64% (MobileNet-V2), 99.51% (SqueezeNet), 99.24% (AlexNet)	0.05MB (CNN) - 1.55MB (MobileNet-V2)	×	STM32 microcontrollers
		2023	[137]	MLP, CNN	Acc: 94.6% (MLP)	3.704 kB (MLP)	25.44ms	nRF52840 MCU (64MHz)
		2025	[138]	BPNN	Acc: 94.35%	×	Batch processing (6000 points)	STM32F407ZGT6 MCU
	Distraction / Impairment Detection	2023	[132]	CNN	Acc: 99.3%, F1-Score: > 95%	52.7 KB	72 ms	Arduino Portenta H7 + Vision Shield
		2022	[131]	Neural Network with Leaky-ReLU	High accuracy	×	Real-time	Wio Terminal (ATSAMD51) + Grove Multichannel Gas Sensor
		2023	[130]	Transfer Learning SqueezeNet 1.1	Acc: 99.93%	4.79MB	Real-time (30 fps)	Raspberry Pi 4B + Logitech Webcam
	Advanced Driver Assistance Systems	2022	[133]	MobileNetV2-SSD	mAP: 60.72, Acc: 98-99%	12.8MB	46 FPS (PC), 3 FPS (TX2)	NVIDIA Jetson TX2 + Multi-view cameras
		2024	[134]	Random Forest + PCA + SMOTE	Acc: 93.8%, F1-Score: 0.65-0.99	×	0.19s (6000 samples)	MEMS/IMU Sensors (Bosch) + Test Vehicle
		2022	[135]	MLP Regressor	MAPE: 27%	10-17x smaller (quantized)	37-173μs	Macchina A0 (ESP32-WROVER-E)
UAVs (Drones) and Aerial Mobility	Resource Allocation	2024	[143]	Federated Learning + Neural Networks	Acc: 80-85%	×	Real-time processing	UAV + C-V2X Infrastructure
		2024	[144]	MLP	High accuracy	1.39 kB	< 1s	UAV embedded system
		2024	[145]	LSTM + Clustering	F1-score: 94.5%	×	Real-time processing	eBPF probes, STM32 MCU
	Navigation and Surveillance	2024	[146]	Improved RetinaNet + Grid-based Tracking	F1-score: 0.92, Recall: 94%	×	10ms radar cycle	Vehicle-mounted radar + PTZ camera
		2025	[147]	ANN, SVM, Decision Tree	Acc: 92-95%, DoS Detection: 94%	677 KB (ANN)	0.000055 s (ANN)	Onboard UAV processors
		2021	[148]	MobileNetV2	Acc: 97%	585 KB	859 ms	DJI Tello + OpenMV H7
Traffic Management and Infrastructure		2021	[149]	Random Forest Regressor + Piezoelectric Sensing	Acc: 95%	1.75 KB (MCU optimized)	< 100 ms	Arduino Uno + Piezoelectric Sensors
		2021	[150]	Unsupervised TEDA Algorithm	F1-score: 0.76-0.78, Acc: 99%	< 1KB (MCU optimized)	Real-time processing (1Hz sampling)	Arduino Nano 33 IoT + Accelerometer
		2024	[152]	CBin-NN	Acc: 78.8%	2.3 KB	0.6 ms	ARM Cortex-M0
Rail Vehicle		2022	[153]	ANN	Acc: 99.7% (6 states)	×	0.5 s intervals	ESP32 microcontroller
Micromobility and Personal Vehicles	Electric Bicycles and PMVs	2023	[154]	ANN	MAE: 0.021, R^2: 0.983	45.2 KB (int8)	18.7 ms	Arduino Portenta H7
		2022	[155]	Random Forest	Acc: 97.17%, F1: 0.9697	45.83 KB	11 μ s	Raspberry Pi Zero W
	Autonomous Vehicles	2021	[156]	tinyCNNs (VNN family) with imitation learning	Acc: 97.4%, Energy: 18.9 μJ	0.48-6.04 KB	0.37-0.91 ms	GAP8 RISC-V MCU
		2020	[157]	tinyCNNs (VNN family) with closed-loop learning	Acc: 98.74%, Throughput: 1000 fps	0.48-6.04 KB	0.37-0.91 ms	GAP8 RISC-V MCU
		2024	[158]	SVM, Shallow Neural Network	Acc: 97.2%, Precision: 95.3%	×	Real-time validation	Arduino Nano 33 BLE Sense
Logistics and Shipping		2024	[159]	J48 Decision Tree	Acc: 99.55%, F1-Score: 0.990-1.000	Minimal (on-sensor processing)	Real-time (104 Hz)	OpenMV Cam + Actinius Icarus IoT
Edge Optimization and General Frameworks		2022	[129]	TAC, AutoTAC	CR: 77.77-85.71%, MAE: 0.2026-0.3544	×	×	Arduino Nano 33 BLE Sense + GPS NEO-6M

such as driver safety, vehicle monitoring, traffic control, and environmental sensing. However, a critical examination of current literature and deployment efforts reveals key trends, limitations, and opportunities for further development.

1) There is a strong emphasis on intelligent vehicle systems and driver monitoring, including behavior analysis, drowsiness detection, and distraction recognition. While this reflects the urgency of addressing road safety, many systems remain limited to proof-of-concept implementations. These solutions are often trained and validated on clean, well-curated datasets that may not generalize well in unstructured or real-world scenarios involving poor lighting, sensor misalignments, or diverse driver behaviors. Additionally, there is limited attention to long-term user adaptation, hardware degradation, or real-world conditions such as vibrations, temperature changes, and user fatigue that could affect system reliability over time. This gap underscores the need for longitudinal field studies and robustness benchmarks before such solutions can be considered production-ready.

2) A notable strength in this domain is the integration of multimodal sensing and efficient embedded models, allowing inferences directly on low-power hardware. Systems that combine IMUs, cameras, and gas sensors have demonstrated the feasibility of edge-based driver behavior classification, fatigue monitoring, and accident detection. However, most studies evaluate power and latency metrics in isolation, without profiling the energy impact of continuous data acquisition, wireless transmission, or peripheral operations. For instance, a fatigue detection system may achieve sub-100 ms inference time, but if it requires high-frequency sampling from multiple sensors or continuous Bluetooth connectivity, the overall energy consumption may negate its suitability for long-term deployment on battery-powered platforms. A more system-level evaluation of power and memory profiles is needed to ensure sustainable operation.
3) The literature on AAVs and aerial mobility showcases promising use cases, especially in real-time surveillance, traffic analytics, and vehicular communication networks. These works benefit from high mobility and overhead perspectives that ground-based systems cannot offer. Yet, many of these applications are constrained to lab-scale or simulated environments, and rarely address external factors like weather variability, GPS interference, or coordination among multiple AAVs in shared airspace. Additionally, regulatory and safety challenges such as flight permissions, collision avoidance, or privacy in densely populated areas are often overlooked, leaving a gap between technical feasibility and regulatory readiness. To bridge this, future work must integrate airspace constraints and ethical design into AAV-TinyML system planning.
4) Traffic infrastructure use cases such as adaptive traffic light control and road anomaly detection highlight TinyML's potential to contribute to smart cities. These systems can operate autonomously at the edge, reducing dependence on cloud infrastructure and enabling real-time local responses. However, scalability remains a concern, as most implementations are demonstrated in single intersections or short road segments. There is limited research on how such systems perform when scaled across districts or integrated into heterogeneous urban networks with legacy infrastructure. In addition, interoperability protocols, deployment costs, and maintenance logistics are seldom discussed, all of which are critical for large-scale, city-wide adoption.
5) Across the board, ethical and regulatory considerations remain insufficiently addressed. Applications such as in-cabin video monitoring, facial expression analysis, and CAN bus surveillance raise significant concerns around privacy, consent, and algorithmic bias. Although TinyML's on-device processing is frequently cited as a privacy-enhancing feature, such claims lack empirical validation. For example, few studies evaluate whether sensitive data is adequately anonymized or if the models introduce bias based on user demographics. Furthermore, there is no consensus on best practices for informed consent, especially in multi-occupant or public settings. Future research should integrate ethical audits, user studies, and transparent documentation to assess social impacts and build trust.
6) Evaluation practices across studies are highly variable. While accuracy and inference latency are commonly reported, they rarely capture system usability, reliability, or robustness under stress conditions (e.g., sensor failure, adversarial input). Few works include operational measures such as false alarm rates, mean time between failures, or user acceptance metrics that are crucial for applications in transportation where safety and reliability are paramount. Moreover, standard datasets and testing protocols are often missing, making comparisons between systems challenging. This lack of uniformity hinders progress toward industry-standard benchmarks for TinyML in transportation.
7) Finally, while there is considerable progress in vehicle-focused TinyML applications, other transportation sectors are underrepresented. Rail systems, maritime logistics, and micromobility platforms (e.g., e-scooters, bicycles) offer substantial opportunities for embedded AI, especially in low-connectivity or resource-constrained settings. These domains often have unique data modalities (e.g., vibration patterns, geospatial telemetry) that could benefit from custom TinyML models. The relative absence of research in these areas suggests an untapped potential for applying TinyML beyond traditional automotive use cases.

C. LESSONS LEARNED AND RECOMMENDATIONS

The review of TinyML applications in transportation reveals numerous technological advances alongside gaps that must be addressed to ensure effective, ethical, and scalable adoption. The following lessons and recommendations are derived from patterns observed across the literature.

1) Bridge the gap between proof-of-concept and deployment. Many studies demonstrate high model performance in controlled environments, yet few transition into field-tested systems. To improve real-world readiness, researchers should prioritize end-to-end validation under diverse, unpredictable conditions. This includes stress testing models across different weather conditions, road surfaces, and user populations. Additionally, incorporating iterative prototyping and continuous feedback from actual end-users (drivers, operators, city planners) will ensure more robust and usable solutions.
2) Adopt holistic system-level optimization, not just model-level tuning. TinyML development often emphasizes reducing model size or inference latency in isolation. However, entire system pipelines including sensor sampling rate, power draw from communication mod-

ules (e.g., BLE, NB-IoT), and memory consumption by preprocessing steps must be evaluated comprehensively. Future work should include full-stack profiling tools and methodologies to ensure sustainable, power-efficient operation in embedded platforms over extended periods.

3) Standardize benchmarking protocols and evaluation criteria. The diversity of models, datasets, and metrics makes it difficult to compare TinyML systems fairly. Establishing standardized benchmarks (e.g., specific transportation datasets for driver monitoring, traffic anomalies, or AAV navigation) and reporting protocols including operational metrics like false positive rates, uptime, or inference jitter would facilitate cross-study comparisons and accelerate innovation. This is particularly critical for safety-critical domains where reproducibility and reliability are non-negotiable.
4) Expand focus to underexplored transportation modes. The literature is heavily biased toward automotive applications, while rail, maritime, cargo logistics, and micromobility sectors receive comparatively little attention. These domains often face constraints such as limited connectivity, irregular maintenance cycles, or rugged environments conditions where TinyML could be especially impactful. Researchers should explore domain-specific sensing modalities and collaborate with industry stakeholders in these overlooked sectors to co-design relevant, deployable systems.
5) Integrate ethical, legal, and social considerations from the design phase. While TinyML enables on-device inference that reduces reliance on the cloud, this does not inherently guarantee privacy or fairness. Systems that involve video or biometric data must address consent, data retention, and potential bias proactively. Multidisciplinary collaboration with ethicists, legal experts, and community representatives should become standard in the development pipeline, especially for applications that could impact civil liberties or involve vulnerable populations.
6) Incorporate user-centered design for better acceptance and adoption. The success of a TinyML system is not only technical but also behavioral whether it is trusted, understood, and adopted by end-users. Developers should involve drivers, fleet managers, and public officials early in the design process to ensure usability, accessibility, and transparency. Features like intuitive alerts, configurable thresholds, and meaningful feedback can greatly influence the real-world effectiveness of TinyML-based safety or monitoring systems.
7) Encourage open hardware and open dataset ecosystems. Many TinyML transportation studies rely on proprietary hardware or private datasets, which limits reproducibility and collaborative progress. Sharing annotated datasets (especially sensor data collected in real environments), pre-trained models, and edge-ready firmware can lower entry barriers and support global research, especially in regions lacking access to high-end computing or sensors.

VII. OPEN CHALLENGES AND RESEARCH OPPORTUNITIES

While TinyML has enabled energy-efficient machine learning on edge devices, the technology still faces critical limitations that hinder its deployment at scale and in real-world environments. The following subsections explore the most pressing open challenges and articulate forward-looking research directions grounded in current system constraints, emerging paradigms, and practical feasibility (see Figure 9).

A. MISALIGNMENT BETWEEN MODEL DESIGN AND HARDWARE CONSTRAINTS

TinyML models are frequently developed in isolation from the hardware they target, resulting in inefficient execution patterns, unpredictable memory usage, and energy overheads. Post-training quantization, pruning, or architecture simplification are often applied as afterthoughts [196], missing opportunities for true hardware-aware optimization. This creates friction during deployment, especially when models need to fit within the limited SRAM, cache line sizes, and memory access bandwidths of microcontrollers or SoCs.

Future research should shift toward integrated hardware–software co-design through microcontroller-aware neural architecture search (MCU-NAS). Such a framework should simultaneously optimize for performance, accuracy, and energy by embedding hardware profiles such as DMA access latency [212], memory alignment boundaries, and peripheral interrupt latencies into the model design loop. Moreover, toolchains like CMSIS-NN, TVM-micro, and Glow should be extended to support operator fusion, memory reuse strategies, and zero-copy inference execution. By adopting these approaches, researchers could achieve significant reductions in peak current draw, improve throughput predictability, and enable more robust deployments for real-time applications such as autonomous mobility or medical wearables.

B. UNDERUTILIZATION OF EVENT-DRIVEN AND NEUROMORPHIC PARADIGMS

Current TinyML implementations mostly rely on periodic sampling and dense inference schedules, regardless of whether new information is present in the input stream. This leads to wasted energy [1], especially in scenarios where meaningful events are sparse or irregular common in environmental monitoring, behavioral tracking, or medical sensing. Event-driven computing and neuromorphic models like spiking neural networks (SNNs) remain underexplored in embedded contexts despite their inherent suitability for sparse, asynchronous data.

Future work should explore hybrid pipelines that integrate event-triggered sampling, asynchronous sensor readouts, and temporal coding schemes such as rate or latency-based spike encoding [213]. Embedding

FIGURE 9. Open challenges and research opportunities.

these into neuromorphic-compatible microcontrollers (e.g., Loihi [214], Intel's neuromorphic chip [215], or ultra-low-power analog compute-in-memory arrays [216]) could reduce inference energy by orders of magnitude. Moreover, software frameworks need to support spiking models with efficient training pipelines, spike-based loss functions [217], and interpretability methods that work at the microcontroller level [218]. Use cases such as gesture recognition from event cameras [110], anomaly detection in ECG signals, or wildlife activity sensing in remote deployments could benefit directly from these advancements.

C. CHALLENGES IN SECURE FEDERATED LEARNING FOR TinyML

While federated learning holds promise for privacy-preserving model updates across distributed devices, conventional approaches remain infeasible on resource-constrained microcontrollers [220]. The memory and compute demands of gradient aggregation, update synchronization, and secure encryption protocols are far beyond what typical TinyML endpoints can support. Furthermore, communication latency and power usage from frequent transmissions degrade the viability of FL in practice.

To address this, future research should explore techniques such as split learning [221], where only partial model layers are executed on the device, offloading deeper computations to gateways or edge servers. Another direction is to adopt knowledge distillation using compact teacher-student frameworks that only transmit high-level representations instead of raw gradients [222]. Secure aggregation protocols [223] should also be revisited with microcontroller constraints in mind, possibly using lightweight homomorphic hashing [224], sketching methods [225], or compressed sparse updates [226].

D. LACK OF ON-DEVICE EXPLAINABILITY AND MODEL ASSURANCE MECHANISMS

As TinyML enters safety-critical domains like healthcare, transportation, and defense, the need for trustworthy and interpretable decision-making grows. Unfortunately, the most widely used interpretability methods such as SHAP, LIME [227], or Integrated Gradients are computationally intensive and require post-hoc processing on high-end devices [228]. There is a fundamental gap between the explainability expectations of end-users and the capabilities of TinyML platforms.

Future directions should focus on embedding simplified surrogate models, rule-extraction techniques, or activation-based confidence scoring directly into the inference path [229]. For example, linear approximations or decision-tree mimics could serve as interpretable proxies alongside more complex models. Additionally, uncertainty estimation frameworks such as Monte Carlo dropout or Bayesian neural networks should be adapted to fit within kilobyte-scale RAM limits [230]. Combining these with lightweight model validation mechanisms, such as runtime bounds checking or formal interval analysis [231], could support real-time model assurance in constrained deployments. This would be particularly impactful in regulated fields like clinical diagnostics, drug dispensing, or autonomous navigation, where every output may have legal or safety implications.

E. DEFICIT OF EMBEDDED-FIRST DATASETS AND EVALUATION METRICS

Most TinyML research still relies on public datasets collected in ideal lab conditions using high-resolution, high-frequency sensors. When these models are transferred to embedded systems, they often encounter degraded signal quality, poor lighting, limited sampling rates, or hardware noise, leading to a steep drop in real-world performance [1], [194]. Furthermore, conventional metrics such as accuracy or F1-score fail to capture deployment-specific constraints like robustness to environmental shifts, sensor failure, or latency jitter.

Future work should prioritize the development of embedded-first datasets collected under constrained acquisition settings (e.g., low-resolution thermal cameras, grayscale image sensors, or intermittent IMU readouts). Data augmentation [232] should go beyond rotation or cropping and instead simulate deployment conditions through temporal jittering, sensor dropout, or quantization noise. Evaluation protocols must also evolve to consider holistic metrics such as total energy per correct inference, real-time decision latency under multi-threaded loads, and long-term drift in sensor calibration [198], [199]. Creating community benchmarks and leaderboards tailored to TinyML deployments will help establish fair comparisons and stimulate innovation toward real-world robustness [197].

F. INCOMPLETE ENERGY AND LIFECYCLE PROFILING FRAMEWORKS

While many TinyML papers report model inference time or power consumption during idle inference, few account for the full energy cost of sensing, preprocessing, wake-up triggers, communication, and battery self-discharge [194], [195]. In reality, the lifecycle energy of a TinyML system depends heavily on how often it wakes up, how much preprocessing is offloaded to sensor fusion ICs, and whether inference is event-triggered or timer-based. These gaps create over-optimistic assumptions about battery life and sustainability [199].

To close this gap, future research must introduce end-to-end energy modeling frameworks that account for all pipeline stages from sensor wake-up to inference to sleep re-entry. Power-tracing hardware such as Joulescope or Monsoon Power Monitors should be integrated into development workflows [233]. Researchers should also simulate lifecycle usage under different duty cycles, sensor degradation rates, and environmental changes. Such tools would allow designers to make informed trade-offs between sampling frequency, model complexity, and transmission costs.

G. REGULATORY, ETHICAL, AND SOCIAL INTEGRATION CHALLENGES

The increasing deployment of TinyML in healthcare, transportation, and public surveillance brings the technology into close contact with legal and ethical frameworks most of which are not yet designed to accommodate embedded AI [194]. Challenges include consent acquisition in passive monitoring systems, compliance with data minimization laws, algorithmic bias, and unclear liability in case of system failure [1]. These sociotechnical hurdles are no longer secondary; they directly affect whether TinyML solutions are adopted or rejected [197], [198], [199].

Future directions should focus on embedding privacy-preserving mechanisms at the firmware level, such as on-device anonymization, zero-retention buffers, and differential privacy guards [234]. Collaboration between engineers, legal experts, and ethicists should lead to the creation of standardized guidelines and certification procedures specific to embedded AI. Prototypes should be deployed in real-world regulatory sandboxes such as smart cities or clinical trial testbeds to explore edge cases and inform policymaking [235]. Additionally, participatory design methods should be applied, involving end-users in the design of user interfaces, alert mechanisms, and feedback loops. These practices can improve trust, usability, and ethical compliance across applications.

H. SUMMARY

Each challenge presented here represents not just a limitation, but an invitation for meaningful, multidisciplinary progress. The next phase of TinyML research will require tighter hardware-software integration, new paradigms in model design, robust real-world testing frameworks, and proactive societal alignment. By addressing these challenges with concrete, forward-looking strategies, the TinyML community can ensure reliable, ethical, and scalable AI at the extreme edge.

VIII. CONCLUSION AND FUTURE WORKS

This review has systematically explored the landscape of TinyML applications across healthcare, education, and transportation, offering a targeted synthesis of 136 publications from 2020 to 2025. Our work analyzed technical dimensions such as deployment frameworks, hardware platforms, model compression strategies, and evaluation metrics while emphasizing domain-specific challenges. We highlighted the increasing feasibility of deploying machine learning at the ultra-low-power edge, driven by advancements in microcontroller-based inference and specialized software stacks. The review emphasized how TinyML supports real-time, privacy-preserving analytics with constrained resources yet noted persisting limitations in scalability, energy profiling, dataset availability, and explainability. By structuring the survey around concrete application domains, we delivered a practical and comparative reference that not only identifies trends but also exposes critical gaps in current TinyML adoption.

In our future work, we intend to address these limitations by developing domain-specific benchmarks, hardware-aware model design pipelines, and reproducible reference implementations for TinyML applications in healthcare, education,

and transportation. Specifically, we will focus on creating curated datasets that reflect real-world noise, latency, and power constraints, as well as investigating lightweight federated learning techniques adapted to resource-constrained devices. Additionally, we aim to publish open-source tools, including pre-quantized models and deployment templates for MCUs like STM32 and ESP32, to support researchers and developers working on embedded AI. Our goal is to close the gap between academic advancements and field-deployable systems by enabling accessible, low-power machine learning that is robust, explainable, and contextually adaptive across mission-critical domains.

REFERENCES

[1] I. Lamaakal, S. Essahraui, Y. Maleh, K. E. Makkaoui, I. Ouahbi, M. F. Bouami, A. A. A. El-Latif, M. Almousa, J. Peng, and D. Niyato, "A comprehensive survey on Tiny machine learning for human behavior analysis," *IEEE Internet Things J.*, vol. 12, no. 16, pp. 32419–32443, Aug. 2025, doi: 10.1109/JIOT.2025.3565688.

[2] A. P. Sampaio, P. C. M. A. Farias, and R. A. Bittencourt, "A case study of using machine learning in K-12 education," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, College Station, TX, USA, Oct. 2023, pp. 1–8.

[3] A. Paul, N. Rakhaine, N. J. Ohee, and A. Ahammad, "A comprehensive Android app based solution for automated attendance and management in institutions using IoT and TinyML," in *Proc. Int. Conf. Inf. Commun. Technol. Sustain. Develop. (ICICT4SD)*, Sep. 2023, pp. 382–387.

[4] S. Liu, "Edge computing enables assessment of student community building: An emotion recognition method based on TinyML," *Internet Technol. Lett.*, vol. 8, no. 2, p. 645, Mar. 2025.

[5] A. Debbarma and S. Sinha, "TinyML based physical computing for education," in *Proc. IEEE 19th India Council Int. Conf. (INDICON)*, Nov. 2022, pp. 1–4.

[6] S. Cui, "Context-aware TinyML model for Korean handwriting recognition under online assisted learning scenes," *Internet Technol. Lett.*, vol. 8, no. 6, p. 636, Nov. 2025.

[7] A. F. Cotrino Herrera, J. A. López Sotelo, J. C. Blandón Andrade, and A. T. Lazo, "Low-cost prototype for bearing failure detection using tiny ML through vibration analysis," *HardwareX*, vol. 22, Jun. 2025, Art. no. e00658.

[8] S. Jiang, "TinyML based edge intelligent English classroom quality assessment scheme," *Internet Technol. Lett.*, vol. 8, no. 4, p. 70072, Jul. 2025.

[9] I. Lamaakal, K. E. Makkaoui, I. Ouahbi, and Y. Maleh, "A TinyML model for gesture-based air handwriting Arabic numbers recognition," *Proc. Comput. Sci.*, vol. 236, pp. 589–596, Jan. 2024.

[10] I. Lamaakal, N. El Mourabit, K. El Makkaoui, I. Ouahbi, and Y. Maleh, "Efficient gesture-based recognition of tifinagh characters in air handwriting with a TinyDL model," in *Proc. 6th Int. Conf. Intell. Comput. Data Sci. (ICDS)*, Oct. 2024, pp. 1–8.

[11] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.

[12] S. Hammoudi, Z. Aliouat, and S. Harous, "Challenges and research directions for Internet of Things," *Telecommun. Syst.*, vol. 67, no. 2, pp. 367–385, 2017.

[13] S. C. Mukhopadhyay and N. K. Suryadevara, "Internet of Things: Challenges and Opportunities," in *Internet of Things* (Smart Sensors, Measurement and Instrumentation), vol. 9, S. Mukhopadhyay, Ed., Cham, Switzerland: Springer, 2014, doi: 10.1007/978-3-319-04223-7_1.

[14] E. Kim, J. Kim, J. Park, H. Ko, and Y. Kyung, "TinyML-based classification in an ECG monitoring embedded system," *Comput., Mater. Continua*, vol. 75, no. 1, pp. 1751–1764, 2023.

[15] C. V. Vivekanand, R. Purushothaman, S. C. Kushbu, and M. A. J. Selvam, "Tiny ML-based non-invasive approach of cardiac monitoring," in *Proc. 7th Int. Conf. Contemp. Comput. Informat. (IC3I)*, Sep. 2024, pp. 529–534.

[16] V. Gautam, S. Sinha, and S. Prasad, "VATML: Towards on device ventricular arrhythmia detection using TinyML," in *Proc. IEEE Int. Symp. Smart Electron. Syst. (iSES)*, Dec. 2024, pp. 1–6.

[17] G. V. Silva, M. D. Lima, J. A. Filho, and M. J. Rovai, "Atrial fibrillation and sinus rhythm detection using TinyML (embedded machine learning)," in *Proc. Latin Amer. Conf. Biomed. Eng.*, Oct. 2022, pp. 633–644.

[18] P. Edouard and D. Campo, "Design and validation of withings ECG software 2, a tiny neural network based algorithm for detection of atrial fibrillation," *Comput. Biol. Med.*, vol. 185, Feb. 2025, Art. no. 109407.

[19] S. Iqbal, X. Zhong, M. Alhussein, Z. Wu, K. Aurangzeb, W. Liu, and Y. Zhang, "FusionGCNN: An IoT-based novel spatiotemporal graph convolutional network for ECG arrhythmia detection," *IEEE Internet Things J.*, vol. 12, no. 22, pp. 46038–46050, Nov. 2025.

[20] B. Bengherbia, M. R. Aymene Berkani, Z. Achir, A. Tobbal, M. Rebiai, and M. Maazouz, "Real-time smart system for ECG monitoring using a one-dimensional convolutional neural network," in *Proc. Int. Conf. Theor. Appl. Comput. Sci. Eng. (ICTASCE)*, Sep. 2022, pp. 32–37.

[21] M. Meza-Rodriguez, L. De La Cruz, and J. A. Cáceres-Delaguila, "Development of an electrocardiographic signal classifier for bundle branch blocks, applying tiny machine learning," in *Proc. IEEE 30th Int. Conf. Electron., Electr. Eng. Comput. (INTERCON)*, Nov. 2023, pp. 1–6.

[22] Y.-S. Agrignan, S. Zhou, J. Bai, S. Islam, S. Nabavi, M. Xie, and C. Ding, "A deep learning approach for ventricular arrhythmias classification using microcontroller," in *Proc. 24th Int. Symp. Quality Electron. Design (ISQED)*, Apr. 2023, pp. 1–5.

[23] C. Hwang, J. So, J. Rhe, J. Kim, J. Park, K. E. Jeon, and J. H. Ko, "An efficient ventricular arrhythmias detection on microcontrollers with optimized 1D CNN," in *Proc. IEEE 6th Int. Conf. AI Circuits Syst. (AICAS)*, Apr. 2024, pp. 572–576.

[24] Y. R. Thota and T. Nikoubin, "TinyML for ECG biometrics on resource constrained devices," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2024, pp. 521–526.

[25] S. Mukhopadhyay, S. Dey, A. Ghose, P. Singh, and P. Dasgupta, "Generating tiny deep neural networks for ECG classification on microcontrollers," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Affiliated Events (PerCom Workshops)*, Mar. 2023, pp. 392–397.

[26] A. Ukil, I. Sahu, A. Majumdar, S. C. Racha, G. Kulkarni, A. D. Choudhury, S. Khandelwal, A. Ghose, and A. Pal, "Resource constrained CVD classification using single lead ECG on wearable and implantable devices," in *Proc. 43rd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Nov. 2021, pp. 886–889.

[27] S. Rahman, Y. A. Khan, Y. Pratap Singh, S. A. Ali, and M. Wajid, "TinyML based classification of fetal heart rate using Mother's abdominal ECG signal," in *Proc. 5th Int. Conf. Multimedia, Signal Process. Commun. Technol. (IMPACT)*, Nov. 2022, pp. 1–5.

[28] Z. Jia, D. Li, C. Liu, L. Liao, X. Xu, L. Ping, and Y. Shi, "TinyML design contest for life-threatening ventricular arrhythmia detection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 1, pp. 127–140, Jan. 2024.

[29] D. Luong, B. N. Quang, L. Nguyen, and M. N. Ngoc, "TinyML-based system for cost-effective ECG rhythm classification on embedded device," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2024, pp. 209–214.

[30] R. Banerjee and A. Ghose, "A light-weight deep residual network for classification of abnormal heart rhythms on tiny devices," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, Sep. 2022, pp. 317–331.

[31] B. Sun, S. Bayes, A. M. Abotaleb, and M. Hassan, "The case for TinyML in healthcare: CNNs for real-time on-edge blood pressure estimation," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, Mar. 2023, pp. 629–638.

[32] K. Ahmed and M. Hassan, "TinyCare: A TinyML-based low-cost continuous blood pressure estimation on the extreme edge," in *Proc. IEEE 10th Int. Conf. Healthcare Informat. (ICHI)*, Jun. 2022, pp. 264–275.

[33] N. F. Ali, M. Hussein, F. Awwad, and M. Atef, "Convolutional autoencoder for real-time PPG based blood pressure monitoring using TinyML," in *Proc. Int. Conf. Microelectron. (ICM)*, Dec. 2023, pp. 41–45.

[34] M. O.-E. Aoueileyine, "Tiny machine learning for IoT and eHealth applications: Epileptic seizure prediction use case," in *Proc. Int. Conf. Digit. Technol. Appl.*, Jan. 2023, pp. 242–251.

[35] E. Tsakanika, V. Tsoukas, A. Kakarountas, and V. Kokkinos, "High accuracy of epileptic seizure detection using tiny machine learning technology for implantable closed-loop neurostimulation systems," *BioMedInformatics*, vol. 5, no. 1, p. 14, Mar. 2025.

[36] Y. B. Dhiab, M. Hizem, N. Karmous, M. O. E. Aoueileyine, and R. Bouallegue, "An IoT-based multimodal wearable framework for real-time epileptic seizures detection using TinyML," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2025, pp. 70–80.

[37] R. Shankar, K. Aadarsh, and G. K. Chellamani, "Epilepsy detection using embedded machine learning," in *Proc. 9th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, vol. 1, Mar. 2023, pp. 1914–1922.

[38] S. Qiu, W. Wang, and H. Jiao, "LightSeizureNet: A lightweight deep learning model for real-time epileptic seizure detection," *IEEE J. Biomed. Health Informat.*, vol. 27, no. 4, pp. 1845–1856, Apr. 2023.

[39] S. Soumya, P. S. Aithal, and N. N. Bappalige, "TinyML for Alzheimer's research: Harnessing neural networks for insights and diagnostics," *Future Trends Inf., Commun. Comput. Technol.*, vol. 67, no. 3, p. 773, 2017.

[40] A. Tragoudaras, C. Antoniadis, and Y. Massoud, "TinyML for EEG decoding on microcontrollers," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2023, pp. 1–5.

[41] O. Chepyk, A. Bruna, M. Campolo, N. Mammone, F. C. Morabito, G. Ruggeri, and V. Tomaselli, "An embedded EOG-based brain computer interface system for robotic control," in *Proc. 8th Int. Conf. Smart Sustain. Technol. (SpliTech)*, Jun. 2023, pp. 1–6.

[42] M. Hashir, N. Khalid, N. Mahmood, M. A. Rehman, M. Asad, M. Q. Mehmood, M. Zubair, and Y. Massoud, "A TinyML based portable, low-cost microwave head imaging system for brain stroke detection," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2023, pp. 1–4.

[43] Y. Abadade, N. Benamar, M. Bagaa, and H. Chaoui, "Empowering healthcare: TinyML for precise lung disease classification," *Future Internet*, vol. 16, no. 11, p. 391, Oct. 2024.

[44] S. O. Ooko and J. Nsenga, "Tiny machine learning (TinyML) based self diagnostic kit for respiratory diseases," in *Proc. Int. Conf. Comput. Appl. (ICCA)*, Nov. 2023, pp. 1–6.

[45] S. O. Ooko, D. Mukanyiligira, J. P. Munyampundu, and J. Nsenga, "Synthetic exhaled breath data-based edge AI model for the prediction of chronic obstructive pulmonary disease," in *Proc. Int. Conf. Comput. Commun. Appl. Technol. (I3CAT)*, Sep. 2021, pp. 1–6.

[46] S. O. Ooko, D. Mukanyiligira, J. P. Munyampundu, and J. Nsenga, "Edge AI-based respiratory disease recognition from exhaled breath signatures," in *Proc. IEEE Jordan Int. Joint Conf. Electr. Eng. Inf. Technol. (JEEIT)*, Nov. 2021, pp. 89–94.

[47] O. Elallam, O. Jami, and M. Zaki, "Tiny-ML and IoT based early COVID19 detection wearable system," in *Proc. Int. Conf. Smart Appl. Data Anal.*, Apr. 2024, pp. 268–280.

[48] A. Rana, Y. Dhiman, and R. Anand, "Cough detection system using TinyML," in *Proc. Int. Conf. Comput., Commun. Power Technol. (IC3P)*, Jan. 2022, pp. 119–122.

[49] R. Shankar, K. M. Gautham Mythireyan, N. R. Nalla, and M. Venkateshkumar, "Cough recognition using tiny ML," in *Proc. IEEE Ind. Electron. Appl. Conf. (IEACon)*, Oct. 2022, pp. 111–116.

[50] L. Arief, M. Risky, Derisma, W. Kasoep, and N. Puteri, "Portable cough classification system based on sound feature extraction using tiny machine learning," *Indonesian J. Comput. Sci.*, vol. 10, no. 2, pp. 201–225, Oct. 2021.

[51] S. P. Jena, P. G. Deepika, G. V. H. Prasad, and S. Chakravarty, "Enhancing healthcare with edge AI for analysis of cough detection," in *Proc. IEEE 9th Int. Women Eng. (WIE) Conf. Electr. Comput. Eng. (WIECON-ECE)*, Nov. 2023, pp. 1–6.

[52] M. S. Diab and E. Rodriguez-Villegas, "A TinyML motion-based embedded cough detection system," in *Proc. 46th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2024, pp. 1–4.

[53] Y. Wang, X. Zhang, J. M. Chakalasiya, X. Xu, Y. Jiang, Y. Li, S. Patel, and Y. Shi, "HearCough: Enabling continuous cough event detection on edge computing hearables," *Methods*, vol. 205, pp. 53–62, Sep. 2022.

[54] A. Gudiño-Ochoa, J. A. García-Rodríguez, R. Ochoa-Ornelas, J. I. Cuevas-Chávez, and D. A. Sánchez-Arias, "Noninvasive diabetes detection through human breath using TinyML-powered E-nose," *Sensors*, vol. 24, no. 4, p. 1294, Feb. 2024.

[55] A. Gudiño-Ochoa, J. A. García-Rodríguez, J. I. Cuevas-Chávez, R. Ochoa-Ornelas, A. Navarrete-Guzmán, C. Vidrios-Serrano, and D. A. Sánchez-Arias, "Enhanced diabetes detection and blood glucose prediction using TinyML-integrated E-nose and breath analysis: A novel approach combining synthetic and real-world data," *Bioengineering*, vol. 11, no. 11, p. 1065, Oct. 2024.

[56] M. Zeynali, K. Alipour, B. Tarvirdizadeh, and M. Ghamari, "Non-invasive blood glucose monitoring using PPG signals with various deep learning models and implementation using TinyML," *Sci. Rep.*, vol. 15, no. 1, p. 581, Jan. 2025.

[57] A. Sabatini, C. Cenerini, L. Vollero, and D. Pau, "Calibrating glucose sensors at the edge: A stress generation model for tiny ML drift compensation," *BioMedInformatics*, vol. 4, no. 2, pp. 1519–1530, Jun. 2024.

[58] R. Nikandish, C. Sheedy, J. He, R. Crowe, and D. Rao, "Contactless glucose sensing using miniature mm-wave radar and tiny machine learning," *IEEE J. Microw.*, vol. 5, no. 2, pp. 281–290, Mar. 2025.

[59] N. F. Ali, A. Aldhaheri, B. Wodajo, M. Alshamsi, S. Alshamsi, and M. Atef, "Non-invasive continuous real-time blood glucose estimation using PPG features-based convolutional autoencoder with TinyML implementation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2024, pp. 1–5.

[60] M. Dima Genemo, "Federated learning for bronchus cancer detection using tiny machine learning edge devices," *Indonesian J. Data Sci.*, vol. 5, no. 1, pp. 64–69, Mar. 2024.

[61] C. Yuan, D. Zhao, and S. S. Agaian, "MUCM-Net: A mamba powered UCM-Net for skin lesion segmentation," 2024, *arXiv:2405.15925*.

[62] D.-M. Armanda, A.-G. Cococi, I. Dogaru, and R. Dogaru, "Skin cancer detection using constrained neural networks and the development of an optimized technical infrastructure for resource-efficient diagnosis," in *Proc. E-Health Bioeng. Conf. (EHB)*, Nov. 2024, pp. 1–4.

[63] M. Z. M. Shamim, "Hardware deployable edge-AI solution for prescreening of oral tongue lesions using TinyML on embedded devices," *IEEE Embedded Syst. Lett.*, vol. 14, no. 4, pp. 183–186, Dec. 2022.

[64] A. Rostami, B. Tarvirdizadeh, K. Alipour, and M. Ghamari, "Real-time stress detection from raw noisy PPG signals using LSTM model leveraging TinyML," *Arabian J. Sci. Eng.*, vol. 50, no. 10, pp. 1–23, May 2025.

[65] M. Gibbs, K. Woodward, and E. Kanjo, "Combining multiple tiny machine learning models for multimodal context-aware stress recognition on constrained microcontrollers," *IEEE Micro*, vol. 44, no. 3, pp. 67–75, May 2024.

[66] A. Messina, F. M. Bregantin, D. Mallamaci, A. Vita, and M. Merenda, "Preliminary analysis on the use of tiny machine learning for the exploitation of biosensors in emotional state estimation," in *Proc. Int. Workshop Quantum Biomed. Appl., Technol., Sensors (Q-BATS)*, Oct. 2024, pp. 99–104.

[67] A. Abu-Samah, D. Ghaffa, N. F. Abdullah, N. Kamal, R. Nordin, J. C. D. Cruz, G. V. Magwili, and R. J. Mercado, "Deployment of TinyML-based stress classification using computational constrained health wearable," *Electronics*, vol. 14, no. 4, p. 687, Feb. 2025.

[68] I. Lamaakal, C. Yahyati, K. El Makkaoui, I. Ouahbi, and Y. Maleh, "SNAP-UQ: Self-supervised next-activation prediction for single-pass uncertainty in TinyML," 2025, *arXiv:2508.12907*.

[69] M. Deji Dere, R. O. Dere, A. Adesina, and A. R. Yauri, "SmartCall: A real-time, sign language medical emergency communicator," in *Proc. 5th Inf. Technol. Educ. Develop. (ITED)*, Nov. 2022, pp. 1–6.

[70] S. Sharma, R. Gupta, and A. Kumar, "A TinyML solution for an IoT-based communication device for hearing impaired," *Expert Syst. Appl.*, vol. 246, Jul. 2024, Art. no. 123147.

[71] C. Alin-Gabriel and D. Radu, "Ocular disease recognition using TinyML for efficient Android implementation," in *Proc. Int. Conf. E-Health Bioeng.*, Nov. 2024, pp. 702–710.

[72] S. S. Saha, M. A. Puthenveedu, and M. Chowdhary, "On-device, continuous, cuffless, and accelerometer-based blood pressure monitoring," in *Proc. 47th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2025, pp. 1–10.

[73] L. Cao, Y. W. Lim, M. P. Tan, and F. Z. Rokhani, "Cognitive frailty classification models for older adults in a point-of-care system," in *Proc. IEEE Asia–Pacific Conf. Circuits Syst. (APCCAS)*, Nov. 2024, pp. 631–635.

[74] A. Krishna, S. Debnath, M. Srivatsav, A. van Schaik, M. Mehendale, and C. Singh Thakur, "Neural signal compression using Raman TinyML accelerator for BCI applications," 2025, *arXiv:2504.06996*.

[75] S. Ghosh, Y. Gholap, and S. Tallur, "Wearable sensing module for table tennis stroke detection," in *Proc. IEEE Appl. Sens. Conf. (APSCON)*, Jan. 2023, pp. 1–3.

[76] M. S. Diab and E. Rodriguez-Villegas, "Assessing the feasibility of cough detection using statistical features extracted from accelerometry data," in *Proc. Int. Conf. E-Health Bioeng.*, Nov. 2024, pp. 485–493.

[77] W. L. Pang, G. C. Chung, K. Y. Chan, L. I. Ee, M. Roslee, E. Fitrey, Y. W. Sim, and M. D. Prasetio, "Smart machine learning-based IoT health and cough monitoring system," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 13, no. 5, pp. 1645–1653, Oct. 2023.

[78] A. Abdel-Ghani, Z. Abughazzah, M. Akhund, A. Abdalla, K. Abualsaud, and E. Yaacoub, "Joint use of vital signs and cough sounds for pandemic detection," in *Proc. Int. Telecommun. Conf. (ITC-Egypt)*, Jul. 2024, pp. 20–25.

[79] M. S. Diab and E. Rodriguez-Villegas, "Comparison of the performance of statistical and spectral feature based models for embedded cough detection using accelerometry data," in *Proc. Int. Conf. E-Health Bioeng.*, Nov. 2024, pp. 476–484.

[80] C.-E.-A. Tai, E. Janes, C. Czarnecki, and A. Wong, "Double-condensing attention condenser: Leveraging attention in deep learning to detect skin cancer from skin lesion images," *Sensors*, vol. 24, no. 22, p. 7231, Nov. 2024.

[81] Vincent, G. Darian, and N. Surantha, "Performance evaluation of convolutional neural network (CNN) for skin cancer detection on edge computing devices," *Appl. Sci.*, vol. 15, no. 6, p. 3077, Mar. 2025.

[82] C. Yuan, D. Zhao, and S. S. Agaian, "UCM-NetV2 and BNN-UCM-NetV2: Efficient and accurate deep learning models for skin lesion segmentation on mobile devices," Tech. Rep., 2024.

[83] C. Yuan, D. Zhao, and S. S. Agaian, "UCM-Net: A lightweight and efficient solution for skin lesion segmentation using MLP and CNN," *Biomed. Signal Process. Control*, vol. 96, Oct. 2024, Art. no. 106573.

[84] A. Gupta, K. Gautam, Y. P. Singh, B. Singh, M. Wajid, and M. Usman, "Hardware deployable classifier for invasive ductal carcinoma using tiny machine learning model," in *Proc. 16th Int. Conf. Sens. Technol. (ICST)*, Dec. 2023, pp. 1–6.

[85] J. Lee, J. Park, and Y. Lee, "Towards efficient cancer detection on mobile devices," *IEEE Access*, vol. 13, pp. 34613–34626, 2025.

[86] S. R. Chokhandre, P. Jadhav, N. Gole, A. N. Thakare, A. R. Bondade, and P. Singh, "Clinical breast imaging and deep learning model for image malignancy prediction using resource-constrained embedded devices," in *Proc. 4th Int. Conf. Artif. Intell. Signal Process. (AISP)*, Oct. 2024, pp. 1–5.

[87] B. G. Simões, V. B. D. S. R. Dos, A. D. C. J. Santos, and W. Augusto, "Specialized brain tumor detection system for superior view of cranial MRI using AI and TinyML techniques," in *Proc. Anais Do Workshop De Micro-Ondas*, Oct. 2023, pp. 1–4.

[88] P. Srivastava, N. J. Shah, and K. Jaiswal, "Microcontroller-based EdgeML: Health monitoring for stress and sleep via HRV," *Eng. Proc.*, vol. 78, no. 1, p. 3, 2024.

[89] J. Xie, Q. Wu, N. Dey, F. Shi, R. S. Sherratt, and Y. Kuang, "Empowering stroke recovery with upper limb rehabilitation monitoring using TinyML based heterogeneous classifiers," *Sci. Rep.*, vol. 15, no. 1, pp. 1–18, May 2025.

[90] I. Lamaakal, Z. Charroud, Y. Maleh, I. Ouahbi, and K. E. Makkaoui, "Optimizing breast calcification detection in mammography using PySpark: A big data and machine learning approach," in *Proc. Int. Conf. Data Anal. Manage.*, Jun. 2025, pp. 617–627.

[91] B. Riyanta, H. A. Irianta, and B. P. Kamiel, "Development of speech command control based TinyML system for post-stroke dysarthria therapy device," *J. Robot. Control*, vol. 4, no. 4, pp. 466–478, Jul. 2023.

[92] Sanjana, Karunya, K. Ushashree, V. J. Shetty, and G. Hiremath, "Safe steps: A smart security system for elderly," in *Proc. Int. Conf. Comput., Semiconductor, Mechatronics, Intell. Syst. Commun. (COSMIC)*, Nov. 2024, pp. 212–219.

[93] Z. Abushaban, S. E. Yüzbaşıoğlu, S. Dilek, and S. Tosun, "Optimizing deep learning models for ophthalmic disease detection on resource-constrained devices," in *Proc. 7th Int. Congr. Human–Comput. Interact., Optim. Robotic Appl. (ICHORA)*, May 2025, pp. 1–9.

[94] N. Schärer, F. Villani, A. Melatur, S. Peter, T. Polonelli, and M. Magno, "ElectraSight: Smart glasses with fully onboard non-invasive eye tracking using hybrid contact and contactless EOG," 2024, *arXiv:2412.14848*.

[95] Puneet, R. Kumar, and M. Gupta, "Optical coherence tomography image based eye disease detection using deep convolutional neural network," *Health Inf. Sci. Syst.*, vol. 10, no. 1, p. 13, Jun. 2022.

[96] M. Kumar, H. P. Mohan Kumar, H. Sultana, and J. S. Shweta, "Mobile-based classification and detection of diabetic retinopathy using TinyML," in *Proc. IEEE North Karnataka Subsection Flagship Int. Conf. (NKCon)*, Nov. 2023, pp. 1–6.

[97] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "TVM: An automated end-to-end optimizing compiler for deep learning," in *Proc. 13th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, 2018, pp. 578–594.

[98] R. David, J. Duke, A. Jain, J. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, T. Wang, P. Warden, and R. Rhodes, "TensorFlow lite micro: Embedded machine learning for tinyml systems," in *Proc. Mach. Learn. Syst.*, vol. 3, 2021, pp. 800–811.

[99] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for arm cortex-M CPUs," 2018, *arXiv:1801.06601*.

[100] *Edge Impulse—The Leading Edge AI Platform*. Accessed: May, 30, 2025. [Online]. Available: https://www.edgeimpulse.com/

[101] N. Tan. *Utensor/Utensor: TinyML AI Inference Library*. Accessed: May, 30, 2025. [Online]. Available: https://github.com/uTensor/uTensor

[102] *X-CUBE-AI AI Expansion Pack for STM32CubeMX Stmicroelectronics*. Accessed: May, 26, 2025. [Online]. Available: https://www.st.com/en/embedded-software/x-cube-ai.html

[103] Cartesiam. *Nanoedge AI Library*. Accessed: May, 29, 2025. [Online]. Available: https://cartesiam.ai/

[104] *Eloquentarduino*. Accessed: May, 28, 2025. [Online]. Available: https://github.com/eloquentarduino

[105] L. Tsutsui da Silva, V. M. A. Souza, and G. E. A. P. A. Batista, "EmbML tool: Supporting the use of supervised learning algorithms in low-cost embedded systems," in *Proc. IEEE 31st Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2019, pp. 1633–1637.

[106] D. Morawiec. *Sklearn-Porter*. Accessed: May, 28, 2025. [Online]. Available: https://github.com/nok/sklearn-porter

[107] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 199–213.

[108] *Arduino Nano 33 BLE Sense*. Accessed: May, 10, 2025. [Online]. Available: https://store-usa.arduino.cc/products/arduino-nano-33-ble-sense

[109] May, 10, 2025. (2025). *TinyML-Language Detector-Based on Edge Impulse and Arduino*. [Online]. Available: https://create.arduino.cc/projecthub/enzo2/tinyml-language-detector-based-on-edge-impulse-arduino-f5cfa8?ref=part&ref_id=107215&ofset=0

[110] I. Lamaakal, I. Ouahbi, K. El Makkaoui, Y. Maleh, P. Pławiak, and F. Alblehai, "A TinyDL model for gesture-based air handwriting Arabic numbers and simple Arabic letters recognition," *IEEE Access*, vol. 12, pp. 76589–76605, 2024.

[111] (2024). *TinyML: Speech Commands Detection*. [Online]. Available: https://create.arduino.cc/projecthub/ugotan/tinyml-speech-commands-detection-a3b51b?ref=part&ref_id=107215&ofset=1

[112] *Jetson Nano Developer Kit*. Accessed: May 10, 2025. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[113] (2025). *Adafruit EdgeBadge—TensorFlow Lite for MCUs*. [Online]. Available: https://www.adafruit.com/product/4400

[114] (2025). *Google Coral Dev Board*. [Online]. Available: https://coral.ai/products/dev-board/

[115] (2025). *Seeed Studio Wio Terminal*. [Online]. Available: https://wiki.seeedstudio.com/Wio-Terminal-Getting-Started/

[116] (2025). *Adafruit CLUE*. [Online]. Available: https://learn.adafruit.com/adafruit-clue/

[117] (2025). *PYNQ-Z2*. [Online]. Available: https://www.amd.com/en/corporate/university-program/aup-boards/pynq-z2.html

[118] (2025). *BeagleBone Black*. [Online]. Available: https://www.beagleboard.org/boards/beaglebone-black

[119] (2025). *Syntiant Tiny Machine Learning Development Board*. [Online]. Available: https://www.syntiant.com/tinyml

[120] (2025). *ESP32-DevKitC*. [Online]. Available: https://www.espressif.com/en/products/devkits/esp32-devkitc/overview

[121] (2025). *Himax WE-I*. [Online]. Available: https://www.himax.com.tw/products/intelligent-sensing/always-on-smart-sensing/

[122] (2025). *Spresense*. [Online]. Available: https://developer.sony.com/develop/spresense/

[123] (2025). *Raspberry PI 4*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[124] (2025). *Arducam Pico4ML TinyML Dev Kit*. [Online]. Available: https://www.arducam.com/docs/pico/arducam-pico4mltinymldevkit/

[125] (2025). *SparkFun Edge Development Board—Apollo3 Blue*. [Online]. Available: https://www.sparkfun.com/products/15170

[126] (2025). *Portenta H7*. [Online]. Available: https://www.arduino.cc/pro/hardware/product/portenta-h7

[127] (2025). *ESP-EYE*. [Online]. Available: https://www.gotronic.fr/art-module-esp-eye-32629.htm

[128] (2025). *Raspberry Pi Pico*. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-pico/

[129] M. Silva, G. Signoretti, T. Flores, P. Andrade, J. Silva, I. Silva, E. Sisinni, and P. Ferrari, "A data-stream TinyML compression algorithm for vehicular applications: A case study," in *Proc. IEEE Int. Workshop Metrology Ind. 4.0 IoT (MetroInd4.0&IoT)*, Jun. 2022, pp. 408–413.

[130] G. K. Sahoo, S. K. Das, and P. Singh, "A deep learning-based distracted driving detection solution implemented on embedded system," *Multimedia Tools Appl.*, vol. 82, no. 8, pp. 11697–11720, Mar. 2023.

[131] A. Gkogkidis, V. Tsoukas, and A. Kakarountas, "A TinyML-based alcohol impairment detection system for vehicle accident prevention," in *Proc. 7th South-East Eur. Design Autom., Comput. Eng., Comput. Netw. Social Media Conf. (SEEDA-CECNSM)*, Preveza, Greece, Sep. 2022, pp. 1–6.

[132] T. Flores, M. Silva, M. Azevedo, T. Medeiros, M. Medeiros, I. Silva, M. M. D. Santos, and D. G. Costa, "TinyML for safe driving: The use of embedded machine learning for detecting driver distraction," in *Proc. IEEE Int. Workshop Metrology Automot. (MetroAutomotive)*, Naples, Italy, Jun. 2023, pp. 62–66.

[133] K.-F. Lee, X.-Z. Chen, C.-W. Yu, K.-Y. Chin, Y.-C. Wang, C.-Y. Hsiao, and Y.-L. Chen, "An intelligent driving assistance system based on lightweight deep learning models," *IEEE Access*, vol. 10, pp. 111888–111900, 2022.

[134] L. Sousa, R. Silva, H. Peixoto, P. Melo-Pinto, A. Costa, C. Melo, P. Delgado, V. Fukuda, and J. Machado, "Enhancing automotive products with TinyML and MEMS sensors: A preliminary approach," in *Proc. Int. Conf. Data Sci. Artif. Intell.*, Nov. 2024, pp. 194–208.

[135] T. Flores, M. Silva, P. Andrade, J. Silva, I. Silva, E. Sisinni, P. Ferrari, and S. Rinaldi, "A TinyML soft-sensor for the Internet of Intelligent Vehicles," in *Proc. IEEE Int. Workshop Metrology Automot. (MetroAutomotive)*, Jul. 2022, pp. 18–23.

[136] N. N. Alajlan and D. M. Ibrahim, "DDD TinyML: A TinyML-based driver drowsiness detection model using deep learning," *Sensors*, vol. 23, no. 12, p. 5696, Jun. 2023.

[137] H.-T. Nguyen, N.-D. Mai, B. G. Lee, and W.-Y. Chung, "Behind-the-ear EEG-based wearable driver drowsiness detection system using embedded tiny neural networks," *IEEE Sensors J.*, vol. 23, no. 19, pp. 23875–23892, Oct. 2023.

[138] C. Lin, X. Zhu, R. Wang, W. Zhou, N. Li, and Y. Xie, "Early driver fatigue detection system: A cost-effective and wearable approach utilizing embedded machine learning," *Vehicles*, vol. 7, no. 1, p. 3, Jan. 2025.

[139] M. Medeiros, T. Flores, M. Silva, and I. Silva, "A multi-layered methodology for driver behavior analysis using TinyML and edge computing," in *Proc. IEEE Int. Conf. Evolving Adapt. Intell. Syst. (EAIS)*, May 2024, pp. 1–8.

[140] M. Silva, T. Medeiros, M. Azevedo, M. Medeiros, M. Themoteo, T. Gois, I. Silva, and D. G. Costa, "An adaptive TinyML unsupervised online learning algorithm for driver behavior analysis," in *Proc. IEEE Int. Workshop Metrology Automot. (MetroAutomotive)*, Jun. 2023, pp. 199–204.

[141] S. Ullah and D.-H. Kim, "Lightweight driver behavior identification model with sparse learning on in-vehicle CAN-BUS sensor data," *Sensors*, vol. 20, no. 18, p. 5030, Sep. 2020.

[142] H. A. Khalil, S. A. Hammad, H. E. Abd El Munim, and S. A. Maged, "Low-cost driver monitoring system using deep learning," *IEEE Access*, vol. 13, pp. 14151–14164, 2025.

[143] G. Chopra, S. Rani, W. Viriyasitavat, G. Dhiman, A. Kaur, and S. Vimal, "UAV-assisted partial co-operative NOMA-based resource allocation in CV2X and TinyML-based use case scenario," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 21402–21410, Jun. 2024.

[144] R. Liu, M. Xie, A. Liu, and H. Song, "Joint optimization risk factor and energy consumption in IoT networks with TinyML-enabled Internet of UAVs," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 20983–20994, Jun. 2024.

[145] T. Zheng, Y. Qiu, Y. Zheng, Q. Wang, and X. Chen, "Enhancing TinyML-based container escape detectors with systemcall semantic association in UAVs networks," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 21158–21169, Jun. 2024.

[146] A. Kiran, J. V. N. Ramesh, A. Quraishi, J. C. Patni, I. Keshta, H. Byeon, M. Raparthi, M. Sandhu, and M. Soni, "Tiny machine learning approach for grid-based monitoring of UAV tracking and cyber-physical systems in hydraulic surveying," *IEEE Trans. Intell. Transp. Syst.*, vol. 26, no. 9, pp. 14029–14038, Sep. 2025.

[147] A. Shah, V. Vivek, D. Savaliya, R. Gupta, S. Tanwar, and J. Bhatia, "Tiny ML-based secure and energy efficient unmanned aerial vehicles surveillance framework for smart cities," in *Proc. 3rd Int. Conf. Intell. Syst., Adv. Comput. Commun. (ISACC)*, Feb. 2025, pp. 98–103.

[148] W. Raza, A. Osman, F. Ferrini, and F. D. Natale, "Energy-efficient inference on the edge exploiting TinyML capabilities for UAVs," *Drones*, vol. 5, no. 4, p. 127, Oct. 2021.

[149] A. N. Roshan, B. Gokulapriyan, C. Siddarth, and P. Kokil, "Adaptive traffic control with TinyML," in *Proc. 6th Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET)*, Chennai, India, Mar. 2021, pp. 451–455.

[150] P. Andrade, I. Silva, G. Signoretti, M. Silva, J. Dias, L. Marques, and D. G. Costa, "An unsupervised TinyML approach applied for pavement anomalies detection under the Internet of Intelligent Vehicles," in *Proc. IEEE Int. Workshop Metrology Ind. 4.0 IoT (MetroInd4.0&IoT)*, Rome, Italy, Jun. 2021, pp. 642–647.

[151] I. Lamaakal, Y. Maleh, I. Ouahbi, K. E. Makkaoui, and A. A. A. El-Latif, "A deep learning-powered TinyML model for gesture-based air handwriting simple Arabic letters recognition," in *Proc. Int. Conf. Digit. Technol. Appl.*, May 2024, pp. 32–42.

[152] W. Wang, A. Ullah, L. Li, and M. Wang, "Optimized binary neural networks for road anomaly detection: A TinyML approach on edge devices," *Comput., Mater. Continua*, vol. 80, no. 1, pp. 527–546, 2024.

[153] S. Zhou, Y. Du, B. Chen, Y. Li, and X. Luan, "An intelligent IoT sensing system for rail vehicle running states based on TinyML," *IEEE Access*, vol. 10, pp. 98860–98871, 2022.

[154] T. Flores, M. Andrade, M. Medeiros, M. Amaral, M. Silva, and I. Silva, "Leveraging IoT and TinyML for smart battery management in electric bicycles," in *Proc. Symp. Internet Things (SIoT)*, Oct. 2023, pp. 1–5.

[155] R. Sanchez-Iborra, L. Bernal-Escobedo, J. Santa, and A. Skarmeta, "TinyML-based fall detection for connected personal mobility vehicles," *Comput., Mater. Continua*, vol. 71, no. 2, pp. 3869–3885, 2022.

[156] M. de Prado, M. Rusci, A. Capotondi, R. Donze, L. Benini, and N. Pazos, "Robustifying the deployment of TinyML models for autonomous mini-vehicles," *Sensors*, vol. 21, no. 4, p. 1339, Feb. 2021.

[157] M. de Prado, M. Rusci, R. Donze, A. Capotondi, S. Monnerat, L. Benini, and A. N. Pazos, "Robust navigation with TinyML for autonomous mini-vehicles," 2020, *arXiv:2007.00302*.

[158] U. Ahmad, M. Han, and S. Mahmood, "Enhancing security in connected and autonomous vehicles: A pairing approach and machine learning integration," *Appl. Sci.*, vol. 14, no. 13, p. 5648, Jun. 2024.

[159] A. Albanese, D. Gotta, and D. Brunelli, "A TinyML-based IoT device for advanced shipping monitoring," in *Proc. Int. Conf. Appl. Electron. Pervading Ind.*, Sep. 2025, pp. 131–138.

[160] I. Lamaakal, C. Yahyati, Y. Maleh, K. E. Makkaoui, I. Ouahbi, and D. Niyato, "An explainable tiny-fast Kolmogorov–Arnold network for gesture-based air handwriting recognition of tifinagh letters in resource-constrained IoT device," *IEEE Internet Things J.*, early access, Oct. 23, 2025, doi: 10.1109/JIOT.2025.3625087.

[161] I. Lamaakal, C. Yahyati, Z. Charroud, K. E. Makkaoui, I. Ouahbi, Y. Maleh, S. A. Chelloug, A. A. A. El-Latif, H. S. Khalifa, and D. Niyato, "Tiny deep learning models with hybrid compression techniques for gesture-based air handwriting recognition of English alphabets on edge device," *IEEE Internet Things J.*, early access, Oct. 22, 2025, doi: 10.1109/JIOT.2025.3624283.

[162] S. Mallick, P. Ruparel, S. Gawali, and N. Goveas, "Resource-constrained device characterization for detecting sleep apnea using machine learning," in *Proc. Int. Conf. Smart Appl., Commun. Netw. (SmartNets)*, Jul. 2023, pp. 1–7.

[163] O. Hassan, T. Paul, N. Amin, T. Titirsha, R. Thakker, D. Parvin, A. S. M. Mosa, and S. K. Islam, "An optimized hardware inference of SABiNN: Shift-accumulate binarized neural network for sleep apnea detection," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, 2023.

[164] S. Mallick, S. Gawali, C. Onime, and N. Goveas, "Sleep apnea detection system using machine learning on resource-constrained devices," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, Apr. 2023, pp. 1–6.

[165] I. Lamaakal, C. Yahyati, I. Ouahbi, K. E. Makkaoui, and Y. Maleh, "A survey of model compression techniques for TinyML applications," in *Proc. Int. Conf. Circuit, Syst. Commun. (ICCSC)*, Jun. 2025, pp. 1–6, doi: 10.1109/ICCSC66714.2025.11135279.

[166] C. Yahyati, I. Lamaakal, K. E. Makkaoui, I. Ouahbi, and Y. Maleh, "TinyML-based facial recognition for embedded systems," in *Proc. Int. Conf. Circuit, Syst. Commun. (ICCSC)*, Jun. 2025, pp. 1–6, doi: 10.1109/ICCSC66714.2025.11134957.

[167] S. Essahraui, I. Lamaakal, Y. Maleh, K. El Makkaoui, M. F. Bouami, I. Ouahbi, A. A. Abd El-Latif, M. Almousa, and J. J. P. C. Rodrigues, "Human behavior analysis: A comprehensive survey on techniques, applications, challenges, and future directions," *IEEE Access*, vol. 13, pp. 128379–128419, 2025, doi: 10.1109/ACCESS.2025.3589938.

[168] C. Rami, I. Lamaakal, I. Ouahbi, K. El Makkaoui, and Y. Maleh, "Quantized YOLOv11 for real-time road object detection in smart city environments," in *Proc. Int. Conf. Circuit, Syst. Commun. (ICCSC)*, Jun. 2025, pp. 1–6, doi: 10.1109/ICCSC66714.2025.11135434.

[169] A. Sahoo, S. Sharma, and S. Pattnaik, "Smart physiotherapy: Monitoring distal carpal kinematics using TinyML," in *Proc. IEEE 4th Int. Conf. Appl. Electromagn., Signal Process., Commun. (AESPC)*, Nov. 2024, pp. 1–6.

[170] H. Wu and K. Nazarpour, "A TinyML-based system for prosthetic control," in *Proc. IEEE 11th Int. Conf. Inf., Commun. Netw. (ICICN)*, Aug. 2023, pp. 549–553.

[171] S. Shakya, A. Taparugssanagorn, and C. Silpasuwanchai, "Convolutional neural network-based low-powered wearable smart device for gait abnormality detection," *IoT*, vol. 4, no. 2, pp. 57–77, Mar. 2023.

[172] M. O. Seddar, G. Rao, A. Fleury, and M. Kahn, "Detecting the pre-impact of falls in the elderly, along with the use of an airbag belt for protection against femoral neck fractures," in *Proc. Int. Conf. Smart Homes Health Telematics*, Jul. 2023, pp. 117–129.

[173] D. J. Mala, T. V. Padmavathy, A. P. Reynold, and M. Meena, "Edge analytics with TinyML technique for MIoT applications for personalized healthcare in smart nation," in *Proc. 2nd Int. Conf. Smart Technol. Smart Nation (SmartTechCon)*, Aug. 2023, pp. 39–54.

[174] A. Manoj, A. Paramasivam, S. Vijayalakshmi, B. K. B., and T. S. K., "An edge computing enabled Internet of Medical Things device for remote patient health monitoring applications," in *Proc. IEEE Students Conf. Eng. Syst. (SCES)*, Jun. 2024, pp. 1–6.

[175] R. Arthi and S. Krishnaveni, "Optimized tiny machine learning and explainable AI for trustable and energy-efficient fog-enabled healthcare decision support system," *Int. J. Comput. Intell. Syst.*, vol. 17, no. 1, p. 229, Sep. 2024.

[176] C. C. Vu, T. N. Nguyen, and M. H. Nguyen, "Thin and flexible pressure sensors for classifying body signals enhanced by tiny machine learning algorithms," *Measurement*, vol. 253, Sep. 2025, Art. no. 117799.

[177] I. Lamaakal, C. Yahyati, Y. Maleh, K. El Makkaoui, and I. Ouahbi, "BayesQ: Uncertainty-guided Bayesian quantization," 2025, *arXiv:2511.08821*.

[178] S. Essahraui, I. Lamaakal, Y. Maleh, K. El Makkaoui, M. F. Bouami, I. Ouahbi, H. Elmannai, and A. A. Abd El-Latif, "FastKAN-DDD: A novel fast Kolmogorov–Arnold network-based approach for driver drowsiness detection optimized for TinyML deployment," *PLoS ONE*, vol. 20, no. 11, Nov. 2025, Art. no. e0332577, doi: 10.1371/journal.pone.0332577.

[179] S. Essahraui, I. Lamaakal, Y. Maleh, K. E. Makkaoui, M. F. Bouami, I. Ouahbi, M. Almousa, A. A. S. AlQahtani, and A. A. A. El-Latif, "Deep learning models for detecting cheating in online exams," *Comput., Mater. Continua*, vol. 85, no. 2, pp. 3151–3183, 2025, doi: 10.32604/cmc.2025.067359.

[180] M. Hizem, L. Bousbia, Y. Ben Dhiab, M. O.-E. Aoueileyine, and R. Bouallegue, "Reliable ECG anomaly detection on edge devices for Internet of Medical Things applications," *Sensors*, vol. 25, no. 8, p. 2496, Apr. 2025.

[181] A. C. Johnvictor, M. Poonkodi, N. Prem Sankar, and V. Thinesh, "TinyML-based lightweight AI healthcare mobile chatbot deployment," *J. Multidisciplinary Healthcare*, vol. 17, pp. 5091–5104, Nov. 2024.

[182] P. Fusco, G. P. Rimoli, and M. Ficco, "TinyML and federated learning for resource-constrained medical devices," in *Artificial Intelligence Techniques for Analysing Sensitive Data in Medical Cyber-Physical Systems: System Protection and Data Analysis*. Cham, Switzerland: Springer, 2025, pp. 113–126.

[183] Y. Wang and L. Sun, "Energy-efficient dynamic sensor time series classification for edge health devices," *Comput. Methods Programs Biomed.*, vol. 254, Sep. 2024, Art. no. 108268.

[184] X. Zhou, Z. Kang, R. Canady, S. Bao, D. A. Balasubramanian, and A. Gokhale, "Exploring cloud assisted tiny machine learning application patterns for PHM scenarios," in *Proc. Annu. Conf. PHM Soc.*, Nov. 2021, pp. 1–10.

[185] J. I. de Oliveira Filho, M. C. Faleiros, D. C. Ferreira, V. Mani, and K. N. Salama, "Empowering electrochemical biosensors with AI: Overcoming interference for precise dopamine detection in complex samples," *Adv. Intell. Syst.*, vol. 5, no. 10, Oct. 2023, Art. no. 2300227.

[186] R. M. Umutoni, M. M. Ogore, R. L. Savanna, D. Hanyurwimfura, J. Nsenga, D. Mukanyirigira, F. Nzanywayingoma, D. Ngabo, and J. Habiyaremye, "Integration of TinyML-based proximity and couch sensing in wearable devices for monitoring infectious disease's social distance compliance," in *Proc. 12th Int. Conf. Softw. Comput. Appl.*, Feb. 2023, pp. 349–355.

[187] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4794–4802.

[188] Z. Yao, Z. Dong, Z. Zheng, A. Gholaminejad, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. W. Mahoney, and E. K. Keutzer, "HAWQ-v3: Dyadic neural network quantization," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 11875–11886.

[189] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun, "MetaPruning: Meta learning for automatic neural network channel pruning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3296–3305.

[190] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–34, May 2022.

[191] Y. Li, F. Hu, Y. Liu, M. Ryan, and R. Wang, "A hybrid model compression approach via knowledge distillation for predicting energy consumption in additive manufacturing," *Int. J. Prod. Res.*, vol. 61, no. 13, pp. 4525–4547, Jul. 2023.

[192] R. Goyal, J. Vanschoren, V. V. Acht, and S. Nijssen, "Fixed-point quantization of convolutional neural networks for quantized inference on embedded platforms," 2021, *arXiv:2102.02147*.

[193] C. Yuan and S. S. Agaian, "A comprehensive review of binary neural network," *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 12949–13013, Nov. 2023, doi: 10.1007/s10462-023-10464-w.

[194] V. Tsoukas, A. Gkogkidis, E. Boumpa, and A. Kakarountas, "A review on the emerging technology of TinyML," *ACM Comput. Surv.*, vol. 56, no. 10, pp. 1–37, Oct. 2024.

[195] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, and A. S. Hafid, "A comprehensive survey on TinyML," *IEEE Access*, vol. 11, pp. 96892–96922, 2023, doi: 10.1109/ACCESS.2023.3294111.

[196] I. Lamaakal, Y. Maleh, K. El Makkaoui, I. Ouahbi, P. Pławiak, O. Alfarraj, M. Almousa, and A. A. Abd El-Latif, "Tiny language models for automation and control: Overview, potential applications, and future research directions," *Sensors*, vol. 25, no. 5, p. 1318, Feb. 2025.

[197] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the extreme edge: A survey on reformable TinyML," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 1–30, Dec. 2023.

[198] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, and M. Cristani, "A machine learning-oriented survey on tiny machine learning," *IEEE Access*, vol. 12, pp. 23406–23426, 2024, doi: 10.1109/ACCESS.2024.3365349.

[199] D. L. Dutta and S. Bharali, "TinyML meets IoT: A comprehensive survey," *Internet Things*, vol. 16, Dec. 2021, Art. no. 100461.

[200] A. Elhanashi, P. Dini, S. Saponara, and Q. Zheng, "Advancements in TinyML: Applications, limitations, and impact on IoT devices," *Electronics*, vol. 13, no. 17, p. 3562, Sep. 2024.

[201] D. Kudithipudi, A. Daram, A. M. Zyarah, F. T. Zohora, J. B. Aimone, A. Yanguas-Gil, N. Soures, E. Neftci, M. Mattina, V. Lomonaco, C. D. Thiem, and B. Epstein, "Design principles for lifelong learning AI accelerators," *Nature Electron.*, vol. 6, no. 11, pp. 807–822, Nov. 2023.

[202] M. Vaithianathan, "Memory hierarchy optimization strategies for high-performance computing architectures," *Int. J. Emerg. Trends Technol. Comput. Sci.*, vol. 6, pp. 1–24, Jan. 2025.

[203] H. Guo, R. Fu, Y. Geng, S. Liu, S. Shi, T. Wang, C. Qiang, C. Li, Y. Li, Z. Wen, Y. Liu, and X. Liu, "Mel-refine: A plug-and-play approach to refine mel-spectrogram in audio generation," 2024, *arXiv:2412.08577*.

[204] V. K. Quy, N. V. Hau, D. V. Anh, N. M. Quy, N. T. Ban, S. Lanza, G. Randazzo, and A. Muzirafuti, "IoT-enabled smart agriculture: Architecture, applications, and challenges," *Appl. Sci.*, vol. 12, no. 7, p. 3396, Mar. 2022.

[205] R. A. Sekar, T. Prabakaran, A. Sudhakar, and R. S. Kumar, "Industrial automation using IoT," *AIP Conf. Proc.*, vol. 2393, May 2022, Art. no. 020083.

[206] A. Biswas and H.-C. Wang, "Autonomous vehicles enabled by the integration of IoT, edge intelligence, 5G, and blockchain," *Sensors*, vol. 23, no. 4, p. 1963, Feb. 2023.

[207] G. Mois, S. Folea, and T. Sanislav, "Analysis of three IoT-based wireless sensors for environmental monitoring," *IEEE Trans. Instrum. Meas.*, vol. 66, no. 8, pp. 2056–2064, Aug. 2017.

[208] B. Sudharsan, J. G. Breslin, M. Tahir, M. Intizar Ali, O. Rana, S. Dustdar, and R. Ranjan, "OTA-TinyML: Over the air deployment of TinyML models and execution on IoT devices," *IEEE Internet Comput.*, vol. 26, no. 3, pp. 69–78, May 2022.

[209] J. Lee, N. Chirkov, E. Ignasheva, Y. Pisarchyk, M. Shieh, F. Riccardi, R. Sarokin, A. Kulik, and M. Grundmann, "On-device neural net inference with mobile GPUs," 2019, *arXiv:1907.01989*.

[210] B. Nadji, "Data security, integrity, and protection," in *Data, Security, and Trust in Smart Cities*. Cham, Switzerland: Springer, 2024, pp. 59–83.

[211] S. El Jaouhari and E. Bouvet, "Secure firmware over-the-air updates for IoT: Survey, challenges, and discussions," *Internet Things*, vol. 18, May 2022, Art. no. 100508.

[212] H. Zhou, Y. Chen, W. Zeng, L. Cui, G. Wang, and X. Liu, "GPComp: Using GPU and SSD-GPU peer to peer DMA to accelerate LSM-tree compaction for key-value store," *IEEE Trans. Parallel Distrib. Syst.*, vol. 36, no. 9, pp. 1920–1936, Sep. 2025.

[213] P. Cariani and J. M. Baker, "Survey of temporal coding of sensory information," *Frontiers Comput. Neurosci.*, vol. 19, Jul. 2025, Art. no. 1571109.

[214] S. M. Meyer, P. Weidel, P. Plank, L. Campos-Macias, S. B. Shreshta, P. Stratmann, J. Timcheck, and M. Richter, "A diagonal structured state space model on loihi 2 for efficient streaming sequence processing," in *Proc. Neuro Inspired Comput. Elements (NICE)*, Mar. 2025, pp. 1–9.

[215] D. Kudithipudi et al., "Neuromorphic computing at scale," *Nature*, vol. 637, no. 8047, pp. 801–812, 2025.

[216] H. Sang, W. Xie, G. Park, and H.-J. Yoo, "An 2.31uJ/inference ultra-low power always-on event-driven AI-IoT SoC with switchable nvSRAM compute-in-memory macro," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 71, no. 5, pp. 2534–2538, May 2024.

[217] X. Long, X. Zhu, F. Guo, C. Chen, X. Zhu, F. Gu, S. Yuan, and C. Zhang, "Spike-BRGNet: Efficient and accurate event-based semantic segmentation with boundary region-guided spiking neural networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 35, no. 3, pp. 2712–2724, Mar. 2025.

[218] Y. Zhang, P. Wang, K. Cheng, J. Zhao, J. Tao, J. Hai, J. Feng, C. Deng, and X. Wang, "Building accurate and interpretable online classifiers on edge devices," *IEEE Trans. Parallel Distrib. Syst.*, vol. 36, no. 8, pp. 1779–1796, Aug. 2025.

[219] I. Lamaakal, C. Yahyati, K. El Makkaoui, I. Ouahbi, and Y. Maleh, "TCUQ: Single-pass uncertainty quantification from temporal consistency with streaming conformal calibration for TinyML," 2025, *arXiv:2508.12905*.

[220] M. Ficco, A. Guerriero, E. Milite, F. Palmieri, R. Pietrantuono, and S. Russo, "Federated learning for IoT devices: Enhancing TinyML with on-board training," *Inf. Fusion*, vol. 104, Apr. 2024, Art. no. 102189.

[221] A. Alhindi, S. Al-Ahmadi, and M. M. B. Ismail, "Advancements and challenges in privacy-preserving split learning: Experimental findings and future directions," *Int. J. Inf. Secur.*, vol. 24, no. 3, p. 125, Jun. 2025.

[222] A. Moslemi, A. Briskina, Z. Dang, and J. Li, "A survey on knowledge distillation: Recent advancements," *Mach. Learn. Appl.*, vol. 18, Dec. 2024, Art. no. 100605.

[223] D. T. Aga, R. Chintanippu, R. A. Mowri, and M. Siddula, "Exploring secure and private data aggregation techniques for the Internet of Things: A comprehensive review," *Discover Internet Things*, vol. 4, no. 1, p. 28, Nov. 2024.

[224] R. Mathews and D. V. Jose, "Hybrid homomorphic-asymmetric lightweight cryptosystem for securing smart devices: A review," *Trans. Emerg. Telecommun. Technol.*, vol. 35, no. 1, p. 4866, Jan. 2024.

[225] H. E. Merzdorf, D. Jaison, M. B. Weaver, J. Linsey, T. Hammond, and K. A. Douglas, "Sketching assessment in engineering education: A systematic literature review," *J. Eng. Educ.*, vol. 113, no. 4, pp. 872–893, Oct. 2024.

[226] B. Chen, A. Bakhshi, G. Batista, B. Ng, and T.-J. Chin, "Update compression for deep neural networks on the edge," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 3075–3085.

[227] D. Minh, H. X. Wang, Y. F. Li, and T. N. Nguyen, "Explainable artificial intelligence: A comprehensive review," *Artif. Intell. Rev.*, vol. 55, no. 5, pp. 3503–3568, Jun. 2022.

[228] E. S. Ortigossa, T. Gonçalves, and L. G. Nonato, "Explainable artificial intelligence (XAI) From theory to methods and applications," *IEEE Access*, vol. 12, pp. 80799–80846, 2024.

[229] H. Li, J. Song, M. Xue, H. Zhang, and M. Song, "A survey of neural trees: Co-evolving neural networks and decision trees," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 7, pp. 11718–11737, Jul. 2025.

[230] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3153–3160, Apr. 2020.

[231] M. Aldughaim, K. Alshmrany, R. Menezes, L. Cordeiro, and A. Stancu, "Incremental symbolic bounded model checking of software using interval methods via contractors," 2020, *arXiv:2012.11245*.

[232] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proc.*, vol. 3, no. 1, pp. 91–99, Jun. 2022.

[233] E. Johannes Husom, A. Goknil, M. Astekin, L. K. Shar, A. Kåsen, S. Sen, B. Andreas Mithassel, and A. Soylu, "Sustainable LLM inference for edge AI: Evaluating quantized LLMs for energy efficiency, output accuracy, and inference latency," 2025, *arXiv:2504.03360*.

[234] C. J. Tan, J. Mohamad–Saleh, K. A. M. Zain, and Z. A. A. Aziz, "Review on firmware," in *Proc. Int. Conf. Imag.*, 2017, pp. 186–190.

[235] J. A. Malek, S. B. Lim, and T. Yigitcanlar, "Social inclusion indicators for building citizen-centric smart cities: A systematic literature review," *Sustainability*, vol. 13, no. 1, p. 376, Jan. 2021.

[236] C. Banbury, C. Zhou, I. Fedorov, R. Navarro, U. Thakker, D. Gope, V. J. Reddi, M. Mattina, and P. N. Whatmough, "MicroNets: Neural network architectures for deploying TinyML applications on commodity microcontrollers," in *Proc. Mach. Learn. Syst.*, vol. 3, 2021, pp. 517–532.

[237] C. Banbury et al., "MLPerf tiny benchmark," 2021, *arXiv:2106.07597*.

[238] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCUNet: Tiny deep learning on IoT devices," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020.

[239] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018, *arXiv:1806.08342*.

[240] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "NetAdapt: Platform-aware neural network adaptation for mobile applications," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 289–304.

[241] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.

[242] M. Rastegari, V. Ordóñez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 525–542.

[243] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "TensorFlow lite micro: Embedded machine learning on TinyML systems," in *Proc. Mach. Learn. Syst. (MLSys)*, vol. 3, 2020, pp. 800–811.

[244] Z. Song, Q. Liu, Q. Yang, and H. Li, "Knowledge distillation for in-memory keyword spotting model," in *Proc. Interspeech*, Sep. 2022, pp. 4128–4132.

[245] A. Chowdhery, P. Warden, J. Shlens, A. Howard, and R. Rhodes, "Visual wake words dataset," 2019, *arXiv:1906.05721*.

CHAYMAE YAHYATI received the Master of Science degree in computer science from the Multidisciplinary Faculty of Nador, Mohammed Premier University, Oujda, Morocco, where she is currently pursuing the Ph.D. degree. Her research primarily focuses on the innovative integration of TinyML and the Internet of Things.

ISMAIL LAMAAKAL (Student Member, IEEE) received the Master of Science degree in computer science from the Multidisciplinary Faculty of Nador, Mohammed Premier University, Oujda, Morocco, where he is currently pursuing the Ph.D. degree. He is as an Artificial Intelligence Scientist, his research primarily focuses on the innovative integration of tiny machine learning, the Internet of Things (IoT), and embedded systems. His work is characterized by its pioneering approach in the field, emphasizing practical applications and advancements in these interconnected domains. His contributions are marked by a commitment to pushing the boundaries of AI and its applications in the modern technological landscape.

YASSINE MALEH (Senior Member, IEEE) is currently a Professor of cybersecurity and IT governance with Sultan Moulay Slimane University, Morocco. He has made contributions in the fields of information security and privacy, the Internet of Things security, wireless, and constrained networks security. He has published over than 200 papers (book chapters, international journals, and conferences/workshops), 17 edited books, and three authored books. His research interests include information security and privacy, the Internet of Things, networks security, information systems, and IT governance. He is a member of the International Association of Engineers IAENG and the Machine Intelligence Research Laboratories. He received Publons Top 1% reviewer award, in 2018 and 2019. He has served and continues to serve on executive and technical program committees and as a Reviewer of numerous international conferences and journals, such as *Ad Hoc Networks* (Elsevier), *IEEE Network Magazine*, IEEE Sensors Journal, *ICT Express*, and *Cluster Computing* (Springer). He was the Publicity Chair of BCCA 2019 and the General Chair of the MLBDACP 19 Symposium and ICI2C'21 Conference. He is the Founding Chair of the IEEE Consultant Network Morocco and the Founding President of African Research Center of Information Technology and Cybersecurity. He was a Guest Editor of a Special Issue on Recent Advances on Cyber Security and Privacy for Cloud-of-Things of *International Journal of Digital Crime and Forensics* (IJDCF), Volume 10, Issue 3, from July 2019 to September 2019. He is the Editor-in-Chief of *International Journal of Information Security and Privacy*, and *International Journal of Smart Security Technologies* (IJSST). He serves as an Associate Editor for IEEE Access (2019 Impact Factor 4.098), *International Journal of Digital Crime and Forensics* (IJDCF), and *International Journal of Information Security and Privacy* (IJISP). He is the Series Editor of *Advances in Cybersecurity Management*, by CRC Taylor and Francis.

KHALID EL MAKKAOUI (Senior Member, IEEE) received the master's degree in networks and systems and the Ph.D. degree in computer science from Hassan I University, Settat, Morocco, in 2014 and 2018, respectively. Since 2019, he has been a Researcher and a Professor of computer science and cybersecurity with the Multidisciplinary Faculty of Nador, Mohammed Premier University, Oujda, Morocco. He has published more than 55 articles, including book chapters, international journal articles, and conference papers. His research interests include cybersecurity and artificial intelligence.

IBRAHIM OUAHBI received the Ph.D. degree in didactics of informatics from Sidi Mohamed Ben Abdellah University, Fes, Morocco, in 2018. He was a Professor of educational technologies with the Faculty of Educational Sciences, Mohammed V University, Rabat, in 2019. He is currently a Professor of computer science with the Multidisciplinary Faculty of Nador, Mohammed Premier University, Oujda, Morocco. His research interests include artificial intelligence, cybersecurity, and ICT integration in science education and learning.

MAY ALMOUSA received the Bachelor of Science degree (Hons.) in computer science from Princess Nourah bint Abdulrahman University (PNU), Riyadh, Saudi Arabia, in 2009, and the Master of Science and Ph.D. degrees in computer science from North Carolina A&T State University, in 2016 and 2022, respectively. She developed her dissertation under the supervision of Dr. Mohd Anwar at the Human-Centered AI Laboratory. In 2011, she joined PNU, as a Faculty Member with the College of Computer Science, Network and Communication Systems Department, where she taught an array of courses in computer science. She was recognized by the College of Engineering's annual graduation reception for her outstanding academic accomplishments. Her current research interests include cyber attack detection, data science, and AI techniques.

AHMED A. ABD EL-LATIF (Senior Member, IEEE) is currently a Professor of computer science with Menoufia University, Egypt. His expertise encompasses quantum cryptography, cybersecurity, chaotic dynamical systems, and AI applications in 5G and 6G networks. A full-stack computer scientist engaged in coding, development, research, and theoretical work, he has led and participated in numerous successful international research projects, securing grants across Egypt, Russia, Saudi Arabia, China, Malaysia, and Tunisia. A prolific author with over 340 publications, including more than 30 IEEE Transactions articles and 20 books, his work has garnered over 13 500 citations and an H-index of 65. He leads the MEGANET 6G Laboratory Research in Russia and has consistently been recognized among the top 2% of scientists in his field by the Stanford List of Top Scientists, in 2019 and 2024. His research interests include quantum communications and cryptography, cybersecurity, the artificial intelligence of things, AI-based image processing, information hiding, and the application of dynamical systems (chaotic systems and quantum walks) in cybersecurity. He founded the Center of Excellence in Quantum and Intelligent Computing and has received several prestigious awards, including the State Encouragement Award in Engineering Sciences, Egypt, in 2016, the Best Ph.D. Student Award, China, in 2013, and the Young Scientist Award, Menoufia University, in 2014. He actively contributes to the scientific community as Chair/Co-Chair of numerous Scopus/EI conferences and holds key editorial positions, including the Editor-in-Chief of *International Journal of Information Security and Privacy* and the Series Editor of "*Quantum Information Processing and Computing*" and "*Advances in Cybersecurity Management*". He also serves as an academic or an associate editor for many Web of Science and Scopus-indexed journals.