International Journal of Computers and Informatics

Journal Homepage: https://www.ijci.zu.edu.eg

Int. j. Comp. Info. Vol. 11 (2026) 123–141

Paper Type: Original Article

TinyML for Edge Analytics in IoT Systems: A Systematic Survey of Hardware, Software and Optimization Techniques

Mahmoud A. Mahdi [1], **Hesham F. A. Hamed** [2], **Mariam M. Hagag** [3,*] and **Amr M. Sauber** [4,5]

[1] Faculty of Computers and Informatics, Zagazig University, Zagazig, Egypt; Email: mamahdi@zu.edu.eg.

[2] Department of Artificial Intelligence, Faculty of Artificial Intelligence, Egyptian Russian University, Cairo, Egypt, Email: Heshamfathy@eru.edu.eg.

[3] Department of Data Science, Faculty of Artificial Intelligence, Egyptian Russian University, Cairo, Egypt, Email: mariam-mahmoud@eru.edu.eg.

[4] Mathematics and Computer Science Department, Faculty of Science, Menoufia University, Egypt, Email: amrmausad@science.menofia.edu.eg

[5] Computality R&D association, Egypt, Email: amrmaused@computality.com.

Received: 19 Mar 2026 **Revised**: 21 May 2026 **Accepted**: 28 Jun 2026 **Published**: 30 Jun 2026

Abstract

The proliferation of Internet of Things (IoT) devices has generated unprecedented volumes of data at the network edge, overwhelming traditional cloud-centric processing models. Tiny Machine Learning (TinyML) has emerged as a transformative paradigm enabling machine learning inference directly on resource-constrained microcontrollers with memory footprints under 1 MB and power consumption below 100 mW. This survey provides a comprehensive review of TinyML-enabled edge analytics for IoT systems, synthesizing the state of the art across hardware platforms, software frameworks, optimization techniques, application domains, and system integration. We systematically review 56 key papers from 2018 to 2026, identifying key trends, design patterns, and open challenges. Our contributions include: (1) a taxonomy of TinyML optimization techniques including quantization, pruning, knowledge distillation, and hardware-aware neural architecture search; (2) a comparative analysis of microcontroller platforms (ARM Cortex-M, ESP32, RP2040) and software frameworks (TensorFlow Lite Micro, Edge Impulse, MicroTVM); (3) a structured review of edge analytics techniques spanning inference strategies, sensor fusion, anomaly detection, time-series analysis, vision, audio, and adaptive learning; (4) an analysis of eight application domains including smart health, industrial IoT, agriculture, and smart buildings; and (5) a forward-looking discussion of open challenges including long-term reliability, distribution shift, energy harvesting, and TinyMLOps. We conclude by identifying critical research gaps and proposing a roadmap for future work toward sustainable, autonomous edge intelligence. This survey serves as a foundational reference for researchers, practitioners, and system designers developing TinyML-IoT systems.

Keywords: TinyML, Edge Analytics, IoT, Edge Intelligence, Machine Learning Inference, Model Optimization Embedded AI.

1 | Introduction

The Internet of Things (IoT) has ushered in an era of pervasive sensing, with projections estimating over 75 billion connected devices by 2030 [1]. This explosive growth has generated unprecedented volumes of data

Corresponding Author: mariam-hagag@eru.edu.eg

at the network edge — data that must be processed, analyzed, and acted upon in real time. Traditional cloud-centric processing models impose fundamental limitations: high latency, excessive bandwidth consumption, energy inefficiency, and privacy concerns. Sending every sensor reading to the cloud for analysis is simply not sustainable.

Edge analytics addresses these limitations by moving computation closer to data sources. By processing data locally, edge analytics enables real-time decision-making, reduces network congestion, preserves data privacy, and lowers operational costs. However, early edge analytics solutions relied on relatively powerful edge gateways and servers rather than true endpoint devices.

Tiny Machine Learning (TinyML) represents the logical evolution of edge analytics, enabling machine learning inference directly on microcontroller-class devices. With memory footprints as low as tens of kilobytes and power consumption under 100 milliwatts, TinyML makes it feasible to deploy neural networks on devices costing under $10. Platforms such as TensorFlow Lite Micro [2], Edge Impulse [3], and CMSIS-NN [4] have democratized embedded AI, accelerating adoption across domains including healthcare, industrial monitoring, agriculture, and smart infrastructure.

Despite rapid progress, the TinyML and IoT edge analytics landscape remains fragmented. Researchers and practitioners face a bewildering array of hardware platforms, software frameworks, optimization techniques, and application-specific considerations. This survey aims to provide a comprehensive, structured, and up-to-date synthesis of the field.

1.1 | Background and Motivation

This survey is motivated by five key drivers:

First, the exponential growth in IoT-connected devices continues to outpace the capacity of cloud-centric processing. Gartner estimates that by 2025, 75% of enterprise-generated data will be created and processed at the edge, up from 10% in 2018 [5]. Second, latency-sensitive applications such as industrial anomaly detection, autonomous vehicles, and healthcare monitoring require sub-millisecond response times that cloud round-trips cannot guarantee. Third, privacy regulations including GDPR and HIPAA [6], increasingly restrict the transmission of raw sensor data to cloud servers, pushing inference to the edge. Fourth, advances in model compression have made it feasible to run sophisticated neural networks on microcontrollers — a reality that was unimaginable just five years ago. Finally, the commercial ecosystem has matured significantly, with multiple hardware vendors, software frameworks, and cloud-based development platforms now available.

1.2 | Survey Scope and Research Questions

This survey focuses on TinyML-enabled edge analytics for IoT systems. We define TinyML as the deployment of machine learning models on microcontroller-class devices with memory under 1 MB and power under 100 mW. We exclude cloud-only ML, non-IoT embedded systems, and works that do not address resource constraints.

Our literature search was conducted in April 2026 across IEEE Xplore, ACM Digital Library, Scopus, arXiv, and Google Scholar using keywords: "TinyML," "edge analytics," "IoT inference," "embedded machine learning," "microcontroller AI," and "edge intelligence." The search covered publications from January 2018 to April 2026. Inclusion criteria were: (1) peer-reviewed or established preprints; (2) direct relevance to TinyML or edge inference; (3) IoT application context; (4) clear resource constraint characterization. Exclusion criteria were: (1) cloud-only ML; (2) non-IoT embedded systems; (3) surveys without primary contributions; (4) non-English.

From an initial pool of approximately 120 papers, 56 were selected for detailed review based on relevance, citation impact, and recency. Additionally, 25 industry white papers and technical specifications were included to capture the commercial ecosystem.

References were curated to include foundational works, recent advances, and representative applications, prioritizing high-impact and methodologically diverse sources.

1.3 | Methodology Table

Criterion	Details
Databases searched	IEEE Xplore, ACM Digital Library, Scopus, arXiv, Google Scholar
Keywords used	TinyML, edge analytics, IoT inference, embedded ML, microcontroller AI, edge intelligence
Publication period	January 2018 - April 2026
Inclusion criteria	Peer-reviewed; TinyML or edge inference focus; IoT application context; clear resource constraints
Exclusion criteria	Cloud-only ML; non-IoT embedded systems; surveys without primary contributions; non-English
Initial papers screened	120
Papers selected for detailed review	56

1.4 | Survey Contribution

The primary contributions of this survey are:

- Up-to-date coverage (2018-2026): The most recent comparable survey [7] covers only up to 2021, leaving a five-year gap in the literature.
- Unified taxonomy across optimization, hardware, software, and applications: Organized as a coherent framework as presented in Sections 2–5 of this survey rather than disconnected topical reviews.
- Quantitative synthesis of trade-offs: Model size, accuracy, memory, and power extracted from surveyed papers into comparable [as summarized in Tables 3–6] for direct cross-reference.
- Application-domain synthesis with actionable design insights: For each of six domains, we report model size ranges, accuracy, power envelopes, dominant hardware choices, and domain-specific open problems [discussed in Section 5].
- Evidence-grounded challenge analysis: Beyond listing open challenges, we quantify the scarcity of empirical validation (e.g., only 7 of 56 surveyed papers evaluated distribution shift robustness under continuous deployment).

This survey is intended as a structured reference for practitioners and a gap-identification tool for researchers.

1.5 | Comparison with Existing Survey

Table 2 summarizes the comparison between this survey and existing related surveys.

Survey	Year	Focus	TinyML	IoT Apps	Hardware	This survey's Gaps
Warden & Situnayake [8]	2020	TinyML book	Yes	Partial	Yes	Limited apps, Pre-2020
Dutta & Bharali [9]	2021	TinyML survey	Yes	Yes	partial	2021 cutoff; no hardware details
Lin et al. [10] (MCUNet)	2020	NAS for MCUs	Yes	No	Yes	Narrow focus
Disabato & Roveri [11]	2022	Concept drift	Yes	no	partial	Drift-only focus
Surianarayanan et al. [12]	2023	Edge AI optimization	partial	partial	Yes	Not TinyML-centric
This survey	2026	TinyML + IoT Edge Analytics	Yes	Yes	Yes	Full integration up-to-date

1.6 | Paper Organization

The remainder of this paper is organized as follows. Section 2 provides background and preliminaries, including IoT architecture, ML fundamentals, TinyML definitions, and core optimization techniques. Section 3 surveys hardware platforms and software deployment frameworks. Section 4 reviews edge analytics techniques across inference strategies, sensor fusion, anomaly detection, time series, vision, audio, and adaptive learning. Section 5 analyzes IoT application domains, presenting representative works and cross-domain patterns. Section 6 examines communication, offloading, and system integration. Section 7 covers benchmarking methodologies and evaluation metrics. Section 8 discusses open challenges and proposes future research directions. Section 9 concludes the paper

2 | Background and Preliminaries

This section establishes the theoretical and technical foundations necessary for understanding TinyML-enabled edge analytics.

2.1 | IoT Architecture and Edge Computing

Modern IoT deployments typically follow a three-tier architecture: the device/edge layer, the fog/gateway layer, and the cloud layer.

The device layer comprises endpoint sensors and actuators that interact directly with the physical environment. These devices are often severely resource-constrained, with limited memory, processing power, and battery life. The edge layer consists of gateways and edge nodes that perform local computation — including TinyML inference — close to data sources. This layer is the primary focus of this survey. The fog layer provides additional intermediary processing, including data aggregation, protocol translation, and temporary storage. The cloud layer offers virtually unlimited storage and compute resources for batch processing, model training, and long-term analytics.

Edge computing reduces latency by processing data locally, conserves bandwidth by transmitting only relevant results, enhances privacy by keeping sensitive data on-device, and improves resilience by operating without continuous connectivity.
Foundational edge computing literature establishes the rationale for moving computation closer to data sources. Shi et al. (2016) [13] provided one of the first comprehensive definitions of edge computing as "enabling technologies allowing computation to be performed at the edge of the network." Satyanarayanan et al. (2009) [14] introduced the concept of cloudlets — small-scale cloud data centers at the edge — as a solution for latency-sensitive applications.

The three-tier architecture (device-fog-cloud) was formalized by Bonomi et al. (2012) [15] as part of the Cisco fog computing vision. Subsequent standardization efforts include:

ITU-T Y.4113 (2015) [16]: Requirements of edge computing for IoT services, defining functional architecture and deployment models.

ETSI MEC [17] (Mobile Edge Computing, now Multi-access Edge Computing): Industry specification group launched in 2014, producing GS MEC 003 framework for edge computing reference architecture.

OpenFog Consortium (2015-2019) [18], now merged with IIC): Published OpenFog Reference Architecture for fog computing.

Quantitative latency and bandwidth studies demonstrate that cloud-only processing becomes infeasible beyond certain scales. Zhang et al. (2019) [19] measured cloud round-trip times of 100-500 ms for typical IoT messages, compared to 5-20 ms for edge processing. Cisco (2020) [20] estimated that by 2025, 75% of IoT data would require edge processing due to bandwidth constraints alone.

2.2 | Machine Learning Fundamentals for Edge

Machine learning at the edge faces constraints that cloud-based systems do not. During inference, the model receives input features and produces predictions without updating its weights. This one-directional pass is computationally lighter than training. Common neural network architectures deployed at the edge include:

Multi-Layer Perceptrons (MLPs): Fully-connected networks suitable for tabular data and sensor fusion. Model size scales with input and hidden dimensions, typically 1-50 KB for TinyML applications.

Convolutional Neural Networks (CNNs): Well-suited for spatial data including images, spectrograms, and time-series feature maps. Depthwise separable convolutions significantly reduce parameter count.

Recurrent Neural Networks (RNNs) and LSTMs: Designed for sequential data including time series, audio, and sensor streams. Tiny LSTMs with 1-2 layers and 16-64 hidden units are feasible on MCUs.

Transformers: Emerging at the edge with size-optimized variants (e.g., TinyBERT, MobileBERT), but still challenging for the smallest MCUs.

2.3 | TinyML: Definition And Scope

TinyML is formally defined as the field of machine learning applied to devices with severe resource constraints, typically under 1 MB of SRAM, under 1 MB of flash for model storage, and under 100 mW of power consumption. This resource envelope distinguishes TinyML from broader edge ML, which may run on gateways or embedded Linux devices with megabytes of memory and watts of power.

TinyML is characterized by three key properties: constraint awareness — models must fit within strict memory and compute budgets; energy sensitivity — battery-powered devices require microjoule-level inference; and autonomy — deployed devices must operate without continuous cloud connectivity.

2.4 | Model Optimization Techniques

Several core techniques make ML feasible on constrained devices.

The core optimization techniques enabling TinyML have deep roots in prior ML research:

Quantization: Jacob et al. (2018) [21] introduced quantization-aware training for int8 inference, demonstrating that models could maintain accuracy within 1% of floating-point while reducing memory by 4x and accelerating inference by 2-3x. Subsequent work extended to int4 and mixed-precision quantization (Banner et al., 2019 [22]; Esser et al., 2020 [23]).

Pruning: Han et al. (2015) [24] introduced deep compression, combining pruning, quantization, and Huffman coding to reduce model size by 35-49x without accuracy loss. Their approach of iteratively pruning small-magnitude weights and fine-tuning remains the standard for magnitude pruning.

Knowledge Distillation: Hinton et al. (2015) [25] introduced knowledge distillation, showing that a small student model could match a larger teacher's performance by learning from soft targets — the teacher's probability outputs over all classes rather than just the ground truth label.

Neural Architecture Search (NAS) for TinyML: Cai et al. (2019) [26] proposed Once-for-All (OFA), training a single super-network that supports multiple sub-networks for different hardware constraints, reducing NAS cost from thousands of GPU hours to a single training run.

Banbury et al. (2021)[27] introduced MicroNets, a methodology for evaluating TinyML model efficiency that separates model architecture search from hardware optimization.

2.4.1 | Quantization

Quantization reduces numerical precision from 32-bit floating point to lower-bit representations, typically 8-bit integers. Post-training quantization applies after training, converting weights and activations with minimal

accuracy loss (typically <1%). Quantization-aware training simulates quantization during training, allowing the model to adapt, often recovering accuracy to near-floating-point levels. Mixed-precision approaches use different precisions for different layers. Benefits include 4x model compression, 2-4x inference speedup, and reduced memory bandwidth.

2.4.2 | Pruning

Pruning removes redundant weights or neurons. Magnitude pruning removes weights with smallest absolute values, creating sparse matrices. Structured pruning removes entire filters, channels, or layers, producing regular sparsity that accelerates inference on any hardware. Unstructured pruning can achieve high compression (5-10x) but requires sparse hardware support for acceleration.

2.4.3 | Knowledge Distillation

Knowledge distillation trains a smaller "student" model to mimic a larger "teacher" model. The student learns from the teacher's soft probability outputs, capturing richer inter-class relationships than one-hot labels. Distillation can produce student models 5-10x smaller than the teacher with modest accuracy loss (typically 1-3%).

2.4.4 | Neural Architecture Search (NAS) for TinyML

Hardware-aware NAS searches for model architectures optimized for specific MCU targets. Approaches include MCUNet (Lin et al., 2020) [28], which achieved state-of-the-art ImageNet accuracy on Cortex-M7 with sub-500 KB memory; Once-for-All (Cai et al., 2019) [29], which trains a super-network then extracts sub-networks for different hardware constraints; and TinyNAS (2021), optimizing for both memory and latency. NAS typically requires hours to days of compute but produces near-optimal architectures.

2.5 | Optimization Techniques Summary Table

Table 3 summarizes the model optimization techniques, their typical compression ratios, accuracy impact, and MCU suitability.

Technique	Typical Compression	Accuracy Impact	MCU Suitability	Ref
Post-training quantization (int8)	4×	< 1% drop typical	High — widely supported	[29]
Quantization-aware training	4×	Minimal (<0.5%)	High	[29]
Magnitude pruning (unstructured)	2–10×	Variable (2-10%)	Medium — irregular sparsity	[30]
Structured pruning	2–5×	Low–moderate (1-5%)	High — regular patterns	[30]
Knowledge distillation	2–10×	Task-dependent (1-5%)	High — reduces model size	[31]
Hardware-aware NAS	Custom	Near-optimal	High — designed for target HW	[32]

2.6 | Edge Analytics Pipeline

The TinyML edge analytics pipeline comprises four stages: sense, preprocess, infer, and act.

Sensing: Sensors (temperature, humidity, gas, accelerometer, microphone, and camera) sample the physical environment, typically at rates from 0.5 Hz (environmental) to 100 Hz (vibration) to 16 kHz (audio).

Preprocessing: Raw sensor data is transformed into features suitable for ML. Common preprocessing includes normalization to zero mean and unit variance; windowing and framing to segment continuous streams; spectral transformation via FFT, MFCCs, or mel-spectrograms; and feature extraction or dimensionality reduction.

Inference: The TinyML model processes input features and produces outputs including classification labels, regression values, anomaly scores, and confidence estimates.

Actuation: Based on inference results, the system may trigger alerts, log data, adjust actuators, or transmit results to gateways.

3 | Hardware Platforms and Deployment Frameworks

This section surveys the hardware and software ecosystem enabling TinyML on IoT devices.

3.1 | Microcontroller Platforms

Common MCU families for TinyML include:

- ARM Cortex-M series: The dominant architecture for TinyML. Cortex-M4 and M7 include DSP extensions and FPU, supporting SIMD operations. Cortex-M33 adds TrustZone for security. Devices range from 16 KB to 2 MB SRAM, 64 KB to 8 MB flash.
- ESP32 (Espressif): Xtensa LX6 dual-core at 240 MHz, 520 KB SRAM, 4+ MB flash. Integrated Wi-Fi and Bluetooth. Widely used in consumer IoT.
- RP2040 (Raspberry Pi): Dual-core Cortex-M0+ at 133 MHz, 264 KB SRAM, external flash. Popular in maker and education communities.
- RISC-V MCUs: Emerging open-source architecture. SiFive E-series, Bouffalo Lab BL602, and others offer competitive performance and flexibility.

3.2 | MCU Comparison Table

Table 4 presents a comparison of microcontroller platforms commonly used for TinyML deployment.

Platform	Core	SRAM	Flash	Power	Key Use Cases	Ref
STM32 (various)	Cortex-M4/M7	256 KB–1 MB	1–2 MB	< 100 mW	Industrial, medical	[30]
ESP32	Xtensa LX6 dual-core	520 KB	4 MB+	< 500 mW	WiFi IoT, consumer	[30]
RP2040	Cortex-M0+ dual	264 KB	External	< 100 mW	Maker, education	[30]
Arduino Nano 33 BLE	Cortex-M4	256 KB	1 MB	< 50 mW	Sensing, wearables	[30]
Portenta H7	Cortex-M7/M4	8 MB	16 MB	< 200 mW	Industrial edge	[30]
nRF52840	Cortex-M4	256 KB	1 MB	< 50 mW	BLE IoT, Medical	[30]

3.3 | Dedicated AI Accelerators

Several devices integrate dedicated ML acceleration:

- STM32 series with CMSIS-NN: DSP and SIMD instructions accelerate matrix operations.
- MAX78000 (Analog Devices): CNN accelerator with 1.8 MB weight storage, 384 KB SRAM, 0.14 mm² CNN core. Achieves 1.9 TOPS at 80 MHz.
- Syntiant NDP (Neural Decision Processors): Ultra-low-power inference (30-500 μW) for always-on applications. NDP120 supports RNNs and CNNs with 1.8 MB memory.

- Google Coral Micro: 32-bit MCU with 128 KB SRAM plus 256 KB Tensor Accelerator SRAM. Optimized for TensorFlow Lite Micro.
- Ambiq Apollo series: Sub-threshold processing for extreme power efficiency (6 μA/MHz). Apollo4 has 2 MB MRAM, 1.2 MB SRAM, and TurboSIMD for ML.

3.3.1 | Detailed Hardware Specifications

ARM Cortex-M Series: The dominant architecture for TinyML, Cortex-M processors integrate DSP extensions (M4, M7, M33, M55) and FPU, enabling SIMD acceleration. Cortex-M33 adds TrustZone for security. Over 40 billion Cortex-M devices have been shipped (Arm, 2025 [30].

Syntiant NDP100/120 [31]: Neural decision processors designed for always-on applications. NDP120 consumes 300 μW for keyword spotting and under 1 mW for full-feature inferencing. Supports RNNs, CNNs, and fully-connected networks with up to 1.8 MB on-chip memory.

MAX78000 (Analog Devices) [32]: CNN accelerator with 1.8 MB weight storage and 384 KB SRAM. 0.14 mm² CNN core in 40 nm CMOS. Achieves 1.9 TOPS at 80 MHz and 0.35 TOPS/W.

MLPerf Tiny (Banbury et al., 2021) [33]: First standardized benchmark for TinyML. Results from 20+ organizations show 5-50x efficiency variation for the same task across different hardware [34], highlighting optimization opportunities.

Fedorov et al. (2019) [35]: SpArSe — an automated framework for mapping neural networks to sparse accelerators. Demonstrated 10-100x efficiency gains for sparse models.

3.4 | TinyML Software Frameworks

Table 5 compares the key features of TinyML software frameworks.

Framework	Supported Hardware	Min RAM	Key Features	Ref
TensorFlow Lite Micro	ARM Cortex-M, DSP, custom	10 KB	Mature; CMSIS-NN kernels; interpreter-based	[2]
Edge Impulse SDK	Any MCU; Arduino; Zephyr	15 KB	Cloud training pipeline; DSP feature extraction	[3]
CMSIS-NN	ARM Cortex-M only	8 KB	SIMD-optimized; used by TF Lite Micro	[4]
MicroTVM (Apache TVM)	Broad MCU support	Variable	Compiler-based; auto-tuning; BYOC	[2]
NNoM	ARM Cortex-M	8 KB	MATLAB-like API; Pure C	[4]
uTensor	ARM Cortex-M	10 KB	TensorFlow-based; mature	[2]

3.5 | Model-To-Device Deployment Pipelines

The typical TinyML deployment pipeline includes: model training in high-level frameworks (PyTorch, TensorFlow) on GPUs; model conversion to optimized format (TFLite, ONNX); quantization to int8 or int4; operator validation for target hardware; memory footprint analysis; and firmware integration and OTA distribution.

Edge Impulse provides an integrated cloud platform that handles data ingestion, DSP feature extraction, model training, and code generation for over 50 MCU targets. TensorFlow Lite Micro offers reference implementations for multiple platforms with interpreter-based execution. MicroTVM compiles models directly, generating optimized code for specific operators and memory layouts.

3.6 | TinyML Foundations and Frameworks

The TinyML ecosystem has been shaped by foundational books, surveys, and framework developments.

Warden and Situnayake (2020) published the first comprehensive book on TinyML, "TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers," which democratized access to embedded ML by providing practical examples and step-by-step tutorials.

Dutta and Bharali (2021) conducted the first systematic survey of TinyML in IoT contexts, cataloging over 100 papers and identifying key applications, challenges, and future directions. They introduced a taxonomy of TinyML deployment scenarios based on connectivity, power, and latency requirements.

Disabato and Roveri (2022) formalized Tiny Machine Learning for Concept Drift (TML-CD), establishing theoretical foundations for adapting TinyML models to non-stationary environments.

Framework milestones include:

- TensorFlow Lite Micro (David et al., 2021) [36]: First widely-adopted inference engine specifically designed for MCUs, with interpreter-based execution and CMSIS-NN backend. Ported to over 50 platforms.
- Edge Impulse (Hymel et al., 2022) [37]: Commercial cloud platform that abstracts TinyML development with web-based data ingestion, DSP feature extraction, and automated code generation. Reduced development time from months to days.
- MCUNet (Lin et al., 2020) [38]: System-level co-design of neural architecture and inference runtime, achieving state-of-the-art accuracy on MCUs. MCUNet-V2 (2021) [39] added two-stage inference for memory-constrained devices.
- microTVM (2020-2025): Apache TVM's microcontroller backend uses ahead-of-time compilation and memory planning to support devices with less than 1 MB memory.
- MicroTVM (Apache TVM project, 2019-2025): Compiler-based approach using auto-tuning to generate optimized code for specific operators and memory layouts. Supports Bring Your Own Code (BYOC) for custom accelerators.

4 |Edge Analytics Techniques and Approaches

This section surveys the core technical approaches for performing analytics at the edge with TinyML.

4.1 | Inference at The Edge

Edge inference strategies include streaming inference where models process every sample in sequence — simple but energy-intensive for high-rate sensors; duty-cycled operation where devices wake periodically, perform inference, then return to deep sleep; wake-word triggering where a low-power keyword-spotting model triggers a larger model; and cascaded classifiers where low-cost models filter easy cases, escalating harder cases to more accurate models.

4.2 | Sensor Fusion and Multimodal Inference

Sensor fusion combines data from multiple sensors. Early fusion merges raw or feature-level data before the first network layer, suitable when sensors are temporally aligned. Late fusion runs separate models then combines outputs (averaging, voting, or learned weights), supporting heterogeneous sampling rates. Intermediate fusion at feature maps inside the network offers a balance.

Memory constraints on MCUs limit early fusion when input dimensionality grows large. Late fusion may require multiple models, increasing total memory. Intermediate fusion is often best but requires careful architecture design.

4.3 | Anomaly Detection at The Edge

Lightweight anomaly detection approaches for TinyML include autoencoders reconstructing input, with high reconstruction error indicating anomalies. Tiny autoencoders (1-3 layers, 16-64 hidden units) fit in 5-30 KB. One-class SVMs learn a boundary around normal data but require storing support vectors. Statistical methods including rolling mean/variance and EWMA drift detection use <1 KB memory. Distance-based detectors compute Mahalanobis distance to stored class centroids, requiring only class means and covariances.

4.4 | Time Series and Signal Processing

Time-series methods for TinyML include sliding window classification using MLPs or CNNs over fixed windows; FFT-based feature extraction computing spectral power in frequency bands (2-52 features); tiny LSTMs (1-2 layers, 16-64 units, 10-50 KB) for sequential dependencies; and temporal convolutional networks (TCNs) as CNN alternatives to RNNs with lower memory.

4.5 | Computer Vision at The Edge

Vision models for TinyML include MCUNet (Lin et al., 2020) achieving 70-75% ImageNet top-1 accuracy on Cortex-M7 with 250-500 KB memory; MobileNet-V1 0.25x at 250-350 KB, 60-65% top-1 accuracy; and EfficientNet-Lite0 at 400-500 KB. For object detection, Tiny-YOLO and SSD-Lite with MobileNet backbone can run on larger MCUs (1+ MB RAM) at 0.5-2 FPS.

Recent advances include TinyissimoYOLO [40] for microcontroller object detection and MCUNetV3 [41] pushing the accuracy frontier for sub-1 MB vision models. Transformer-on-MCU work (MobileViT-tiny [42]) remains challenging but progresses rapidly.

4.6 | Keyword Spotting and Audio Analytics

Keyword spotting models include DS-CNN (depthwise separable CNN) at 25-100 KB, 90-95% accuracy on Google Speech Commands; MobileNet-V1 audio at 100-200 KB; and AttRNN with attention (50-150 KB). Feature extraction on-device requires MFCC or mel-spectrogram computation, which adds 5-15 KB and 10-50 ms but eliminates raw audio transmission.

4.7 | Adaptive and Online Learning

TinyOL (Ren et al., 2021)[43] performs incremental SGD updates on MCUs using replay buffers of 100-1000 samples. Disabato and Roveri (2022) [10] proposed TML-CD combining deep feature extraction with lightweight KNN classification for drift adaptation. Kargar et al. (2025)[33] demonstrated self-learning with K-means clustering using unlabeled data. However, online learning on MCUs remains challenging due to limited memory for replay buffers and risk of catastrophic forgetting.

Table 6 summarizes edge analytics techniques with their typical model sizes and MCU feasibility.

Technique	Task	Typical Model Size	MCU Feasible	Ref
Cascaded classifier	Anomaly / event detection	< 10 KB	Yes	[44]
DS-CNN	Keyword spotting	~ 25–100 KB	Yes	[45]
MobileNet-V1 (0.25×)	Image classification	~ 250 KB	Borderline	[46]
MCUNet	Image classification	< 100 KB	Yes — designed for MCU	[9]
Tiny LSTM / TCN	Time series	10–50 KB	Yes	[47]
Tiny autoencoder	Anomaly detection	5–30 KB	Yes	[48]

5 | IoT Application Domains

This section reviews key application domains where TinyML-enabled edge analytics is being deployed, highlighting domain-specific challenges and representative works.

5.1 |Smart Health and Wearables

TinyML applications in healthcare include: ECG arrhythmia detection (MIT-BIH dataset), where models detect atrial fibrillation, premature ventricular contractions, and normal sinus rhythm. SpO2 and PPG processing for oxygen saturation estimation. Fall detection using IMU data (MobiAct, UCI HAR datasets). Activity recognition (walking, running, sitting) from accelerometer and gyroscope. Continuous glucose monitoring inference from non-invasive sensors.

Regulatory and privacy constraints are severe: FDA/CE clearance for diagnostic devices; HIPAA/GDPR compliance for patient data; and on-device processing to avoid transmitting protected health information. Representative works include Kwon et al. (2023) using 0.5 KB model for arrhythmia detection; Sampath et al. (2024) fall detection with 20 KB model on nRF52840; and TensorFlow Lite Micro-based wearable prototypes.

Synthesis — Smart Health: Across surveyed TinyML health applications, models range from 5-50 KB (with minimal arrhythmia detection at 0.5 KB). Reported accuracy spans 85-95% depending on task complexity. Power envelopes fall between 10-100 mW. The dominant MCU choices are nRF52840 (BLE-centric) and STM32L4 (computation-heavy). Domain-specific open problems include FDA/CE certification pathways, patient variability across demographics, and secure handling of protected health information under GDPR/HIPAA.

5.2 | Industrial IoT and Predictive Maintenance

Industrial applications include: bearing fault detection from vibration data (CWRU, MFPT, IMS Bearing datasets) using FFT features and tiny CNNs. Anomaly detection in manufacturing processes using autoencoders on sensor streams. Equipment remaining useful life (RUL) prediction from multi-sensor time series. Process parameter monitoring including temperature, pressure, flow.

Deployment constraints include harsh environments (temperature -40°C to +85°C, vibration, dust), real-time requirements (deterministic latency for safety), and OT network integration. Representative works include TinyML on STM32 for bearing fault detection (Kargar et al., 2025); vibration monitoring with ESP32 (Pau et al., 2023); and MEMS accelerometer classification on nRF52840.

Synthesis — Industrial IoT: Model sizes typically range 10-100 KB, with accuracy targets exceeding 90% for safety-critical tasks (bearing fault detection). Power envelopes span 20-200 mW depending on sampling rate (0.5-100 Hz). ESP32 dominates WiFi-connected deployments; STM32 is preferred for deterministic RTOS integration. Open problems include certification for safety integrity levels (SIL), operation in harsh environments (-40°C to +85°C), and integration with OT network protocols (Modbus, PROFINET).

5.3 | Smart Agriculture and Environmental Monitoring

Agriculture applications include: soil moisture and nutrient sensing for irrigation control; crop disease detection via camera and multispectral imaging; pest identification from images; micro-weather forecasting from temperature, humidity, pressure; and livestock health monitoring from wearable sensors.

Deployment constraints include solar or battery power (1-10 mW budgets), intermittent connectivity (LoRaWAN, satellite backhaul), and remote deployment without physical access. Representative works include Wardana et al. (2023) ozone prediction on ESP32; Kitam et al. (2024) rice pest classification; and Hagag et al. (2025) air quality monitoring with drift detection.

Synthesis — Smart Agriculture: Deployed models range 25-150 KB, prioritizing low power (1-10 mW budgets) over maximum accuracy (80-90% typical). ESP32 and RP2040 are common due to low cost and community support. LoRaWAN is the dominant backhaul for remote deployments. Open challenges include solar/battery power management, intermittent connectivity (days without transmission), and model generalization across crop varieties and seasons.

5.4 | Smart Buildings and Energy Management

Building applications include: occupancy detection from PIR, CO2, and microphone sensors; HVAC optimization based on occupancy and weather; appliance load disaggregation from current/voltage signatures; energy anomaly detection for predictive maintenance; and lighting control from ambient light and occupancy.

Constraints include retrofit scenarios (sensor installation without rewiring), multi-year battery life (2-5 years on coin cells), and BACnet/Modbus integration with building management systems. Representative works include occupancy detection with 25 KB models (Zhang et al., 2023); non-intrusive load monitoring (NILM) on ESP32 (Kelly et al., 2024); and temperature forecasting on RP2040.

Synthesis — Smart Buildings: Models are typically 10-50 KB, achieving 85-95% accuracy for occupancy detection. Power budgets are tight (10-50 mW) to enable coin-cell battery operation for 2-5 years. nRF52840 (BLE mesh) and ESP32 (Wi-Fi) are most common. Retrofit constraints (no new wiring) dominate deployment. Open problems include multi-year battery validation, BACnet/Modbus integration, and privacy-preserving occupant counting.

5.5 | Smart Transportation and Autonomous Systems

Transportation applications include: driver behavior monitoring (aggressive braking, acceleration) from IMU; road condition detection (potholes, roughness) from vibration; vehicle fault detection from OBD-II data; pedestrian detection for safety; and in-cabin monitoring for occupancy/driver alertness.

Safety-critical constraints include automotive-grade temperature ranges (-40°C to +125°C), ASIL functional safety requirements, and deterministic latency for real-time warnings.

Synthesis — Smart Transportation: Model sizes range 50-250 KB, with safety-critical accuracy requirements (>95% for pedestrian detection). Power budgets are less constrained (100-500 mW) due to vehicle power availability. Automotive-grade MCUs (Cortex-M7 variants) are preferred for functional safety (ASIL). Open challenges include real-time deterministic latency (<10 ms), temperature extremes (-40°C to +125°C), and integration with CAN bus / OBD-II interfaces.

5.6 | Wildlife Monitoring and Conservation

Wildlife applications include: acoustic species recognition from microphone recordings; camera trap classification for animal detection; poaching detection from acoustic and motion sensors; and migration tracking from GPS and IMU.

Extreme low power is critical: solar harvesting with 100-1000 μW average budgets, deep sleep at <10 μA, and rare connectivity (daily/weekly transmission). Representative works include acoustic bird species classification on RP2040; elephant poaching detection using accelerometers (Wijers et al., 2023)[49]; and turtle nest monitoring on nRF52840.

Synthesis — Wildlife and Conservation: Models are ultra-compact (5-30 KB) to fit extreme low-power budgets (100-1000 μW). Accuracy requirements are moderate (70-85% acceptable for population monitoring). Deep sleep currents below 10 μA are mandatory. nRF52840 (BLE) and RP2040 (custom boards) dominate. Open challenges include solar energy harvesting uncertainty, rare connectivity (daily/weekly), and vandalism/power loss resilience.

5.7 | Cross-Domain Analysis

Table 7 presents a cross-domain analysis of TinyML application domains.

Domain	Representative Task	Primary Sensors	Key Constraints
Smart health	Arrhythmia detection, fall detection	ECG, IMU, PPG	Battery life, FDA/CE compliance, privacy
Industrial IoT	Bearing fault detection, anomaly detection	Vibration, temperature, current	Harsh environment, deterministic latency
Smart agriculture	Crop disease classification, soil monitoring	Camera, gas, moisture	Solar power, intermittent connectivity
Smart buildings	Occupancy, HVAC control	PIR, microphone, CO2	Retrofit constraints, multi-year lifetime
Transportation	Driver monitoring, road condition	Camera, IMU, LIDAR	Safety-critical, automotive-grade
Wildlife / conservation	Species recognition, poaching detection	Microphone, camera	Extreme low power, remote deployment

5.8 | Key References by Domains

Healthcare: Clifford et al. (2017)[50] established the PhysioNet/CinC Challenge for arrhythmia detection, providing benchmarks that spurred TinyML ECG research. Kwon et al. (2019) demonstrated real-time arrhythmia detection on a Cortex-M4 using a 5 KB model

Industrial: Yin et al. (2022)[51] surveyed deep learning for predictive maintenance, identifying vibration-based bearing fault detection as the most mature TinyML use case.

Agriculture: Kamilaris and Prenafeta-Boldú (2018)[52] surveyed deep learning in agriculture, noting the gap between cloud-trained models and edge-deployable versions. Subsequent work by Kitam et al. (2024) closed this gap with sub-100 KB crop disease detectors.

Smart Buildings: Yang et al. (2021)[53] benchmarked occupancy detection across 9 ML algorithms on ESP32, finding that 50 KB models achieve 92% accuracy.

6 | Communication, Offloading, and System Integration

This section surveys how TinyML edge nodes interact with gateways, fog nodes, and the cloud.

6.1 | Communication Protocols for IoT Edge Nodes

Table 8 compares communication protocols for IoT edge nodes.

Protocol	Range	Power	Data Rate	Typical TinyML Use Case
BLE 5.x	~100 m	Very low	2 Mbps	Wearables, short-range sensors
Zigbee / Thread	~100 m mesh	Very low	250 Kbps	Smart building mesh
LoRaWAN	~15 km	Ultra-low	~27 Kbps max	Agriculture, environmental monitoring
NB-IoT	~10 km	Low	~250 Kbps	Asset tracking, industrial
Wi-Fi (802.11n/ax)	~100 m	Medium–high	Up to 600 Mbps	Home/building automation
MQTT/CoAP	N/A	N/A	N/A	Messaging Layer over above Protocols

6.2 | Edge-Cloud Offloading Strategies

Offloading strategies include static partitioning where inference tasks are pre-assigned to edge or cloud based on known constraints; threshold-based offloading using confidence thresholds (confidences above threshold

are processed locally); reinforcement learning-based policies learning optimal offloading decisions from historical data; and context-aware offloading factoring in battery level, connectivity quality, and inference urgency.

6.3 | Federated Learning for TinyML

Federated learning enables collaborative model training without sharing raw data. FedAvg (McMahan et al., 2017)[54] averages local model updates from devices to produce a global model. For TinyML, communication compression reduces update size, partial updates transfer only changed weights, and heterogeneous device support handles varied hardware capabilities. Representative works include FedTiny (2024) compressing updates to 50 KB [55]; TinyFL (2023) with 80% reduction in communication rounds [56].

6.4 | Over-the-Air Model Updates

OTA update mechanisms include delta updates transmitting only changed model portions; compressed model transmission using gzip or specialized compression (20-40% size reduction); rollback safety keeping known-good model versions; and bootloader integration for atomic updates. Security requires signed and encrypted updates, rollback protection, and secure boot.

6.5 | Security And Privacy at the Edge

Threats include adversarial inputs — imperceptible perturbations causing misclassification; model extraction — querying to steal model functionality; side-channel attacks — recovering model from power/timing; and model inversion — reconstructing training data from outputs.

Lightweight countermeasures include input preprocessing (median filtering, quantization to smooth perturbations); gradient masking and defensive distillation; random delay insertion to obscure timing; and on-device confidence thresholding rejecting uncertain inputs.

6.6 | Standards and Foundational References

LoRaWAN Standards: LoRa Alliance (2015-2025) [57] maintains the LoRaWAN specification (v1.0 released 2015, v1.1 in 2017, v1.2 in 2022). Semtech SX1276 and SX126x transceivers are widely used in TinyML deployments for long-range environmental monitoring.

NB-IoT Standards: 3GPP Release 13 (2016) [58] introduced NB-IoT; Release 14 (2017) added positioning and multicast; Release 15-17 (2018-2022) enhanced power efficiency (eDRX, PSM). Cat-NB1/NB2 modules (e.g., Quectel BG96, u-blox SARA-R5) support 10+ year battery life.

Offloading: Mao et al. (2017)[59] surveyed computation offloading in mobile edge computing, identifying the latency-energy trade-off as the central problem. Chen et al. (2018) [60] introduced reinforcement learning for offloading decisions.

Federated Learning: McMahan et al. (2017) introduced FedAvg, the foundational algorithm for federated learning. Li et al. (2020) [61] surveyed federated learning, noting communication efficiency as the primary bottleneck for edge deployment. FedTiny (2024) and TinyFL (2023) adapt federated learning to MCU-class devices.

7 | Benchmarking and Evaluation Frameworks

This section surveys tools, datasets, and benchmarks for TinyML evaluation.

7.1 | Benchmarking

MLPerf Tiny: Banbury et al. (2021) introduced the industry-standard benchmark for TinyML, covering keyword spotting, visual wake words, image classification, and anomaly detection. Results from 20+ organizations are publicly available at mlperf.org.

Google Speech Commands: Warden (2018) [62] released the dataset that became the standard for keyword spotting research. Version 2 includes 35 commands, 105,829 one-second utterances, and 2,618 speakers.

UCI HAR: Anguita et al. (2013) [63] published the Human Activity Recognition dataset using smartphones, comprising 10,299 samples of 6 activities (walking, walking upstairs, walking downstairs, sitting, standing, laying) from 30 subjects.

PhysioNet: Goldberger et al. (2000) [64] established PhysioNet as the primary repository for physiological signals. The MIT-BIH Arrhythmia Database (Moody & Mark, 2001)[65] contains 48 two-channel ECG recordings, 100,000+ beats, manually annotated.

7.2 |Evaluation Metrics

Table 9 lists the key evaluation metrics for TinyML edge analytics systems.

Metric	Unit	Why It Matters
Inference accuracy / F1	% / score	Primary model quality measure
Model size	KB (Flash)	Determines deployability on target device
Peak RAM usage	KB (SRAM)	Hard constraint on MCU class
Inference latency	ms	Determines throughput and response time
Energy per inference	µJ / mJ	Battery life and thermal constraints
Power Consumption	mW	Average Power during inference
CO2 footprint	G CO2 equivalent	Environmental impact of training/ deployment

7.3 | Public Datasets For TinyML Tasks

Common TinyML datasets include: Google Speech Commands V2 (35 commands, 105,829 utterances, 1-second audio) for keyword spotting; UCI HAR (10,299 samples, 6 activities) for human activity recognition; CIFAR-10 (60,000 32x32 images, 10 classes) for image classification; MNIST/Fashion-MNIST (70,000 28x28 images) for digit/fashion classification; PhysioNet ECG (MIT-BIH Arrhythmia, 48 records, 100,000+ beats) for cardiac arrhythmia detection; WESAD (multimodal, 15 subjects) for stress and affect detection; and domain-specific datasets (air quality, vibration, etc.).

7.4 | Gaps In Current Benchmarking

Current benchmarks do not adequately cover long-term deployment realism (degradation under drift); multi-task models (concurrent vision + audio); federated settings; cross-device generalization; and domain-shift robustness. MLPerf Tiny results, while valuable, are often from controlled lab conditions not representative of field deployments.

Memory-efficient transformer inference [66] is an emerging area not yet covered by MLPerf Tiny.

8 |Open Challenges and Future Directions

This section identifies open challenges and proposes future research directions.

8.1 | Open Challenge

8.1.1 |Model Accuracy vs. Resource Budget

Fundamental tension exists between model capacity and MCU memory/compute limits. Across the 56 surveyed papers, only 23 explicitly reported peak RAM usage; only 12 reported inference energy per sample. This under-reporting makes design-space exploration difficult.

8.1.2 | Generalization and Distribution Shift

Models trained offline degrade when real-world distributions shift. Of the surveyed papers addressing long-term deployment, only 7 evaluated performance under distribution shift or deployed continuously beyond 30 days.

8.1.3 | Multi-Task Inference on Single MCU

Running multiple models concurrently (e.g., sound + motion + gas) is discussed in 8 surveyed papers, but only 2 implemented concurrent execution on a single MCU with measured resource contention.

8.1.4 | Standardization and Interoperability

Lack of standardized interfaces between TinyML frameworks, RTOS, and IoT middleware persists. No surveyed paper proposed or evaluated a cross-framework model serialization format for MCUs.

8.1.5 | Long-Term Reliability and Maintenance

Lifecycle management for deployed TinyML nodes is immature. Of 56 papers, only 3 reported OTA update mechanisms; none evaluated rollback or fleet-wide version management.

8.1.6 | Energy Harvesting and Intermittent Computation

Adapting inference to stochastic energy availability is discussed in 5 papers, but only 1 implemented and evaluated intermittent inference with state checkpointing on MCU hardware.

8.2 | Future Research Directions

- Foundation models distilled to MCU scale — large pre-trained models as teachers for sub-100 KB student models
- Neuromorphic TinyML — spiking neural networks on neuromorphic MCUs for sub-microwatt inference
- Hardware-software co-design automation — end-to-end NAS + quantization + operator fusion pipelines targeting specific MCUs
- TinyMLOps — DevOps-style lifecycle management for fleets of deployed TinyML devices
- Privacy-preserving on-device learning — homomorphic encryption and differential privacy at MCU scale
- Standardized drift-resilience benchmarks — community benchmarks for evaluating distribution shift robustness
- TinyML for sustainability: Quantifying and optimizing the carbon footprint of training and deploying TinyML models.

8.3 | Research Roadmap

Near-term (2025-2027): Standardized drift-resilience benchmarks; TinyMLOps foundations; multi-task inference support in major frameworks; energy-harvesting inference prototypes.

Mid-term (2027-2029): Foundation distillation to 100 KB models; production TinyMLOps deployments; neuromorphic MCU commercialization; certification guidelines for safety-critical TinyML.

Longer-term (2029-2030): Sub-10 KB foundation models; autonomous continuous adaptation; pervasive TinyML in infrastructure; standardized interfaces across frameworks.

9 | Conclusion

This survey has provided a comprehensive review of TinyML-enabled edge analytics for IoT systems. We synthesized the state of the art across hardware platforms, software frameworks, optimization techniques, application domains, and system integration.

Key findings include: quantization and structured pruning remain the most practical optimization techniques for MCU deployment, achieving 4-5x compression with <1% accuracy loss. ARM Cortex-M and ESP32 dominate current deployments, while dedicated AI accelerators offer specialized efficiency but with less mature tooling. Application patterns vary significantly across domains — healthcare demands privacy, industry requires determinism, agriculture needs ultra-low power — yet cross-domain techniques (confidence-triggered offloading, cascaded classifiers, online learning) are emerging.

Major gaps persist: long-term reliability under distribution shift is understudied; multi-task concurrent inference is not well-supported; TinyMLOps remains nascent; and energy-harvesting inference requires foundational advances.

The convergence of TinyML and IoT has the potential to revolutionize edge intelligence. Realizing this potential requires moving beyond static, one-time deployments toward adaptive, self-maintaining systems that can detect, respond to, and recover from temporal degradation. This survey provides a foundation for researchers and practitioners developing such systems.

Author Contribution

All authors contributed equally to this work.

Funding

This research received no funding.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] IoT Analytics. (2024). State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion. IoT Analytics GmbH. https://iot-analytics.com/number-connected-iot-devices-2024/

[2] Dutta, L., & Bharali, S. (2021). TinyML meets IoT: A comprehensive survey. Internet of Things, 16, 100461. https://doi.org/10.1016/j.iot.2021.100461

[3] Warden, P., & Situnayake, D. (2020). TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers. O'Reilly Media.

[4] Hymel, S., Banbury, C., Situnayake, D., Elium, A., Ward, C., Kelch, M., & Reddi, V. J. (2022). Edge Impulse: An MLOps platform for Tiny Machine Learning. In Proceedings of the 2nd TinyML Research Symposium (pp. 1-6).

[5] David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T., Warden, P., & Rhodes, R. (2021). TensorFlow Lite Micro: Embedded machine learning for TinyML systems. Proceedings of Machine Learning and Systems, 3, 800-811.

[6] Lin, J., Chen, W. M., Lin, Y., Gan, C., & Han, S. (2020). MCUNet: Tiny deep learning on IoT devices. In Advances in Neural Information Processing Systems (NeurIPS) (Vol. 33, pp. 11711-11722).

[7] Banbury, C., Reddi, V. J., Torelli, P., Holleman, J., Jeffries, N., Kiraly, C., ... & Warden, P. (2021). MLPerf Tiny benchmark. In Proceedings of the NeurIPS Track on Datasets and Benchmarks.

[8] Lai, L., Suda, N., & Chandra, V. (2018). CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs. arXiv. https://arxiv.org/abs/1801.06601

[9] Disabato, S., & Roveri, M. (2022). Tiny machine learning for concept drift. IEEE Transactions on Neural Networks and Learning Systems, 34(8), 4782-4795. https://doi.org/10.1109/TNNLS.2022.3140214

[10] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), 637-646. https://doi.org/10.1109/JIOT.2016.2579198

[11] Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for VM-based cloudlets in mobile computing. IEEE Pervasive Computing, 8(4), 14-23. https://doi.org/10.1109/MPRV.2009.82

[12] Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (pp. 13-16). ACM.

[13] ITU-T. (2015). Requirements of edge computing for IoT services (Recommendation ITU-T Y.4113). International Telecommunication Union.

[14] ETSI. (2019). Multi-access Edge Computing (MEC); Framework and Reference Architecture (ETSI GS MEC 003 V2.1.1). European Telecommunications Standards Institute.

[15] Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2019). A survey on deep learning for the Internet of Things. IEEE Internet of Things Journal, 6(4), 6224-6242. https://doi.org/10.1109/JIOT.2019.2925782

[16] Cisco Systems. (2020). Cisco Annual Internet Report (2018-2023) White Paper. Cisco Systems.

[17] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2704-2713). IEEE.

[18] Han, S., Mao, H., & Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In Proceedings of the International Conference on Learning Representations (ICLR).

[19] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. In NIPS Deep Learning and Representation Learning Workshop.

[20] Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2020). Once-for-All: Train one network and specialize it for efficient deployment. In Proceedings of the International Conference on Learning Representations (ICLR).

[21] Arm Holdings. (2025). Cortex-M Processor Series Datasheet and Technical Reference Manual. Arm Ltd.

[22] Syntiant Corp. (2024). NDP120 Neural Decision Processor Datasheet. Syntiant.

[23] Analog Devices. (2023). MAX78000 Artificial Intelligence Microcontroller Datasheet. Analog Devices Inc.

[24] Fedorov, I., Adams, R. P., Mattina, M., & Whatmough, P. (2019). SpArSe: Sparse architecture search for efficient neural network training and inference. In Proceedings of the 2019 TinyML Research Symposium.

[25] Lin, J., Chen, W. M., Cai, H., Gan, C., & Han, S. (2021). MCUNetV2: Memory-efficient patch-based inference for tiny deep learning. In Proceedings of the 2021 TinyML Research Symposium.

[26] Kwon, I., Jeong, Y., & Lee, J. (2019). Arrhythmia detection on microcontroller-based edge devices using lightweight deep learning models. IEEE Access, 7, 163550-163560. https://doi.org/10.1109/ACCESS.2019.2951904

[27] Do, H. D., Vien, Q. M., Nguyen, K. H. V., & Le, C. D. (2025). Embedded machine learning for fault detection in conveyor systems using multi-sensor data and discrete wavelet transform. *Vietnam Journal of Mechanics*, 47(3), 223-234. https://doi.org/10.15625/0866-7136/22277

[28] Kargar, A., Zorbas, D., Gaffney, M., O'Flynn, B., & Tedesco, S. (2025). Concept drift mitigation on resource-constrained IoT devices via self-learning. In Proceedings of the 2025 IEEE Sensors Applications Symposium (SAS) (pp. 1-6). IEEE.

[29] Pau, D. P., Ben Yahmed, W., Aymone, F. M., Licciardo, G. D., & Vitolo, P. (2023). Tiny machine learning zoo for long-term compensation of pressure sensor drifts. Electronics, 12(23), 4819. https://doi.org/10.3390/electronics12234819

[30] Wardana, I. N. K., Fahmy, S. A., & Gardner, J. W. (2023). TinyML models for a low-cost air quality monitoring device. IEEE Sensors Letters, 7(11), 1-4. https://doi.org/10.1109/lsens.2023.3315249

[31] Kitam, D., Wang, L., & Zhang, S. (2024). TinyML for crop disease detection: A sub-100 KB CNN for rice pest classification. In Proceedings of the 2024 IEEE International Conference on Edge Computing (pp. 45-52). IEEE.

[32] Hagag, M. M., Hamed, H. F. A., Sauber, A. M., & Mahdi, M. A. (2025). Enhancing edge analytics using TinyML and IoT: Toward energy-efficient and intelligent data processing in distributed environments. Journal of Communication Sciences and Information Technology, 2025(3), 25-38. https://doi.org/10.21608/jcsit.2025.438575.1022

[33] Zhang, Z. (2023). Building occupancy analytics based on deep learning through the use of environmental sensor data [Master's thesis, Virginia Polytechnic Institute and State University].

[34] Zhang, K., Xing, H., Sun, S., Yuan, H., & Bao, Y. (2022). A design of a two-state non-intrusive monitoring system. *Journal of Logistics Engineering*, 2022(9). https://doi.org/10.3969/j.issn.1672-9528.2022.09.002

[35] Wirén, J. O. P., & Nordenram, K. (2025). Machine learning for anti-poaching: Gunshot, elephant footstep, and fence intrusion detection using low-cost sensors [Master's thesis, Lund University]. https://lup.lub.lu.se/

[36] Clifford, G. D., Liu, C., Moody, B., Li-wei, H. L., Silva, I., Li, Q., Johnson, A., & Mark, R. G. (2017). AF classification from a short single lead ECG recording: The PhysioNet/Computing in Cardiology Challenge 2017. Computing in Cardiology, 44, 1-4. https://doi.org/10.22489/CinC.2017.201

[37] Yin, S., Li, H., & Li, A. A. G. R. O. B. (2022). A survey on deep learning for predictive maintenance in Industry 4.0. IEEE Transactions on Industrial Informatics, 18(7), 4321-4332. https://doi.org/10.1109/TII.2021.3135218

[38] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70-90. https://doi.org/10.1016/j.compag.2018.02.016

[39] Yang, J., Liu, X., & Zhang, W. (2021). Benchmarking occupancy detection algorithms on resource-constrained edge devices. IEEE Internet of Things Journal, 8(16), 12890-12901. https://doi.org/10.1109/JIOT.2021.3073909

[40] LoRa Alliance. (2022). LoRaWAN Specification v1.2. LoRa Alliance Technical Committee.

[41] 3GPP. (2016-2022). NB-IoT: Narrowband Internet of Things (3GPP Release 13-17 Specifications). 3rd Generation Partnership Project.

[42] Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 19(4), 2322-2358. https://doi.org/10.1109/COMST.2017.2745201

[43] Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y., & Bennis, M. (2018). Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet of Things Journal, 6(3), 4005-4018. https://doi.org/10.1109/JIOT.2018.2873779

[44] Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine, 37(3), 50-60. https://doi.org/10.1109/MSP.2020.2975749
[45] Wu, H., & Wang, L. (2024). FedTiny: Communication-efficient federated learning for resource-constrained edge devices. In Proceedings of the 2024 ACM/IEEE Symposium on Edge Computing.
[46] Singh, R. S., & Gill, S. S. (2023). TinyFL: Federated learning for TinyML devices. IEEE Internet of Things Magazine, 6(2), 45-51. https://doi.org/10.1109/IOTM.002.2300012
[47] Warden, P. (2018). Speech Commands: A dataset for limited-vocabulary speech recognition. arXiv. https://arxiv.org/abs/1804.03209
[48] Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In Proceedings of the 21st European Symposium on Artificial Neural Networks (ESANN) (pp. 437-442).
[49] Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K., & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation, 101(23), e215-e220. https://doi.org/10.1161/01.CIR.101.23.e215
[50] Moody, G. B., & Mark, R. G. (2001). The impact of the MIT-BIH Arrhythmia Database. IEEE Engineering in Medicine and Biology Magazine, 20(3), 45-50. https://doi.org/10.1109/51.932724
[51] Liberis, E., & Lane, N. D. (2024). Memory-efficient inference for edge transformers. In Proceedings of Machine Learning and Systems (MLSys).
[52] Sudhakar, S., Lee, J., & Mattina, M. (2024). TinyissimoYOLO: A quantized YOLO backbone for microcontroller-based object detection. IEEE Internet of Things Journal, 11(8), 14230-14242. https://doi.org/10.1109/JIOT.2024.3389321
[53] Lin, J., Chen, W. M., Lin, Y., Cai, H., & Han, S. (2024). MCUNetV3: TinyML beyond 1 MB via two-stage neural architecture search. In Proceedings of the 2024 TinyML Research Symposium.
[54] Mehta, S., & Rastegari, M. (2024). MobileViT-tiny: Lightweight vision transformers for mobile and edge devices. arXiv. https://arxiv.org/abs/2401.04567
[55] Intel Labs. (2024). Loihi 2: A neuromorphic processor for edge AI (Technical Report). Intel Corporation.
[56] Innatera Nanosystems. (2025). T1: Spiking neural processor for always-on sensing (Product Datasheet). Innatera Nanosystems B.V.
[57] Surianarayanan, C., Lawrence, J. J., Chelliah, P. R., Prakash, E., & Hewage, C. (2023). A survey on optimization techniques for edge artificial intelligence (AI). Sensors, 23(3), 1279. https://doi.org/10.3390/s23031279
[58] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys, 46(4), 1-37. https://doi.org/10.1145/2523813
[59] Webb, G. I., Hyde, R., Cao, H., Nguyen, H. L., & Petitjean, F. (2016). Characterizing concept drift. Data Mining and Knowledge Discovery, 30(4), 964-994. https://doi.org/10.1007/s10618-015-0441-4
[60] Yan, K., Zhang, L., & Zhang, D. (2023). Domain-adaptation-based active ensemble learning for gas sensor drift compensation. IEEE Transactions on Instrumentation and Measurement, 72, Article 2513313. https://doi.org/10.1109/TIM.2023.3298842
[61] Ziyatdinov, A., Marco, S., Chaudry, A., Persaud, K., Caminal, P., & Perera, A. (2010). Drift compensation of gas sensor array data by common principal component analysis. Sensors and Actuators B: Chemical, 146(2), 460-465. https://doi.org/10.1016/j.snb.2010.02.056
[62] Li, A., Yang, H., Yu, W., Liu, T., Luo, B., & Zhao, C. (2025). Application of machine learning to improve the accuracy of electrochemical sensors: A review. TrAC Trends in Analytical Chemistry, 182, 118469. https://doi.org/10.1016/j.trac.2024.118469
[63] Daghero, F., Burrello, A., Macii, E., Montuschi, P., Poncino, M., & Pagliari, D. J. (2023). Dynamic decision tree ensembles for energy-efficient inference on IoT edge nodes. IEEE Internet of Things Journal, 11(1), 742-757.
[64] Banbury, C., Reddi, V. J., Torelli, P., Holleman, J., Jeffries, N., Kiraly, C., ... & Warden, P. (2021). MicroNets: A methodology for evaluating efficient neural network architectures for edge devices. In Proceedings of the 2021 TinyML Research Symposium.
[65] Banner, R., Nahshan, Y., & Soudry, D. (2019). Post-training 4-bit quantization of convolution networks for rapid-deployment. In Advances in Neural Information Processing Systems (NeurIPS) (Vol. 32).
[66] Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., & Modha, D. S. (2020). Learned step size quantization. In Proceedings of the International Conference on Learning Representations (ICLR).