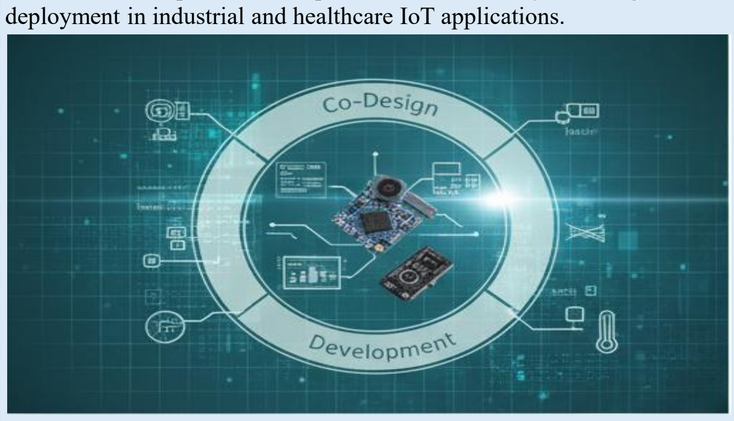
TinyML-CoDE: A Co-Design and Evaluation Framework for Systematic TinyML Development

Dhananjaya Lahiru Bandara Alwitigala
School of Digital, Technology, Innovation and Business, Department of Computing and Esports, University of Staffordshire.
CB016217@students.apiit.lk , a0321000@student.staffs.ac.uk

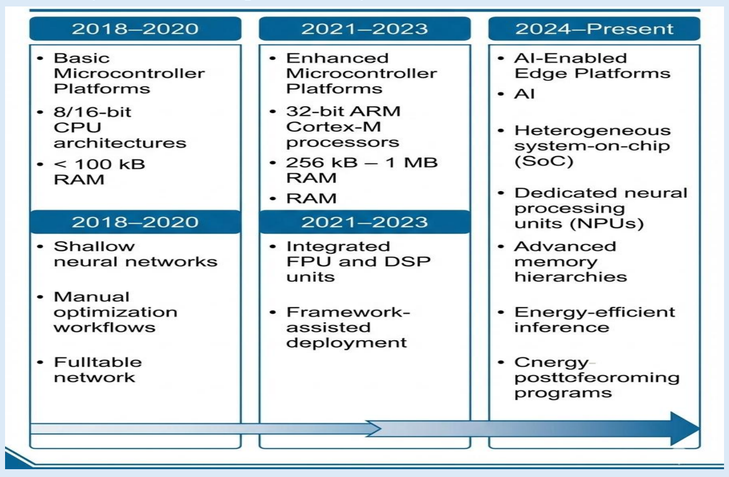
Abstract

This research addresses the fragmented development landscape of Tiny Machine Learning (TinyML) in resource-constrained IoT devices. Through a systematic review of 35 publications, five core research gaps are identified: standardized benchmarking, systematic co-design, lifecycle management, security integration, and toolchain interoperability. The proposed TinyML-CoDE framework integrates constraint-aware specification, hardware-model recommendation, automated optimization, unified benchmarking, and lifecycle management. Expected benefits include 30% reduction in development time and 25% improvement in performance efficiency, enabling scalable deployment in industrial and healthcare IoT applications.



Introduction and Motivation

The rapid expansion of IoT demands intelligent, low-power computing at the edge, driving the adoption of Tiny Machine Learning (TinyML) on microcontrollers. While TinyML enables AI on devices with <1MB memory and milliwatt power budgets, its development is hindered by toolchain fragmentation, unreliable evaluation methods, and poor hardware-software compatibility. Current TinyML development relies heavily on manual tuning and siloed tools, leading to prolonged development cycles, suboptimal solutions, and limited scalability. This paper addresses these challenges by proposing a systematic co-design framework to streamline development, enhance efficiency, and ensure reproducibility.



Research Questions

RQ1: How can standardized benchmarking enable fair comparison across diverse hardware?

RQ2: What systematic co-design principles optimize hardware-software integration?

RQ3: How can toolchain interoperability reduce development complexity?

RQ4: What lightweight security measures are feasible for resource-constrained deployments?

RQ5: How can lifecycle management sustain TinyML systems at scale?

RQ6: How does systematic co-design improve efficiency and maintainability?

Research Objectives

RO1: Design a unified benchmarking model for TinyML across accuracy, latency, memory, and energy.
RO2: Develop a systematic co-design approach considering hardware-software constraints.
RO3: Design an interoperable toolchain for seamless TinyML development.
RO4: Implement lightweight, resource-aware security solutions.
RO5: Build lifecycle management tools for deployed TinyML systems.
RO6: Validate the framework through industrial and healthcare case studies.

Literature Review Highlights

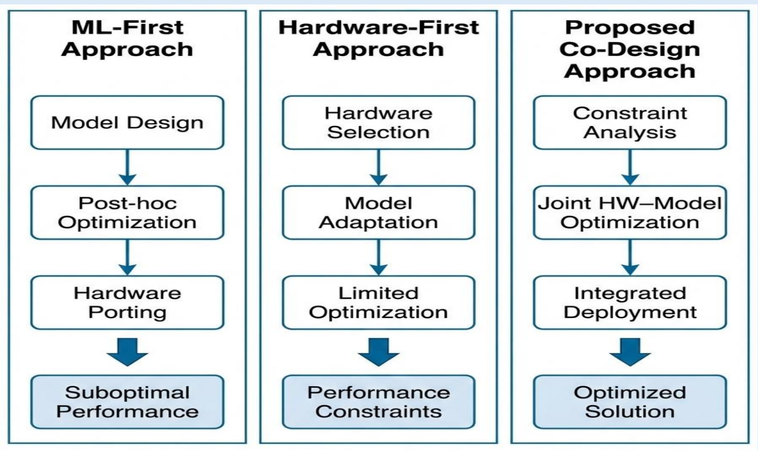
Hardware Diversity: MCU platforms vary in compute, power, connectivity, and cost.

Software Fragmentation: Incompatible toolchains lead to accuracy/performance losses.

Optimization Challenges: Quantization, pruning, and NAS methods are hardware-specific.

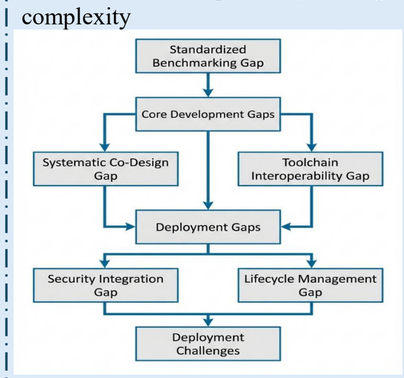
Generalization Issues: Lab-optimized models fail in real-world conditions.

Methodological Shortcomings: Only 22% of studies are reproducible; 15% include energy measurements.



Identified key Research Gaps

1. Standardized Benchmarking – Limited reporting of energy, memory, and latency metrics
2. Systematic Co-Design – Lack of formal methodologies linking models and hardware
3. Lifecycle Management – Focus on deployment rather than long-term maintenance
4. Security Integration – Security often treated as an afterthought
5. Toolchain Interoperability – Fragmented tools increase development complexity



Platform Type	Key Strength	Primary Limitation
ESP32-class	High compute and connectivity	High power consumption
STM32H7-class	High-performance MCU	Moderate energy efficiency
nRF52-class	Ultra-low power and wireless	Limited compute capability
Apollo3-class	Ultra-low power operation	Lower processing speed
RP2040-class	Low cost, flexible I/O	Weak power

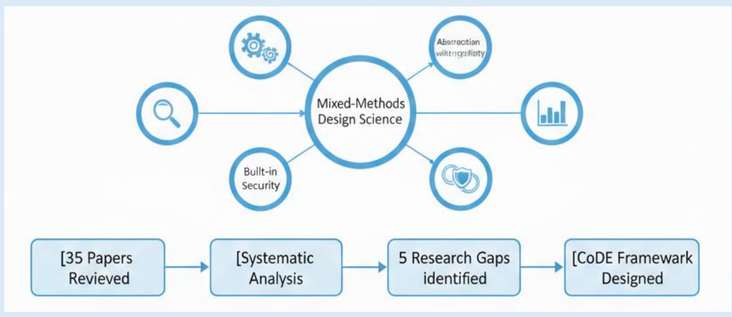
Methodology

This study follows a **design science approach** in three integrated phases:

Systematic Review – Analysis of 35 publications (2020–2025) on TinyML implementation challenges.

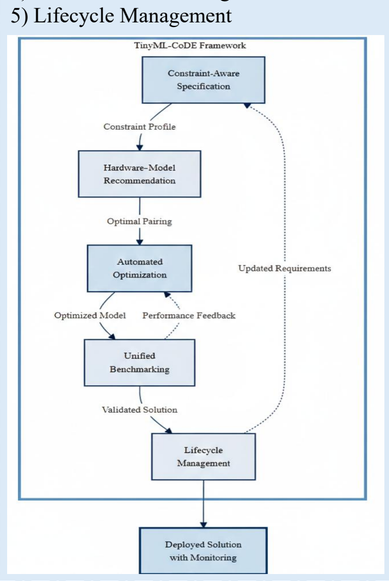
Framework Design – Iterative development of the TinyML-CoDE framework based on identified gaps.

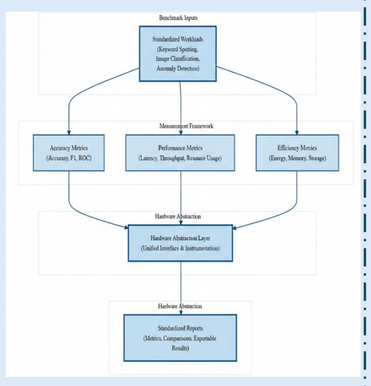
Validation Planning – Preparation for real-world case studies and comparative benchmarking.



TinyML-CoDE Framework

TinyML-CoDE is a structured co-design and evaluation framework that connects application requirements, hardware selection, model development, optimization, benchmarking, and lifecycle support.
Core components:
1) Constraint-Aware Specification
2) Hardware–Model Recommendation
3) Automated Optimization
4) Unified Benchmarking Suite
5) Lifecycle Management



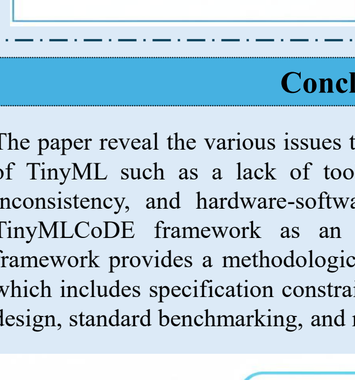


Application	Hardware	Model	Accuracy
Keyword spotting	ESP32-S3	DS-CNN 20k (int8)	94.2%
ECG classification	Apollo3 Blue	TinyLSTM (int16)	96.5%
Predictive maintenance	nRF52840	Random Forest	92.3%
Environmental sensing	Raspberry Pi Pico	Decision Tree	89.8%

Expected Benefits and Analysis

The TinyML-CoDE framework is expected to significantly improve the efficiency and reliability of TinyML development. By enabling systematic hardware–software co-design and unified evaluation, the framework reduces development complexity while improving performance and energy efficiency.

Metric	Projected Impact
Development time	~30% reduction
Performance efficiency	~25% improvement
Model accuracy	~5% improvement
Energy efficiency	~25% improvement
Deployment success rate	~20% increase



B. Validation Approach

Complmentary methods:
- Comparative analysis (benchnaking)
- Case studies (industrial monitoring, healthcare

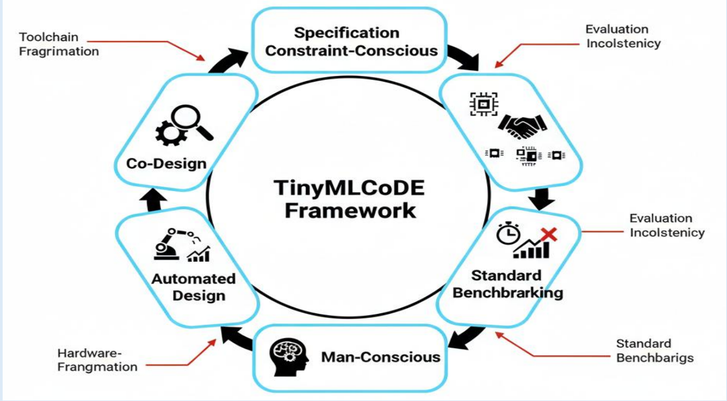
Measures of Success
- Decreased development time
- Improved performance
- Increased reliability

C. Implementation Challenges
- Hardware heterеereity
- Ecosystem adoption barriers
- Measurement consistencs

Mitication Strategies
- Open-source creation
- Hardware manufacturer callaboration
- Integration with existing TinyML ecosstems

Conclusions

The paper reveal the various issues that are inherent in the development of TinyML such as a lack of toolchain frag- mentation, evaluation inconsistency, and hardware-software mismatches and proposes the TinyMLCoDE framework as an answer to these problems. The framework provides a methodological design of TinyML development, which includes specification constraint-conscious, co-design, automated design, standard benchmarking, and man-conscious.



Future Work

The focus of future research will be on a step one step implementation plan and confirmation. Short term (0-12 months): This period will be focused on the development of prototypes of benchmarking and the creation of partnerships. with industry partners The emphasis will be on optimization improvements and introducing lightweight security features that are specifically relevant to resource-constrained settings in the medium term The work will focus on full MLOps integration, building the community-driven standardization, and creating automated and privacy-aware co-design methods in the long term.



References

[1] N. Schizas, A. Karras, C. Karras, and S. Sioutas, “TinyML for Ultra-Low
Power AI and Large Scale IoT Deployments: A Systematic Review,”
Future Internet, vol. 14, p. 363, 2022.
[2] N. Alajlan and D. Ibrahim, “TinyML: Enabling of Inference Deep Learn-
ing Models on Ultra-Low-Power IoT Edge Devices for AI Applications,”
Micromachines, vol. 13, no. 6, p. 851, 2022.
[3] R. Kallimani, K. Pai, P. Raghuwanshi, S. Iyer, and O. Lopez, “TinyML: Tools, applications, challenges, and future research directions,” Multi-media Tools and Applications, vol. 83, pp. 29015–29045, 2023.
[4] R. Sanchez-Iborra and A. Skarmeta, “TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities,” IEEE Circuits and Systems Magazine, vol. 20, pp. 4–18, 2020.
[5] F. Oliveira, D. Costa, F. Assis, and I. Silva, “Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning,” Internet of Things, vol. 26, p. 101153, 2024.
[6] M. Diab and E. Rodr´guez-Villegas, “Embedded Machine Learning Using Microcontrollers in Wearable and Ambulatory Systems for Health and Care Applications: A Review,” IEEE Access, vol. 10, pp. 98450–98474, 2022.
[7] J. Lin, L. Zhu, W. Chen, W. Wang, and S. Han, “Tiny Machine Learning:
Progress and Futures,” IEEE Circuits and Systems Magazine, vol. 23,