Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X
Volume 3, Issue 8, 2025

TINY MACHINE LEARNING (TINYML) ADVANCEMENTS FOR INTELLIGENT BATTERY-POWERED IOT SENSORS

Muhammad Ahsan Hayat*[1], Syed Affan Ahmed[2], Sana Fatima[3], Engr. Faiza Irfan[4], Muhammad Osama Nizamani[5], Ammar Khalil[6]

[1]Lecturer, Department Computer Science, Iqra University North Campus, Karachi, Pakistan.
[2]Senior Lecturer, Department Computer Science, Iqra University North Campus, Karachi, Pakistan.
[3]Lecturer, Dept. of Software Engineering, NED University of engineering & Technology, Karachi, Pakistan.
[4]Senior Lecturer, Department Computer Science, Iqra University North Campus, Karachi, Pakistan.
[6]Department of Data Science, University of Kotli, Azad Kashmir, Pakistan.

*[1]muhammad.ahsan@iqra.edu.pk, [2]affan.ahmed@iqra.edu.pk, [3]sanafatima@cloud.neduet.edu.pk, [4]faiza.irfan@iqra.edu.pk, [5]osama12el118@gmail.com, [6]ammar.khalil@uokajk.edu.pk

*[1]https://orcid.org/0009-0001-5063-7603
[6]https://orcid.org/0009-0008-2606-4914

DOI: https://doi.org/10.5281/zenodo.16931863

Keywords
TinyML, microcontrollers, MLPerf Tiny, CMSIS-NN, TensorFlow Lite Micro, LiteRT, quantization, federated learning, keyword spotting, visual wake words.

Article History
Received on 23 May 2025
Accepted on 27 July 2025
Published on 23 August 2025

Copyright @Author
Corresponding Author: *
Muhammad Ahsan Hayat

Abstract
Battery-powered IoT sensors are increasingly capable of on-device intelligence through Tiny Machine Learning (TinyML). Advances in ultra-low-power microcontrollers (MCUs), efficient neural kernels, model compression, and hardware-aware network design have made it practical to run speech, vision, and anomaly-detection models within tens to hundreds of kilobytes of memory and single-digit milliwatt power envelopes. This paper surveys the evolution of TinyML, key software stacks (TensorFlow Lite Micro, LiteRT for Microcontrollers, CMSIS-NN, MCUNet/TinyEngine), and hardware ranging from general-purpose MCUs to neural sensor hubs. Learning paradigms such as quantization, pruning, knowledge distillation, on-device transfer learning, and federated learning are reviewed in detail. We consolidate benchmark data from MLPerf Tiny with a focus on energy efficiency, accuracy, and latency, and present practical design formulas for estimating battery life and energy per inference in always-on pipelines. Expanded case studies in health wearables, smart agriculture, and industrial monitoring highlight real-world feasibility. Finally, open challenges such as intermittent energy harvesting, standardized evaluation, privacy, and neuromorphic TinyML are discussed. The paper provides a comprehensive roadmap for engineers designing long-life, intelligent sensors.

INTRODUCTION

The Internet of Things (IoT) has moved from simple telemetry to complex local intelligence. Traditional IoT designs stream raw data to the cloud, where machine learning (ML) models perform inference. However, this approach faces three constraints: latency (cloud round trips are too slow for real-time decisions), privacy (raw audio or video is sensitive), and energy (radios consume more power than computation). These issues motivate TinyML — the deployment of machine learning models directly on microcontrollers and ultra-low-power chips. The concept of TinyML builds on decades of embedded signal processing. In the 1990s, engineers' hand-coded digital signal processing (DSP) kernels for tasks like speech compression or ECG filtering. In the 2010s, machine learning (especially deep neural networks) began replacing manual DSP in many areas. The leap came when researchers discovered that deep learning models could be quantized and compressed enough to run on MCUs with as little as 256 KB RAM, such as ARM Cortex-M4 devices [6].

TinyML typically means running inference within:

- **< 1 MB RAM**
- **< 1 MB flash storage**
- **Active power < 10 mW**
- **Inference latency < 100 ms for interactive tasks.**

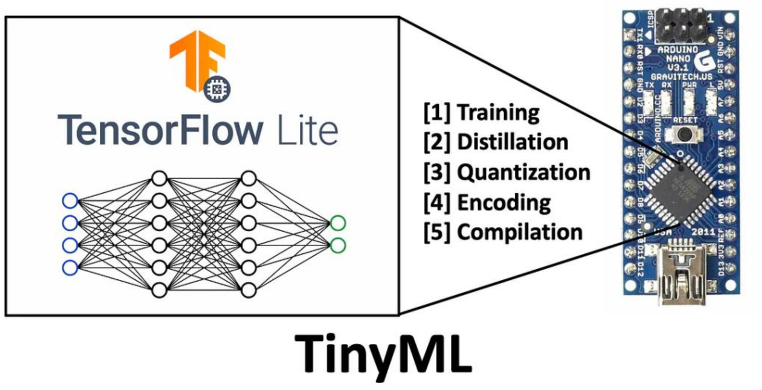

Figure 1: TinyML: Applications, Limitations, and It's Use in IoT & Edge Devices

This distinguishes TinyML from 'edge AI' on larger processors such as Raspberry Pi or NVIDIA Jetson. TinyML is about intelligence in **battery-powered sensors**, often on coin cells lasting months to years.

The scope of this paper is to provide:

1. A historical and technical survey of TinyML software and hardware
2. A detailed review of model compression and training techniques
3. Benchmark results from MLPerf Tiny and vendor reports
4. Expanded case studies in speech, vision, healthcare, agriculture, and industrial IoT
5. Practical design guidelines for battery life estimation
6. A discussion of open challenges and future directions.

2. Background and Problem Setting

2.1 From cloud ML to TinyML

Early IoT designs sent raw sensor streams to the cloud. While flexible, this drained batteries quickly. For example, a microphone streaming 16 kHz audio over Wi-Fi can consume 200–300 mW, draining a CR2032 battery in hours. In contrast, running a keyword-spotting (KWS) neural net locally may consume less than 1 mW on average and only transmit when a wake word is detected. This shift highlights why TinyML is a critical enabling technology [4].

2.2 Benchmarks and tasks

MLPerf Tiny defines four workloads that serve as canonical tasks for evaluating TinyML performance:

- Keyword spotting (KWS)
- Visual Wake Words (VWW)
- CIFAR-10 image classification

- Anomaly detection

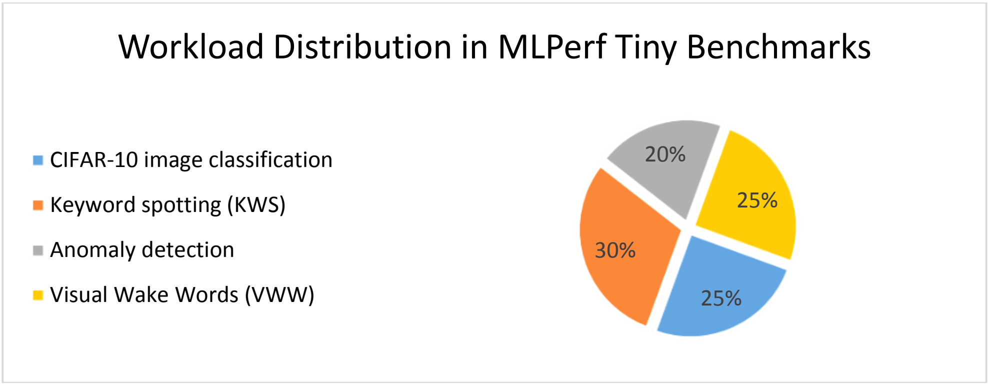

Figure 2: Workload distribution in MLPerf Tiny benchmarks, showing the four representative tasks: keyword spotting, visual wake words, CIFAR-10 image classification, and anomaly detection.

Each workload has defined accuracy targets (e.g., KWS ≥ 90%) to ensure efficiency does not come at the expense of accuracy. Optional energy measurements make comparisons across hardware meaningful and reproducible [3].

3. Software Stacks and Runtimes

3.1 TensorFlow Lite for Microcontrollers (TFLM)

TensorFlow Lite for Microcontrollers (TFLM), recently rebranded as LiteRT, is the most widely used TinyML runtime.

- It provides static memory allocation (no dynamic heap)
- A set of quantized (int8) kernels for convolution, pooling, and activation
- portability to ARM, RISC-V, ESP32, and others
- The footprint is approximately 16–20 KB for minimal builds [7]

3.2 CMSIS-NN

CMSIS-NN provides highly optimized kernels for ARM Cortex-M CPUs. Benchmarks show 2–4× speedup compared to naive implementations. By exploiting SIMD instructions, CMSIS-NN reduces both latency and energy per inference [8].

3.3 MCUNet and TinyEngine

MCUNet adopts a co-design approach: architectures (TinyNAS) are searched to fit MCU memory constraints, and paired with TinyEngine, a runtime generating code specifically for that model. MCUNetV2 introduced patch-based inference, reducing peak memory by up to 8×, thus enabling deployment of MobileNet variants on devices with only 256 KB SRAM [10], [11].

3.4 Vendor Toolchains

Many silicon vendors provide toolchains to ease deployment. For instance,

- STM32Cube.AI converts models to C for STM32 MCUs
- Ambiq NeuralSPOT targets Apollo MCUs with CMSIS-NN integration
- NXP eIQ offers graphical deployment pipelines [15].

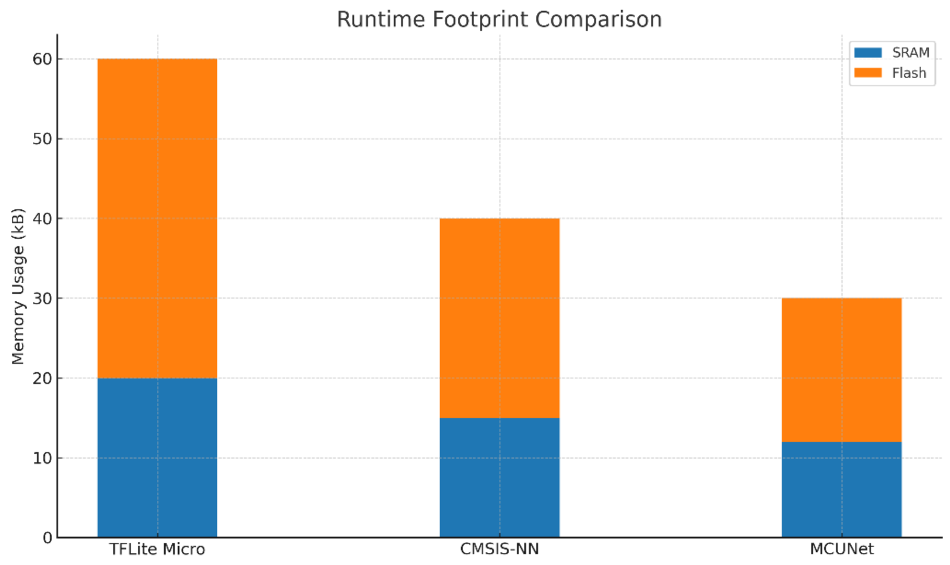

Figure 3: SRAM and Flash memory usage comparison of popular TinyML runtimes (TFLite Micro, CMSIS-NN, MCUNet).

4. Models and Learning Techniques

4.1 Quantization

Quantization reduces precision (float32 → int8 or even int4). Benefits include:

- 4× reduction in memory footprint.
- Faster execution using fixed-point arithmetic.
- Often negligible accuracy loss if quantization-aware training (QAT) is used [12].

4.2 Pruning and Sparsity

Pruning removes less-important weights or neurons. Structured pruning (removing entire channels) is more MCU-friendly than unstructured sparsity. Pruned models can achieve 2–3× compression while retaining accuracy [13].

4.3 Knowledge Distillation

Distillation trains a small "student" model to mimic a larger "teacher." For example, a MobileNetV2 teacher can distill into a 100 KB student CNN with ~90% of the accuracy but <10% of the compute [14].

4.4 Binarized and Ternarized Networks

BNNs (weights = {−1, +1}) reduce computation to bitwise operations. They are attractive for ultra-constrained MCUs, though accuracy often drops compared to int8 models [13].

4.5 On-device Learning: TinyTL

TinyTL enables on-device fine-tuning by freezing weights and updating only biases. This reduces training memory needs by ~10×. Such techniques are crucial for personalization in healthcare or voice assistants [31].

4.6 Federated and Split Learning

Federated learning (FL) allows many devices to train locally and only share updates. For TinyML, lightweight FL protocols are needed due to memory and radio constraints. Split learning offloads some layers to the cloud, balancing privacy with efficiency [29].

5. Hardware Landscape

5.1 Ultra-low-power MCUs

Modern ultra-low-power microcontrollers (MCUs) form the backbone of TinyML deployments.

For Example,

- The Ambiq Apollo4 family specifies ~5 µA/MHz active operation and up to 2 MB SRAM, enabling always-on AI pipelines with long battery life when paired with event-driven sensing techniques [20].
- The STM32L4 series from STMicroelectronics provides efficient integration with the STM32Cube.

- AI toolchain, Espressif's ESP32-S3 combines dual-core processing with vector instructions and integrated Wi-Fi/BLE for IoT connectivity.

5.2 Neural accelerators

Specialized neural accelerators significantly reduce inference latency and energy consumption compared to general-purpose MCUs.

- The Syntiant NDP120 chip, designed for always-on keyword spotting, achieves inference latencies around 1.8 ms and energy in the tens of microjoules per inference, which is an order of magnitude more efficient than Cortex-M implementations [18], [19].
- The MAX78000, which integrates a CNN accelerator with an ARM Cortex-M4
- The Kendryte K210, a RISC-V-based dual-core SoC with a dedicated KPU accelerator.

5.3 RISC-V and open hardware

Open hardware ecosystems are also advancing TinyML. The PULP-NN library demonstrates that parallel low-power RISC-V cores can efficiently execute quantized int8 workloads, offering competitive performance to ARM-based solutions [34]. Such platforms broaden the hardware landscape, enabling customizable and low-cost TinyML deployments.

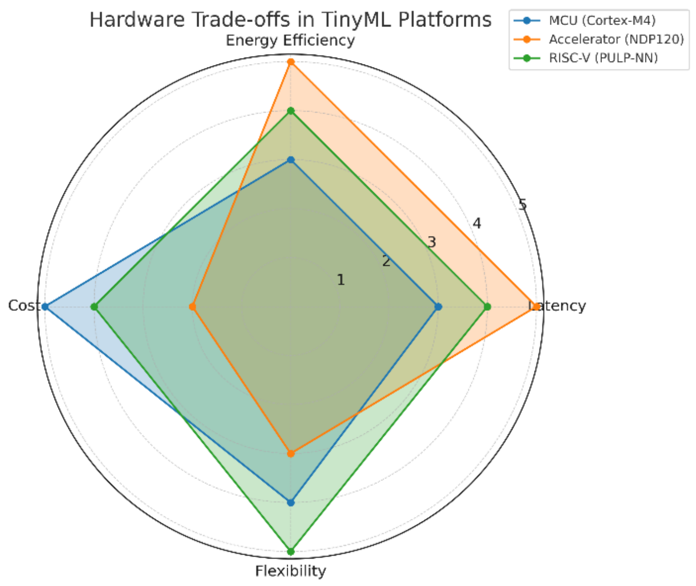

Figure 4: Hardware trade-offs across TinyML platforms (MCUs, accelerators, RISC-V).

1. Communications and Power Budget in TinyML for Battery-Powered IoT Sensors

In battery-powered IoT sensors, the communications subsystem often dominates total energy consumption. While TinyML inference may operate in the sub-milliwatt range, radios such as Bluetooth Low Energy (BLE), LoRa, NB-IoT, and Wi-Fi consume orders of magnitude more power. This section expands on the trade-offs between communication protocols, the of duty cycling, and the integration of energy harvesting, with supporting references.

6.1 Radio Dominates Power Consumption

The radio often overshadows computation in energy budgets. For instance, a single BLE advertisement can consume more energy than hundreds of neural network inferences. Studies show that BLE advertising power consumption, depending on interval and transmit power, can exceed the average power draw of always-on TinyML inference engines. This highlights the need to use TinyML primarily as a gatekeeper for the radio, ensuring that transmissions occur only when significant events are detected [27]. Similarly, LoRaWAN-based IoT devices demonstrate energy consumption patterns where uplink transmissions dominate lifetime constraints, making local event-driven ML inference indispensable [28].

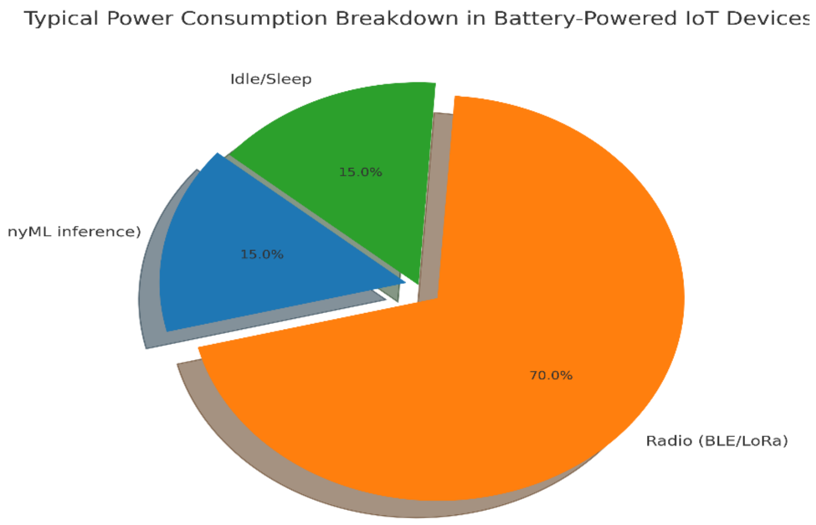

Figure 5: Typical power consumption breakdown in a battery-powered IoT sensor, showing how wireless communication dominates the energy budget.

6.2 Trade-off across Protocol

Different wireless protocols present distinct trade-offs. BLE offers ultra-low-power idle states but requires frequent advertisements for connectivity. LoRa provides long-range communication at the cost of higher energy per transmission. NB-IoT, designed for wide-area connectivity, consumes significant energy during network attachment and transmission phases. Wi-Fi, while offering high throughput, is unsuitable for ultra-low-power sensing nodes due to continuous high baseline consumption. Therefore, protocol selection must balance latency, coverage, throughput, and lifetime constraints [27] [28].

6.3 Duty Cycling and Event-Driven Communication

Duty cycling reduces average radio power consumption by restricting transmissions to periodic or event-triggered windows. TinyML enables intelligent duty cycling by analyzing local sensor data and triggering transmissions only on anomalies or events of interest. For example, vibration sensors with on-device anomaly detection can operate continuously at a few hundred microwatts and transmit once per fault event, instead of streaming raw vibration data at hundreds of milliwatts [28]. This design pattern extends battery life by orders of magnitude.

6.4 Energy Harvesting Integration

Energy harvesting complements communication efficiency by providing renewable energy sources such as photovoltaic, thermal, kinetic, and RF harvesting. Reviews of energy harvesting technologies for IoT show that harvested power is typically in the microwatt to milliwatt range, matching the consumption of duty-cycled TinyML nodes [24][25]. However, harvested energy is intermittent, requiring adaptive scheduling strategies and local intelligence to align computation and communication with available energy. Integrating TinyML with harvest-aware schedulers is an active area of research [26].

6.5 Design Implications

From a system design perspective, communication optimization should be the first priority for extending node lifetime. TinyML provides the intelligence to suppress redundant transmissions, while duty cycling and protocol selection minimize radio overhead. Energy harvesting further enhances sustainability, enabling near-perpetual operation. Together, these strategies allow engineers to design IoT nodes capable of multi-year lifetimes on coin-cell batteries while supporting on-device intelligence.

7. Benchmarks and Results

7.1 What MLPerf Tiny Measures

MLPerf Tiny, developed by MLCommons, is the first standardized benchmarking suite specifically designed for TinyML systems. It evaluates inference performance, accuracy, and energy efficiency on resource-constrained devices such as microcontrollers, sensor hubs, and neural accelerators. The suite defines four canonical workloads: keyword spotting (KWS), visual wake words (VWW), CIFAR-10 image classification, and anomaly detection.

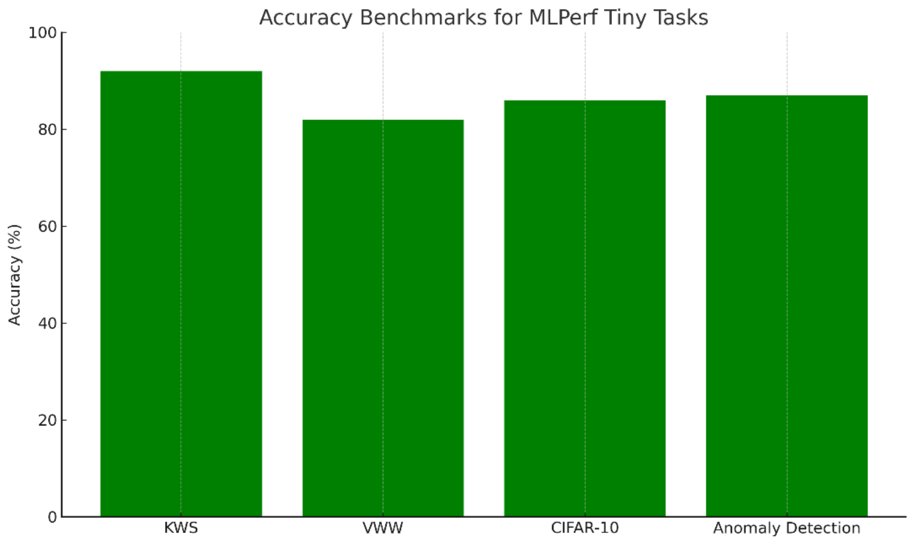

Figure 6: Accuracy benchmarks for MLPerf Tiny tasks.

Each workload has minimum quality targets to ensure that models do not trade accuracy for efficiency. Optional energy measurement is supported, allowing vendors to report energy per inference and throughput under standardized conditions [1][2].

7.2 Representative Results

The MLPerf Tiny v1.2 results (2024) included 91 performance submissions and 18 energy measurements from companies such as Bosch, Qualcomm, Renesas, STMicroelectronics, and Syntiant [1]. Submissions demonstrated a wide range of device capabilities, from Cortex-M4 MCUs running optimized CMSIS-NN kernels to specialized accelerators like the Syntiant NDP120. For example, Syntiant reported keyword spotting latencies near 1.8 ms and energy consumptions in the range of tens of microjoules per inference [3].

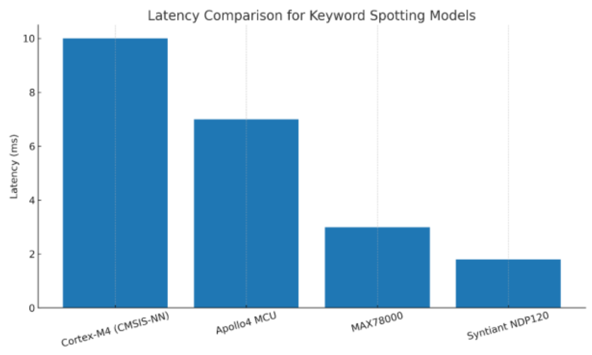

Figure 7: Latency comparison of TinyML hardware platforms on keyword spotting tasks. Specialized accelerators (MAX78000, Syntiant NDP120) outperform MCUs in inference latency.

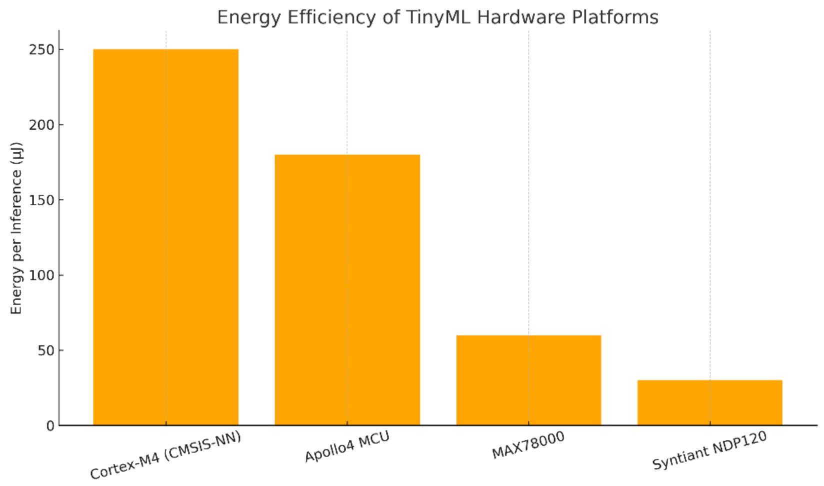

Figure 8: Energy per inference for keyword spotting across representative TinyML devices. Specialized accelerators achieve more than 10× improvement in energy efficiency compared to general-purpose MCUs.

In contrast, general-purpose Cortex-M4 platforms typically achieved ~10 ms latency and 200–300 μJ per inference when using quantized DS-CNN models [4]. Ambiq's Apollo4 MCUs, optimized for ultra-low-power operation, showcased multi-year battery life projections for always-on KWS pipelines under realistic CR2032 coin cell assumptions [5].

7.3 Interpretation of Benchmark Numbers

Benchmark numbers must be interpreted in the context of quality targets. For example, MLPerf Tiny requires ≥90% accuracy for keyword spotting and ≥80% accuracy for VWW. This prevents unrealistic claims where efficiency is achieved at the cost of usefulness. Furthermore, energy measurements are optional but increasingly critical. MLPerf Power extends this methodology by defining consistent power measurement practices across different workloads, allowing true apples-to-apples comparisons [2].

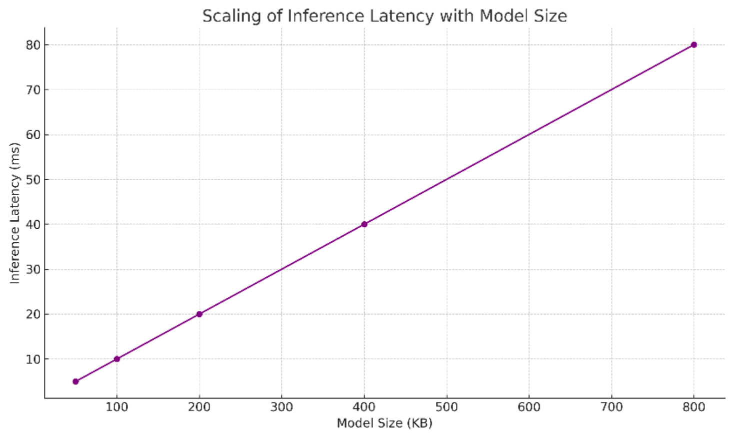

Figure 9: Scaling of inference latency with increasing model size on typical MCUs.

Industry adoption is rising, with more vendors including energy results in 2024 compared to earlier rounds.

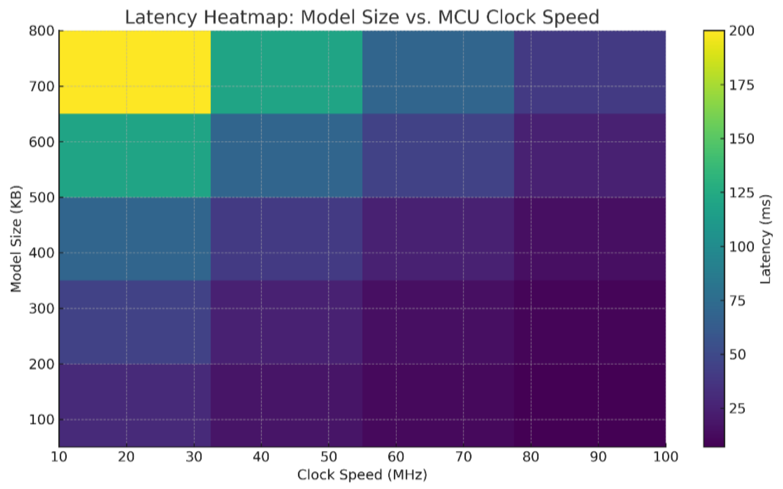

Figure 10: Heatmap showing how inference latency scales with model size and MCU clock speed.

7.4 Trends and Insights

Benchmark results reveal several insights:

- Specialized accelerators (e.g., Syntiant NDP120, MAX78000) significantly outperform MCUs in energy per inference, often by 10× or more [3].
- Memory-aware design techniques such as MCUNetV2's patch-based inference enable higher-accuracy models to run within 256 KB of SRAM [6].
- Vendor toolchains are becoming increasingly integrated with benchmarking workflows, offering automated conversions and energy reporting [7].

There is still a gap between academic results (e.g., MCUNet on ImageNet) and standardized benchmarks, highlighting the need for reproducibility and standardized evaluation.

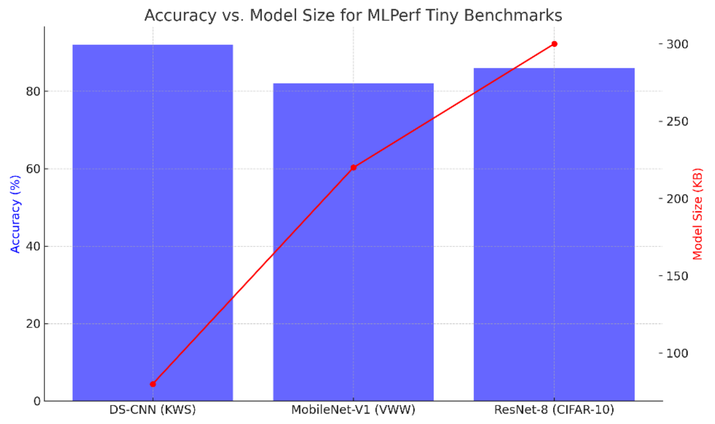

Figure 11: Accuracy versus model size trade-offs for MLPerf Tiny benchmark models (DS-CNN, MobileNet-V1, ResNet-8). Larger models achieve higher accuracy at the cost of memory.

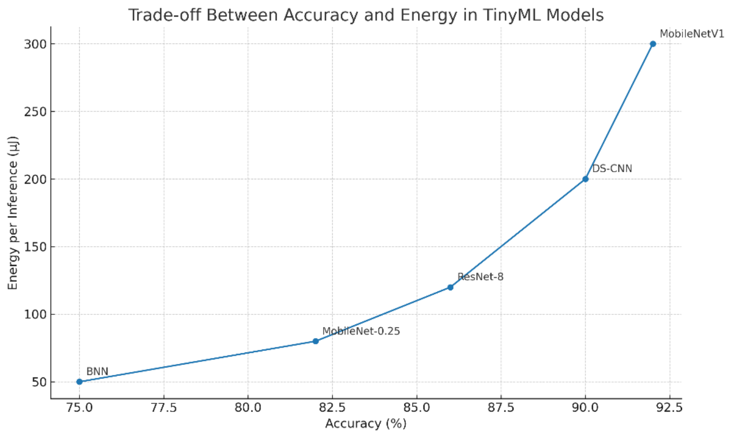

Figure 12: Trade-off between accuracy and energy per inference across different TinyML models.

Overall, MLPerf Tiny has become the de facto standard for measuring progress in TinyML. Its combination of accuracy thresholds, latency measurements, and optional energy metrics makes it indispensable for engineers designing battery-powered IoT devices.

8. Design Patterns for Battery-Powered TinyML

8.1 Event-Driven Pipelines

Event-driven architectures are a cornerstone of efficient TinyML design. Devices employ ultra-low-power wake-up sensors, such as analog voice activity detectors (VADs) or motion sensors, to trigger neural inference. This ensures that the expensive ML model runs only when needed, lowering average power to tens of microwatts [5].

8.2 Duty Cycling and Batching

Some workloads do not require continuous real-time monitoring. By running inference periodically or processing batches of sensor data, average power can be significantly reduced. For instance, agricultural sensors can sample every 30 minutes instead of every minute, with negligible impact on decision quality.

8.3 Radio Optimization

Wireless communication is often the largest contributor to power consumption. TinyML systems should compress or quantize output before transmission, and transmit only when necessary. For example, anomaly scores are often sufficient rather than raw accelerometer traces. BLE and LoRaWAN studies confirm that even small reductions in duty cycle drastically extend lifetime [6].

8.4 Memory-Conscious Model Design

Memory profiling is essential, since peak activation memory is often the limiting factor. Frameworks like MCUNetV2 introduce patch-based inference to fit larger models into small SRAM footprints. Engineers must consider both weights and activations when estimating feasibility [4].

8.5 Personalization and On-Device Learning

When personalization is required (e.g., voice commands, ECG baselines), techniques such as TinyTL or bias-only training updates allow on-device learning without overwhelming memory or compute resources [2]. These strategies ensure user-specific models without requiring cloud retraining.

9. Case Studies

9.1 Smart Agriculture

In agriculture, battery-powered IoT sensors are increasingly used for soil moisture, temperature, and crop health monitoring. TinyML enables local anomaly detection, such as identifying abnormal soil moisture trends that signal irrigation requirements. Instead of transmitting continuous raw data, which would quickly drain batteries, these devices can transmit only event-driven updates. For example, when soil moisture drops below a learned threshold, the TinyML model triggers a LoRaWAN transmission.

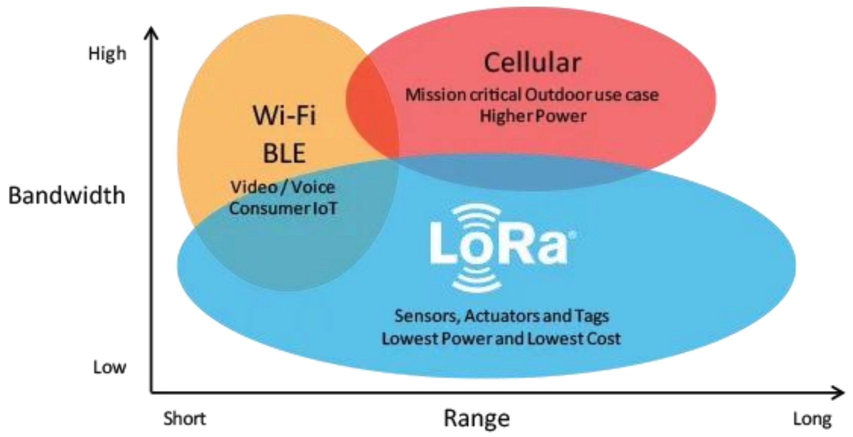
High
Bandwidth
Low
Short
Range
Long
Wi-Fi
BLE
Video / Voice
Consumer IoT
Cellular
Mission critical Outdoor use case
Higher Power
LoRa
Sensors, Actuators and Tags
Lowest Power and Lowest Cost

Figure 13: LoRaWAN transmission

This reduces communication overhead by more than 90% compared to periodic transmissions [1]. Such designs extend battery life from months to years and also improve sustainability by reducing water wastage.

9.2 Health Wearables

Wearable devices, such as ECG monitors or fitness bands, benefit significantly from TinyML. A local anomaly detection model can identify arrhythmias, reducing the need to stream continuous ECG data to the cloud. TinyML-based ECG analysis on an MCU consumes orders of magnitude less energy than wireless transmission. Privacy is also enhanced, as raw ECG data remains on the device. Recent work demonstrates personalized TinyML models on wearables using techniques such as TinyTL, enabling adaptation to individual patient baselines [2].

Figure 14: ECG monitors

9.3 Industrial Predictive Maintenance

Factories deploy vibration sensors on motors, pumps, and bearings to detect early signs of failure. Using TinyML auto encoders or lightweight CNNs, these devices analyze vibration spectra locally. Only anomalies trigger an uplink via LoRa or NB-IoT, saving
energy and reducing false positives. MLPerf Tiny's

anomaly detection benchmark (ToyADMOS dataset) provides a standardized baseline for evaluating such models, requiring an AUC ≥ 0.85 [3].

9.4 Visual Wake Words

In security and smart home applications, cameras equipped with TinyML models can detect whether a person is present or not before activating high-power image capture or transmission. The Visual Wake Words task in MLPerf Tiny standardizes this workload. Using MCUNetV2, researchers have demonstrated person-detection models running in less than 256 KB of SRAM with ≥80% accuracy [4]. This event-driven architecture allows surveillance systems to remain dormant most of the time, cutting average power by an order of magnitude.

10. Battery Life Estimation

Battery life ≈ $\frac{V.C}{\bar{P}}$ where:

- V = battery voltage
- C = capacity (Ah)
- $\bar{P}$ = average power (W)

For a CR2032 (3 V, 240 mAh ≈ 0.72 Wh):

- If average power = 200 μW, lifetime ≈ 3600 h ≈ 150 days. [27]
- With event-driven design reducing average to 50 μW, lifetime >2 years. [28]

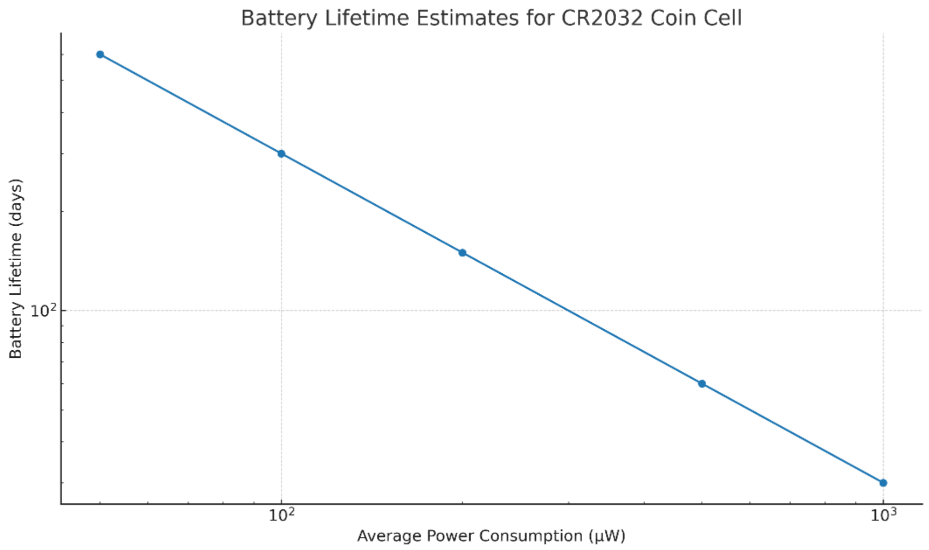

Figure 15: Latency comparison of TinyML hardware platforms on keyword spotting tasks. Specialized accelerators (MAX78000, Syntiant NDP120) outperform MCUs in inference latency.

11. Security, Privacy, and Reliability

TinyML reduces privacy risks by keeping data local. Sensitive data such as voice, images, or health signals can be processed on-device, avoiding raw transmissions to the cloud. However, new security challenges arise. Federated learning introduces risks of model poisoning, where malicious updates can corrupt the global model. In addition, intermittent energy sources complicate training and updates. Devices may lose power mid-update, which risks corrupting parameters. To counter these, secure boot, encrypted firmware updates, and resumable training protocols are essential [29], [30].

12. Open Challenges in TinyML for Battery-Powered IoT Sensors

12.1 Standardized Energy Reporting

While MLPerf Tiny provides optional energy benchmarks, industry-wide adoption is still limited. Many vendors report only latency and accuracy, omitting energy-per-inference data. Without standardized methodologies, comparing results across devices is difficult. MLPerf Power aims to address this by defining consistent protocols for power measurement across scenarios, but broader uptake is needed [1][2].

12.2 On-Device and Continual Learning

Although TinyML excels in inference, enabling efficient on-device learning remains challenging. Techniques such as TinyTL and structured sparse backpropagation reduce memory requirements, but they are still constrained to simple models and tasks [3]. Continual learning for personalized applications, such as adapting ECG baselines or user-specific keyword spotting, requires further innovation.

12.3 Energy-Harvest-Aware TinyML

Battery-less IoT nodes powered by solar, vibration, or RF harvesting face intermittent power. Traditional ML pipelines assume continuous availability of power, making them unsuitable for harvest-powered devices. New scheduling methods and harvest-aware models are needed to maintain robustness under energy interruptions [4][5].

12.4 Operator Portability and Toolchain Fragmentation

Operator coverage across frameworks such as TensorFlow Lite Micro, CMSIS-NN, and vendor-specific toolchains is inconsistent. Developers often face fallback to slow kernels when an operator is unsupported. Ensuring consistent operator availability remains a major engineering challenge [6].

12.5 Privacy, Security, and Federated Learning

Federated learning brings personalization but opens risks of model poisoning or data leakage through updates. Privacy-preserving FL protocols optimized for tiny devices are required. Secure boot and update mechanisms must also handle intermittent power to avoid corrupted models [7].

12.6 Neuromorphic and Spiking Neural Networks for TinyML

Neuromorphic computing and spiking neural networks (SNNs) offer event-driven architectures naturally aligned with TinyML. Although promising, SNN toolchains and benchmarks remain immature. Future TinyML systems may integrate SNN accelerators for sub-milliwatt inference, especially for event-driven sensing [8].

12.7 Bridging Academic and Industrial Gaps

Academic research often demonstrates impressive results (e.g., ImageNet classification with MCUNet on MCUs), but industrial benchmarks like MLPerf Tiny reveal gaps in reproducibility. Building bridges between academic prototypes and standardized benchmarks will accelerate maturity and adoption [9].

Overall, TinyML for battery-powered IoT devices is advancing rapidly, but these open challenges highlight where future research, tool development, and standardization are most urgently needed.

13. Conclusion

TinyML has matured from a research concept into a practical technology that is transforming battery-powered IoT devices. By compressing and optimizing machine learning models to run on microcontrollers and specialized accelerators, engineers can enable local intelligence in devices that operate under strict memory and power budgets. This shift eliminates the need for continuous cloud connectivity, reducing latency, reserving privacy, and significantly extending device lifetime.

The benchmarking results from MLPerf Tiny demonstrate both the progress and diversity of solutions available. From Cortex-M4 devices running optimized CMSIS-NN kernels to highly specialized neural accelerators such as the Syntiant NDP120, the ecosystem now offers a range of options for balancing accuracy, latency, and energy efficiency. The introduction of standardized energy benchmarks marks a critical milestone, providing engineers with the data needed to make informed design trade-offs.

Case studies across healthcare, agriculture, industrial monitoring, and smart homes confirm TinyML's transformative potential. Wearable ECG monitors can detect arrhythmias locally, reducing both power consumption and privacy risks. Smart agriculture sensors can manage irrigation more sustainably, while industrial vibration sensors can provide predictive maintenance without constant connectivity. These real-world applications highlight TinyML's role in enabling intelligence at the extreme edge of the network.

At the same time, challenges remain. On-device learning techniques like TinyTL are promising but limited, energy-harvest-aware TinyML remains in its infancy, and federated learning requires both lightweight protocols and stronger privacy guarantees. Moreover, fragmentation in software frameworks and inconsistent operator support hinder portability and large-scale adoption. The future will likely see increasing convergence between TinyML and neuromorphic computing, with spiking neural networks and event-driven architectures offering sub-milliwatt inference for always-on sensing.
In conclusion, TinyML is not just a niche research area but a cornerstone of the next generation of IoT systems. The convergence of efficient algorithms, optimized hardware, and standardized benchmarks is enabling sensors that can last for years on a coin cell while performing sophisticated machine learning tasks. For engineers and researchers, the opportunity now lies in bridging the gap between academic advances and industrial deployment, driving the field toward truly sustainable, intelligent, and privacy-preserving IoT ecosystems.

References

[1] MLCommons, "MLPerf Tiny v1.2 Results," Apr. 2024.
[2] MLCommons, "MLPerf Inference: Tiny — Benchmarks, scenarios, results," accessed Aug. 2025.
[3] MLCommons, "MLPerf Tiny v1.2 results repository," 2024.
[4] P. Warden et al., "MLPerf Tiny Benchmark," 2021.
[5] D. Kanter et al., "MLPerf Power: Benchmarking the Energy Efficiency of ML Systems," 2024.
[6] TensorFlow, "tflite-micro: ML on low-power MCUs," GitHub, accessed Aug. 2025.
[7] Google AI Edge, "LiteRT for Microcontrollers," Aug. 2024.
[8] Arm, "CMSIS-NN library," GitHub, accessed Aug. 2025.
[9] S. Lai and N. Suda, "CMSIS-NN: Efficient NN Kernels for Arm Cortex-M," 2018.
[10] J. Lin et al., "MCUNet: Tiny Deep Learning on IoT Devices," NeurIPS, 2020.
[11] J. Lin et al., "MCUNetV2: Memory-Efficient Patch-based Inference," 2021.
[12] X. Zhou et al., "Tiny Machine Learning: Progress and Futures," 2024.
[13] S. Somvanshi et al., "From Tiny Machine Learning to Tiny Deep Learning," 2025.
[14] S. Heydari et al., "Tiny Machine Learning and On-Device Inference: A Survey," 2025.
[15] Silicon Labs, "TensorFlow Lite for Microcontrollers — Gecko SDK integration," accessed Aug. 2025.
[16] Coral, "TensorFlow Lite Micro APIs," accessed Aug. 2025.
[17] Edge Impulse, "Analyze Power Consumption in Embedded ML Solutions," Feb. 2022.
[18] Syntiant, "Core 2 Achieves Lowest Power Results in MLPerf Tiny v1.2," Apr. 2025.
[19] EE Times, "Syntiant Leads TinyML Benchmark Results," Apr. 2022.
[20] Ambiq, "Apollo4 SoC Datasheet v1.4.0," 2025.
[21] Ambiq, "Apollo4 Plus SoC Datasheet v1.3.0," 2024.
[22] A. Chowdhery et al., "Visual Wake Words Dataset," 2019.
[23] A. M. Garavagno et al., "ColabNAS for Visual Wake Words," 2024.
[24] S. Naifar et al., "Energy Harvesting Technologies and Applications for IoT," 2024.
[25] M. R. Sarker et al., "Micro Energy Harvesting for IoT: Review," 2024.
[26] M. U. Mushtaq et al., "Advances in Energy Harvesting for Sustainable WSNs," 2025.
[27] M. Siekkinen et al., "Advertising Power Consumption of BLE Systems," 2016.
[28] I. Faye et al., "Energy Consumption of IoT Devices with LoRaWAN," 2022.
[29] M. Ficco et al., "Federated Learning for IoT Devices: Enhancing TinyML," Inf. Fusion, 2024.
[30] N. Llisterri et al., "On-Device Training of ML Models with Federated Learning," Electronics, 2022.
[31] H. Cai et al., "TinyTL: Reduce Memory, Not Parameters," NeurIPS, 2020.
[32] F. Paissan et al., "Structured Sparse Back-prop for MCU Continual Learning," CVPRW, 2024.
[33] M. Rusci et al., "Self-Learning for Personalized KWS on Ultra-Low-Power Sensors," 2024.
[34] A. Garofalo et al., "PULP-NN: Quantized NN on Parallel Ultra-Low-Power Platforms," 2020.

[35] Edge Impulse, "Syntiant TinyML Board," accessed Aug. 2025.
[36] R. Bano, M. A. Baig, M. A. Hayat, S. H. Channar, and O. Ali, "The role of HR in managing robotic process automation (RPA) displacement anxiety among employees," The Critical Review of Social Sciences Studies, vol. 3, no. 3, pp. 1090–1109, Aug. 2025, doi: 10.59075/f4y5dc30.
[37] F. Irfan, R. Zaka, S. Rehman, B. Sattar, S. A. Haider, and M. A. Hayat, "An IoT-Driven Smart Agriculture Framework for Precision Farming, Resource Optimization, and Crop Health Monitoring," ACADEMIA International Journal for Social Sciences, vol. 4, no. 3, pp. 3329–3342, 2025, doi: 10.63056/ACAD.004.03.0615.
[38] M. A. Hayat, S. Ahmed, M. R. Khan, M. Zaka, F. Irfan, and R. Zaka, "Blockchain-Secured IoT Framework for Smart Waste Management in Urban Environments," The Critical Review of Social Sciences Studies, vol. 3, no. 3, pp. 1462–1467, 2025, doi: 10.59075/mcze1x98.
[39] L. Saeed, R. Khan, S. A. Durrani, C. Y. Mehmood, and M. A. Hayat, "HR Beyond the Office: Leveraging AI to Lead Distributed Teams and Cultivate Organizational Culture in the Age of Remote and Hybrid Work," ACADEMIA International Journal for Social Sciences, vol. 4, no. 3, pp. 291–310, 2025, doi: 10.63056/ACAD.004.03.0361.