

TINYML: BRINGING MACHINE LEARNING TO RESOURCE-CONSTRAINED EMBEDDED SYSTEMS

Jeremy Ponseelan J

Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India

Abstract -

Tiny machine learning is an emerging approach that enables machine learning models to operate directly on resource-constrained embedded devices. This review aims to examine the principles, technologies, applications, challenges and future directions of machine learning on low-power embedded systems. A structured review of existing research was conducted by examining developments in embedded hardware, software frameworks, model optimisation techniques and application areas. Particular attention was given to quantisation, pruning, knowledge distillation and lightweight neural network architectures, which reduce memory requirements, computational complexity and energy consumption while maintaining useful prediction accuracy. The review also examines applications in healthcare, smart agriculture, environmental monitoring, industrial systems and wearable devices. The findings indicate that local machine learning inference can reduce communication requirements and processing delays while enabling intelligent decision-making on resource-constrained devices. However, limitations in memory, computational capability, energy availability, model accuracy and security continue to present significant challenges. Emerging approaches such as on-device learning, federated learning, energy harvesting and hardware–software co-design provide promising opportunities for addressing these limitations. Overall, the review demonstrates that machine learning on resource-constrained embedded systems has significant potential to transform conventional sensing devices into intelligent, low-power systems capable of real-time local decision-making.

Key Words: *Tiny machine learning; Embedded systems; Edge artificial intelligence; Machine learning; Microcontrollers; Internet of Things; Low-power computing.*

1. INTRODUCTION

The rapid growth of the Internet of Things (IoT) has resulted in billions of connected devices capable of sensing, processing and communicating information. Traditional IoT architectures

commonly collect sensor data at an edge device and transmit it to a gateway or cloud server for processing. Although this architecture provides substantial computational resources, it can introduce latency, communication overhead, bandwidth requirements and privacy concerns. The increasing demand for real-time intelligent systems has therefore encouraged researchers to move machine learning capabilities closer to the data source.

Tiny Machine Learning, commonly referred to as TinyML, represents an important development in this direction. TinyML focuses on deploying machine learning inference on highly resource-constrained devices, particularly microcontroller units (MCUs). These devices typically have substantially less computational power, memory and energy availability than conventional computers and smartphones. Consequently, machine learning models must be designed and optimised specifically for such environments.

Recent literature describes TinyML as an interdisciplinary field combining machine learning, embedded systems, edge computing, IoT, hardware architecture and energy-efficient computing. A recent bibliometric study analysed 392 peer-reviewed TinyML publications from 2020 to 2024 and identified strong growth in the field. Electrical and Electronics Engineering represented the largest subject category in that dataset, demonstrating the relevance of TinyML to ECE research.

The fundamental principle behind TinyML is simple: instead of transmitting all sensed information to a remote server, an embedded device can process the data locally and make decisions on the device itself. For example, a microphone connected to a microcontroller could identify a particular sound without continuously sending raw audio to the cloud. Similarly, an environmental sensor could locally identify abnormal conditions and transmit only the relevant result.

This architecture can provide several advantages. Local inference can reduce latency because data does not need to travel to a remote server. It can also reduce communication requirements and potentially improve privacy because sensitive raw data can remain on the device. However, these benefits are achieved under severe hardware constraints.

The objective of this review is to examine the development of TinyML and its significance to embedded systems. The

review discusses the hardware and software requirements, model optimisation methods, deployment process, applications, challenges and future research directions.

2 Fundamentals of TinyML

2.1 Definition and Concept

TinyML can be considered the deployment of machine learning capabilities on devices with very limited computational, memory and energy resources. Unlike conventional AI systems that may execute large models on GPUs or cloud servers, TinyML systems generally operate using microcontrollers and other low-power embedded platforms.

The basic TinyML architecture consists of four major components:

- [1] **Sensors** – acquire information from the physical environment.
- [2] **Microcontroller** – processes sensor information and executes the ML model.
- [3] **Machine learning model** – performs classification, regression, detection or another inference task.
- [4] **Communication interface** – optionally transmits decisions or selected information to another device or cloud platform.

The process can be represented as:

Physical environment → Sensor → Signal preprocessing → TinyML model → Decision → Optional communication

The important distinction is that inference takes place locally rather than requiring continuous cloud computation.

Recent surveys describe TinyML as particularly suitable for applications requiring low latency, reduced bandwidth, privacy and energy efficiency.

2.2 Why TinyML is Important

Traditional cloud-based machine learning requires sensor data to be transmitted to a remote processing platform. This may not be suitable for applications where immediate decisions are required.

For example, an industrial sensor detecting abnormal machine vibration may need to respond within milliseconds. Sending every vibration sample to a cloud server introduces communication delays and consumes additional energy.

TinyML can instead perform preliminary analysis directly on the embedded device. Only important events, classifications or compressed information may subsequently be transmitted.

The principal advantages are:

- [1] Lower inference latency
- [2] Reduced communication bandwidth
- [3] Reduced dependence on cloud connectivity
- [4] Potentially improved privacy
- [5] Lower communication energy consumption
- [6] Local and continuous operation
- [7] Suitability for battery-powered devices

These advantages make TinyML particularly attractive for IoT, wearable electronics, industrial monitoring and environmental sensing.

3 TinyML Hardware

3.1 Microcontroller Units

Microcontrollers form the foundation of many TinyML systems. Unlike high-performance processors, MCUs are designed for low-cost and low-power embedded applications.

Common MCU families used in TinyML include ARM Cortex-M based devices and various commercial development platforms. Recent literature indicates that ARM Cortex-M devices and Arduino-class platforms are frequently used for embedded machine learning research.

The hardware must provide sufficient:

- [8] Flash memory for the model and program
- [9] SRAM for intermediate data and activation tensors
- [10] Processing capability
- [11] Sensor interfaces
- [12] Power efficiency
- [13] Communication interfaces

TinyML models must therefore be designed with the target hardware in mind.

3.2 Memory Constraints

Memory is one of the primary limitations in TinyML.

A conventional deep learning model may contain millions of parameters, while a microcontroller may have only a small amount of available memory. Furthermore, memory is required not only to store model parameters but also for input data, intermediate activations, program instructions and runtime operations.

Therefore, reducing model size is a major research objective.

3.3 Low-Power Hardware

TinyML devices are often expected to operate for long periods using batteries or harvested energy. Consequently, energy efficiency is as important as model accuracy.

Hardware optimisation can involve:

- [14] Low-power processors
- [15] Dedicated neural accelerators
- [16] Efficient memory architectures
- [17] Dynamic power management
- [18] Hardware-specific instruction sets
- [19] Reduced-precision arithmetic

Specialised accelerators can improve neural network execution efficiency by accelerating operations commonly used by machine learning models.

3.4 Hardware–Software Co-Design

An important emerging direction is hardware–software co-design. Instead of designing the machine learning model and hardware independently, both can be optimised together.

For example, a model can be designed to use operations efficiently supported by a particular MCU or accelerator. Similarly, hardware can be designed around the computational requirements of a particular class of neural networks.

This approach can improve the overall balance between accuracy, latency, memory and energy consumption.

4 Machine Learning Model Optimisation

A major challenge in TinyML is reducing the computational and memory requirements of ML models while maintaining acceptable accuracy.

4.1 Quantisation

Quantisation converts model parameters and operations from higher-precision numerical formats to lower-precision representations.

For example, a model using 32-bit floating-point parameters can potentially be converted to 8-bit integer representations.

The benefits include:

- [20] Reduced model size
- [21] Reduced memory requirements
- [22] Lower computational complexity
- [23] Improved inference efficiency

Recent experimental work demonstrates that 8-bit post-training quantisation can significantly reduce model storage requirements while allowing machine learning models to operate within microcontroller memory budgets.

However, excessive quantisation can reduce model accuracy. Therefore, TinyML design involves finding a suitable balance between precision and performance.

4.2 Pruning

Pruning removes parameters or connections that contribute relatively little to model performance.

There are two broad approaches:

Unstructured pruning: individual parameters are removed.

Structured pruning: entire filters, channels or groups of parameters are removed.

Structured pruning is particularly attractive for embedded systems because the resulting models can be easier to execute efficiently on hardware.

4.3 Knowledge Distillation

Knowledge distillation transfers knowledge from a larger teacher model to a smaller student model.

The teacher network can achieve high accuracy but may be too large for a microcontroller. The student network is trained to reproduce important behaviour of the teacher while using fewer parameters.

This technique can therefore provide a practical method for creating compact models.

4.4 Lightweight Neural Networks

Another approach is to design neural networks specifically for constrained hardware.

Examples include lightweight convolutional architectures and compact neural networks designed to reduce the number of operations and parameters.

The development of lightweight models is important for applications such as image recognition, speech recognition and sensor classification.

4.5 Neural Architecture Search

Neural Architecture Search (NAS) attempts to automatically identify neural network architectures suitable for a particular hardware environment.

Rather than manually selecting the architecture, the search process can consider constraints such as:

- [24] Accuracy
- [25] Latency
- [26] Flash usage
- [27] SRAM consumption
- [28] Energy consumption

This can enable hardware-aware model design.

5 TinyML Software Ecosystem

The TinyML software ecosystem bridges the gap between machine learning development and embedded deployment.

A typical workflow consists of:

Dataset collection → Data preprocessing → Model training → Model optimisation → Model conversion → Embedded deployment → Testing

The model is generally trained using comparatively powerful computing resources. After training, the model is converted into an embedded-compatible format and deployed on the target MCU.

Frameworks and libraries are important because they provide optimised implementations of common machine learning operations.

Modern TinyML research also considers compilers, runtime systems and hardware-specific kernels. Recent literature reviewing EdgeML and TinyML identifies model conversion,

inference mechanisms, learning strategies, hardware platforms and software frameworks as major components of the deployment ecosystem.

6 TinyML Deployment Workflow

6.1 Data Collection

The first stage is collecting representative sensor data.

Depending on the application, the data may consist of:

- [29] Audio
- [30] Images
- [31] Accelerometer signals
- [32] Gyroscope measurements
- [33] Temperature
- [34] Pressure
- [35] Vibration
- [36] Electrical signals

The quality and diversity of the dataset strongly influence the final model.

6.2 Data Preprocessing

Raw sensor signals may contain noise or irrelevant information.

Preprocessing can involve filtering, normalisation, feature extraction and segmentation.

For example, vibration signals may be transformed into frequency-domain features before being provided to a machine learning model.

6.3 Model Training

Model training is generally performed off-device using a computer with substantially greater computational resources.

The model is trained using labelled or unlabelled data depending on the learning method.

6.4 Optimisation

After training, the model is optimised for embedded deployment.

Possible techniques include:

- [37] Quantisation
- [38] Pruning
- [39] Knowledge distillation
- [40] Architecture optimisation
- [41] Operator fusion
- [42] Hardware-specific optimisation

6.5 Deployment

The optimised model is converted into a format compatible with the embedded software environment and integrated with the sensor-processing application.

The final system must then be evaluated on the actual hardware.

6.6 Evaluation

TinyML evaluation should not consider accuracy alone.

Important metrics include:

Accuracy: How correctly does the model perform its task?

Latency: How long does one inference require?

Memory: How much Flash and SRAM are required?

Energy: How much energy is consumed per inference or per operation?

Model size: How much storage is required?

A recent TinyML deployment study explicitly evaluated the trade-offs between latency, memory, energy and model size, demonstrating why system-level evaluation is essential.

7 Applications of TinyML

7.1 Healthcare and Wearable Electronics

TinyML can enable intelligent processing in wearable and portable healthcare devices.

Wearable sensors can continuously collect physiological information, while an embedded ML model can identify patterns locally.

Potential applications include:

- [43] Activity recognition

- [44] Physiological signal classification
- [45] Sleep-related monitoring
- [46] Audio-based health assessment
- [47] Personalised wearable systems

A recent IEEE Access systematic review analysed 136 peer-reviewed TinyML publications from 2020–2025 and identified healthcare as one of the major application domains.

7.2 Environmental Monitoring

TinyML is suitable for remote environmental sensors because these devices may need to operate for long periods without frequent maintenance.

Applications include:

- [48] Air-quality monitoring
- [49] Water-quality monitoring
- [50] Wildlife monitoring
- [51] Acoustic environmental sensing
- [52] Weather-related monitoring

Local processing can allow the sensor to identify important events without continuously transmitting all collected data.

7.3 Smart Agriculture

Agricultural environments contain numerous variables such as soil moisture, temperature, humidity and environmental conditions.

TinyML-enabled sensors can analyse these measurements locally and assist in:

- [53] Irrigation optimisation
- [54] Crop monitoring
- [55] Pest detection
- [56] Environmental classification
- [57] Precision agriculture

Recent TinyML literature identifies smart agriculture as an important emerging application area.

7.4 Industrial IoT

Industrial machines produce large quantities of sensor information.

TinyML can analyse vibration, acoustic or electrical signals to identify abnormal operating conditions.

Potential applications include:

- [58] Predictive maintenance
- [59] Machine condition monitoring
- [60] Fault detection
- [61] Industrial safety
- [62] Process monitoring

Industrial applications benefit particularly from low latency because abnormal conditions may need to be detected immediately.

7.5 Smart Cities

TinyML can contribute to intelligent infrastructure through distributed sensing.

Potential applications include:

- [63] Traffic monitoring
- [64] Waste management
- [65] Environmental sensing
- [66] Energy management
- [67] Infrastructure monitoring

A distributed architecture can reduce the amount of raw data transmitted to central servers.

8 Major Challenges

8.1 Accuracy–Efficiency Trade-Off

The most important challenge is achieving sufficient accuracy while maintaining low memory usage, latency and energy consumption.

Increasing model complexity can improve accuracy but may exceed the hardware's capabilities.

Therefore, TinyML is fundamentally a multi-objective optimisation problem.

8.2 Limited Memory

Small SRAM and Flash capacities constrain model size and intermediate computations.

Large neural networks cannot simply be transferred to a microcontroller without modification.

8.3 Energy Consumption

Although TinyML reduces communication requirements, local computation still consumes energy.

This is particularly important for battery-operated sensors expected to operate for months or years.

8.4 Hardware Heterogeneity

TinyML devices vary considerably in processor architecture, memory, accelerators, sensors and communication technologies.

This makes it difficult to develop a single model that performs optimally across all platforms.

8.5 Security and Privacy

Local processing can improve privacy because raw data does not necessarily need to leave the device. However, the embedded device itself can still be exposed to security threats.

Future TinyML systems therefore need lightweight security mechanisms that do not consume excessive computational or energy resources.

8.6 Dataset Limitations

A model trained using one environment may perform poorly in another.

For example, an acoustic classification model may encounter different background noise in real-world deployment compared with its training dataset.

This creates a need for robust datasets and adaptive models.

9 Emerging Research Directions

9.1 On-Device Learning

Most TinyML systems perform inference using a model trained elsewhere.

An emerging direction is allowing the device to adapt the model using newly observed data.

On-device learning could allow systems to respond to changing environments without continuously sending data to the cloud.

Recent research has specifically investigated incremental learning on microcontrollers to enable local adaptation under resource constraints.

9.2 Federated Learning

Federated learning allows multiple devices to contribute to model training without directly sharing their raw data.

This approach could combine the privacy advantages of local processing with the ability to improve a shared model.

However, federated learning on extremely constrained devices remains challenging because of communication, memory and computational requirements.

9.3 Sustainable TinyML

Energy efficiency is becoming increasingly important as the number of intelligent edge devices increases.

Future systems may combine:

- [68] Ultra-low-power MCUs
- [69] Energy-efficient models
- [70] Dynamic power management
- [71] Energy harvesting
- [72] Sustainable hardware

Recent TinyML research specifically identifies energy harvesting and environmentally sustainable hardware as future directions.

9.4 Energy Harvesting

Energy harvesting can provide power from environmental sources such as light, vibration or thermal gradients.

Combining energy harvesting with TinyML could allow sensors to operate with reduced dependence on conventional batteries.

This is particularly promising for remote environmental monitoring and distributed IoT systems.

9.5 Neuromorphic Computing:

Neuromorphic computing attempts to reproduce some principles of biological neural processing using specialised hardware.

Spiking neural networks can potentially provide highly energy-efficient processing for suitable applications.

The integration of neuromorphic hardware with TinyML represents an interesting future research direction.

9.6 Edge-to-Cloud Continuum:

The future of intelligent embedded systems may not involve choosing exclusively between local processing and cloud processing.

Instead, computation can be distributed across:

Sensor → MCU → Edge device → Cloud

Simple, latency-sensitive tasks can be performed locally, while computationally intensive training or analysis can be performed at higher levels.

Recent TinyML research identifies this edge-to-cloud continuum as an important future direction.

10 Discussion:

The development of TinyML represents a shift in the design philosophy of embedded systems. Conventional embedded systems were generally programmed using deterministic algorithms, whereas TinyML systems can learn patterns from data and make intelligent decisions.

This creates an important intersection between ECE and artificial intelligence.

From an ECE perspective, TinyML involves several traditional engineering areas:

- [73] Sensors and instrumentation
- [74] Microcontrollers
- [75] Digital signal processing
- [76] Communication systems
- [77] Low-power electronics
- [78] Embedded programming
- [79] Hardware architecture

At the same time, it introduces machine learning concepts such as neural networks, optimisation and model training.

The literature indicates that TinyML is still a developing field. A 2026 bibliometric analysis found rapid publication growth between 2020 and 2024, with electrical and electronics engineering representing the largest research category in its dataset.

However, the field should not be viewed simply as “running a small AI model on a microcontroller”. Successful TinyML deployment requires system-level optimisation. The sensor, preprocessing algorithm, model, memory architecture, processor, communication system and power source all influence the final performance.

For this reason, future TinyML research is likely to increasingly focus on hardware–software co-design rather than isolated optimisation of machine learning algorithms.

11 Future Scope

The future development of TinyML is likely to involve increasingly capable embedded processors, specialised accelerators and more efficient machine learning algorithms.

One major direction is the development of models capable of learning continuously from their environment. Such systems could adapt to changes without requiring complete retraining in the cloud.

Another important direction is privacy-preserving TinyML. Local inference can reduce the need to transmit sensitive raw information, while lightweight security mechanisms can protect models and data.

The combination of TinyML and energy harvesting could further expand the deployment of autonomous sensors. Such systems could potentially operate for long periods in remote locations with minimal maintenance.

Research is also moving towards specialised hardware and hardware-aware machine learning. Rather than adapting existing large models to constrained hardware, future systems may be designed jointly from the beginning.

Recent literature identifies advanced sensor integration, edge-to-cloud architectures, sustainable TinyML, personalised systems, ethical AI, resilient systems, evolutionary frameworks and improved security as significant future research directions.

CONCLUSIONS

Tiny Machine Learning is emerging as an important technology for bringing artificial intelligence into resource-constrained embedded systems. By enabling machine learning inference directly on microcontrollers and edge devices, TinyML can reduce latency, communication requirements and dependence on cloud computing.

The major technical challenges arise from the limited memory, computational capability and energy availability of embedded hardware. Techniques such as quantisation, pruning, knowledge distillation, lightweight neural networks and hardware–software co-design provide important approaches for addressing these constraints.

TinyML has applications across healthcare, environmental monitoring, agriculture, industrial IoT, smart cities and wearable electronics. Recent research demonstrates that the field is expanding rapidly, with increasing attention towards on-device learning, federated learning, sustainable hardware, energy harvesting, privacy and security.

For Electronics and Communication Engineering, TinyML is particularly significant because it combines embedded hardware, sensors, signal processing, communication and artificial intelligence into a single system. Future progress will depend on developing efficient algorithms, specialised hardware, adaptive models and reliable system architectures.

Overall, TinyML represents a transition from connected sensing to **intelligent sensing**, in which embedded devices are not merely collecting and transmitting information but are increasingly capable of interpreting their environment and making decisions locally

ACKNOWLEDGEMENT

The author would like to express sincere gratitude to the Department of Electronics and Communication Engineering, Faculty of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India, for providing the academic environment and support necessary for the preparation of this review article. The author also acknowledges the researchers and authors whose published works have contributed to the understanding of Tiny Machine Learning, embedded systems, and edge artificial intelligence. Their contributions provided valuable insights for the preparation of this review.

REFERENCES

- [1] R. Kallimani, K. Pai, P. Raghuwanshi, S. Iyer, and O. L. A. López, "TinyML: Tools, applications, challenges, and future research directions," *Multimedia Tools and Applications*, 2023, doi: 10.1007/s11042-023-16740-9.
- [2] L. Dutta and S. Bharati, "TinyML meets IoT: A comprehensive survey," *Internet of Things*, vol. 16, 2021.
- [3] P. P. Ray, "A review on TinyML: State-of-the-art and prospects," *Journal of King Saud University – Computer and Information Sciences*, 2022.
- [4] N. Schizas, K. Karras, C. Karras, and C. Karras, "TinyML for ultra-low power AI and large scale IoT deployments: A systematic review," *Future Internet*, vol. 14, no. 12, 2022.
- [5] M. Han and M. Siebert, "TinyML: Where machine learning meets embedded systems," 2022.
- [6] V. Tsoukas *et al.*, "A review of machine learning and TinyML in healthcare," 2024.
- [7] H. Han, S. Trimi, and S. M. Lee, "Tiny Machine Learning (TinyML): Research trends and future application opportunities," *Array*, vol. 29, Art. no. 100674, 2026, doi: 10.1016/j.array.2025.100674.
- [8] C. Yahyati *et al.*, "A systematic review of state-of-the-art TinyML applications in healthcare, education, and transportation," *IEEE Access*, vol. 13, pp. 204513–204562, 2025, doi: 10.1109/ACCESS.2025.3633575.
- [9] J. D. Velásquez-Henao, L. Cadavid, and C. J. Franco, "Emerging trends and strategic opportunities in tiny machine learning: A comprehensive thematic analysis," *Neurocomputing*, vol. 648, Art. no. 130746, 2025, doi: 10.1016/j.neucom.2025.130746.
- [10] "A literature review on model conversion, inference, and learning strategies in EdgeML with TinyML deployment," *Computers, Materials & Continua*, vol. 83, no. 1, pp. 13–64, 2025, doi: 10.32604/cmc.2025.062819.
- [11] C. M. Bhushan *et al.*, "Deploying TinyML for energy-efficient object detection and communication in low-power edge AI systems," *Scientific Reports*, vol. 15, Art. no. 44299, 2025.
- [12] "Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions," *Sensors*, 2025.
- [13] "Advancing TinyML in IoT: A holistic system-level perspective for resource-constrained AI," *Future Internet*, vol. 17, no. 6, Art. no. 257, 2025.
- [14] "TinyML on microcontrollers: Enabling energy-efficient, real-time, privacy-preserving incremental learning for embedded systems," in *Proc. IEEE International Conference on Sustainable Communication Networks and Application*, 2025, doi: 10.1109/ICSCSN67106.2025.11308574.
- [15] "Embedded low-power machine-learning on microcontrollers: Systematic literature review," *AiBi Journal of Research, Administration and Engineering*, 2025.
- [16] S. A. R. Zaidi *et al.*, "Unlocking edge intelligence through Tiny Machine Learning (TinyML)," 2022.
- [17] R. Sanchez-Iborra and A. F. Skarmeta, "TinyML-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits and Systems Magazine*, 2020.
- [18] X. Y. Wang *et al.*, "FANN-on-MCU: An open-source toolkit for energy-efficient neural network inference at the edge of the Internet of Things," 2020.
- [19] T. S. Ajani *et al.*, "An overview of machine learning within embedded and mobile devices—Optimizations and applications," 2021.
- [20] M. Shafique *et al.*, "TinyML: Current progress, research challenges, and future roadmap," 2021.