A Comprehensive and Comparative Analysis of Tiny Machine Learning (TinyML): Current Trends, Technological Integrations, and Future Opportunities

Ramandeep Kaur, Department of Computational Science, Maharaja Ranjit Singh Punjab Technical University, Bathinda.

Jasveer Kaur, Department of Computational Science, Maharaja Ranjit Singh Punjab Technical University, Bathinda.

1. Introduction and Background

In the past few years the rapid boom of AI led to the implementation of ML on several research and business applications. However, the implementation and operation of large Machine Learning models on high computational power machines not only cause considerable costs but also contribute to an environmental impact. As a general approximation for present computing technology, an average CPU draws power around 85 Watts, a standard GPU could consume 200 to 500 Watts of power. In order to reduce the environmental effects and costs of AI, a paradigm called Tiny Machine Learning (TinyML) has been developed [3].

TinyML is the practice of deploying highly optimized and compressed Machine Learning models on devices that consume extremely low power, such as battery-powered devices like microcontroller units(MCUs). ML inference tasks on hardware under 1 mW are classified as TinyML. Usually devices are characterized by very little storage(typically between 32 to 512 KB of SRAM) and limited amounts of flash [6]. Being usually powered by the usage of a battery it is crucial for these TinyML devices to minimize energy consumption to mere milliwatts or microwatts, so that they may run for months or year.

With time, a special subdomain of TinyML has surfaced, which goes by the name of Tiny Deep Learning (TinyDL). If classical TinyML systems consisted of light and shallow models, such as linear classifiers, trees, or perceptrons, TinyDL involves the deployment of deep neural network architectures, such as convolution- and transformer- based neural network, on severely resource constrained devices, where extreme on-device optimization for low power consumption and minimal memory (often under 1 MB in the KB) requirements is key.

1.1 The Shift from Cloud to Edge

The typical architecture of IoT heavily relies on cloud computing for high-level data processing and data storage. The data from end nodes is transferred and collected in gateway nodes, which further transfers the data to the cloud through broadband internet connection. This architecture has some serious limitations like:

1. Power: Onboard processing consumes considerably less power compared to wireless data transmission (several times less).

2. Latency: The delay in data transmission between the cloud server and source of the data leads to slow decisions and random delays. The end-to-end delay for cloud solutions is non-deterministic.

3. Privacy and Security: Transferring the raw data to the cloud imposes significant risks to the users, while moving the inference to the edge restricts the raw data from going to the cloud.

4. Connectivity: In places with patchy/ no internet connection, cloud based models do not function.

Therefore, the current trend is moving the computational paradigm from cloud to end-edge-cloud and empowering IoT nodes with intelligence.

2. Current Trends: The Evolution from TinyML to TinyDL

The transition from classical TinyML to TinyDL was driven by the limitations of shallow models. Shallow classifiers were unable to match the representational power of deep neural networks and required extensive manual feature engineering, particularly for complex audio and vision tasks.

2.1 Limitations of Classical TinyML

- Generalization Error: Limited capacity of the models often lead to high generalization error and can results in low accuracy on new unseen data.
- Features Engineering: Creating raw sensor data as the meaningful input for classic models, either by hand-crafting or by applying some transformation methods for traditional models is also another pain point of the model construction pipeline. We might miss complex patterns.
- The Performance Limit: for VWW type tasks, the traditional VWW pipeline leveraging classic classifiers over hand-crafted features might cap around 70%–75% of accuracy.

2.2 Enablers of TinyDL

The realization of TinyDL relies on interrelated advances in architecture and model compression.

- Architectural innovations such as depthwise separable convolutions, inverted residuals, and attention mechanisms have made it possible to compress model complexity.
- Optimization techniques, including quantization-aware training (QAT), structured pruning, and knowledge distillation, have dramatically reduced runtime demands [1].
- Neural Architecture Search (NAS) frameworks successfully co-optimize model topology and deployment constraints.

3. Hardware Ecosystem and Constraints

The widespread acceptance of TinyML and TinyDL relies heavily on the capabilities of underlying hardware platforms. MCUs form the backbone of these deployments, acting as single-chip computers designed for low-power operation.

3.1 Microcontroller Units (MCUs)

The market has seen an emergence of a new 32-bit generation of IoT-ready microcontrollers. Because of support for single instruction multiple data (SIMD) and digital signal processing (DSP) instructions, Cortex-M-based devices can now perform previously unachievable tasks.

- **Espressif ESP32:** A dual-core 32-bit MCU running at 240 MHz, equipped with 520 KB SRAM and integrated Wi-Fi/Bluetooth, commonly used for simple keyword spotting (KWS).
- **STM32 Family (e.g., STM32F7):** Features a 216 MHz ARM Cortex-M7 MCU with approximately 320 KB RAM and DSP instructions, ideal for industrial sensing.
- **Arduino Nicla Vision:** Features a Dual Arm Cortex M7/M4 processor running at 480MHz and 240MHz respectively, equipped with 2MB Flash, 1MB SRAM, a camera, microphone, and IMU.
- **Sony Spresense:** Utilized heavily in embedded machine learning pipelines for precision agriculture due to its robust camera and processing capabilities.

3.2 Specialized AI Hardware and Accelerators

While general-purpose MCUs provide flexibility, specialized hardware provides the best TinyML performance efficiency. These application-specific integrated circuits (ASICs) and co-processors are built to handle neural network inference with minimal energy.

- **Google Edge TPU:** Designed to accelerate TensorFlow Lite models at the edge, performing 4 trillion operations per second (4 TOPS) while consuming about 2W of power. It can run MobileNet V2 at nearly 400 frames per second [2].
- **Syntiant NDP120:** An ultra-low-power neural accelerator aimed at always-on workloads. It performs KWS inference in about 4.3 ms while consuming only 35 microjoules of energy per inference.
- **Himax WE-I Plus:** An event-driven AI MCU that stays in a near-standby mode until its sensor accelerator detects a trigger. When running a person-detection CNN, its average power consumption can be under 5 mW.

3.3 Network Connectivity for Edge Devices

Achieving connectivity for tiny embedded devices while adhering to low energy restrictions is heavily supported by Low-Power Wide-Area Networks (LPWAN).

- **LoRaWAN:** Uses a modulation method called Chirp Spread Spectrum (CSS) to send data over vast distances while consuming minimal power [5].
- **Sigfox:** Deploys base stations using ultra-narrowband (UNB) technology, resulting in great receiver sensitivity and low-cost antenna design.
- **NB-IoT:** An air interface that is part of the LTE standard but reduced to promote long battery life and low-cost devices [4].

4. Software Toolchains and MLOps Frameworks

TinyML relies mostly on software optimization, which allows machine learning models to operate on devices with limited processing power and memory. An ML interpreter is a tool used to implement machine learning algorithms on embedded devices without relying on dynamic memory allocation.

4.1 Deployment Frameworks and Interpreters

- **TensorFlow Lite Micro (TFLM):** An extension of TensorFlow designed to execute models on 32-bit microcontrollers with limited memory. It is the de facto baseline in TinyML deployments due to its support for 8-bit quantization and expanded operator coverage. Its core runtime fits in just 16 KB on an Arm Cortex M3 [19][20].
- **Edge Impulse:** A cloud-based solution that facilitates the creation and deployment of machine learning models via a no-code AutoML pipeline. It employs the EON compiler, which compiles neural networks directly into C++ source code, running the same network with 25% to 55% less SRAM compared to TFLM [21].
- **CMSIS-NN:** A collection of optimized neural network kernels designed by Arm for maximum performance and minimum memory footprint on Cortex-M processor cores.
- **MicroTVM:** An extension of the TVM compilation stack that brings auto-tuning and graph optimization to MCU platforms like STM32 and ESP32 [8].
- **Glow:** Developed by Meta, this tool offers ahead-of-time graph lowering for hardware accelerators and NPUs.

4.2 Model Optimization Techniques

To fit models into kilobyte-scale memory, extensive compression is required.

- **Pruning:** Begins with training the network and then eliminating weights that fall below a specific threshold, resulting in a trimmed and sparser model [32]. Structured pruning targets entire network components such as filters or layers, achieving compression ratios up to 13x without significant accuracy loss.
- **Quantization:** Reduces the precision of weights and activations from 32-bit floating-point numbers to 8-bit or lower fixed-point numbers [23]. Post-Training Quantization (PTQ) converts models to INT8 formats, while Quantization-Aware Training (QAT) simulates quantization noise during training to preserve accuracy [22].
- **Low-Rank Factorization:** Decomposes a dense weight matrix into the product of two lower-dimensional matrices with lower ranks, reducing dimensionality while keeping significant properties.

- **Huffman Coding:** A lossless compression method that assigns shorter binary codes to frequently appearing symbols and longer codes for less frequently occurring symbols.
- **Knowledge Distillation:** A technique where a smaller "student" model is trained to mimic the outputs of a larger, more accurate "teacher" model.

4.3 MLOps and Machine Learning as a Service (MLaaS)

MLOps is a set of engineering practices unifying ML system development to ensure continuous integration (CI), continuous development (CD), and continuous testing (CT). TMLaaS (TinyML-as-a-Service) expands on this by splitting learning functions across end devices and gateways.

- **Device Manager:** Coordinates commissioning, authentication, and device shadowing for IoT nodes [15].
- **Cloud Manager:** Authenticates the gateway with backend cloud services and provides connectivity management for LPWAN [27].
- **Resource Manager:** Performs local storage management and memory management for ML functions.
- The MLOps lifecycle for these systems includes strict sampling, data structuring, training, validation, deployment, and continuous updates via CI/CD pipelines to guarantee Quality of Service (QoS) [7].

5. Lightweight Deep Learning Architectures

The adaptation of deep learning for the extreme edge relies on highly specialized architectures designed to bypass traditional computational bottlenecks.

5.1 Convolutional Neural Networks (CNNs)

- **MobileNet Family:** Introduces depthwise separable convolutions that factorize standard convolutions to reduce parameters by an order of magnitude. MobileNetV2 extends this with inverted residual blocks, achieving 71.8% ImageNet accuracy with only a 3.4 MB footprint on STM32H7 MCUs [9].
- **SqueezeNet:** Employs fire modules consisting of squeeze layers followed by expand layers to achieve significant parameter reduction, suitable for flash-constrained MCUs [31].
- **MCUNet:** Achieves 68.7% ImageNet accuracy with only a 0.51 MB model size through the joint optimization of network architecture and inference scheduling.
- **Tiny YOLO & Tiny SSD:** Compact networks built for real-time object detection; Tiny YOLO reduces model size via depthwise and pointwise convolutions, while Tiny SSD uses Fire microarchitecture [33].

5.2 Lightweight Transformers and RNNs

- **TinyBERT:** Employs a two-stage knowledge distillation framework specifically designed for transformers, achieving 96.7% performance of BERT-Base while being 7.5x smaller.

- **DistilBERT:** Reduces the original BERT model by 40% while retaining 97% of its language understanding capabilities via student-teacher training.
- **FastGRNN:** Extends the residual connection to a gate by reusing RNN matrices, achieving state-of-the-art accuracy in a kilobyte-sized model.

6. Comprehensive Applications of TinyML and TinyDL

TinyML has catalyzed a revolution across multiple industries by allowing intelligent processing to occur directly at the source of data generation.

6.1 Healthcare and Human Behavior Analytics

The human body acts as a continuous source of signals that sensors can collect to mitigate healthcare dilemmas via embedded edge computing.

- **Blood-Pressure Monitoring:** Researchers have developed "TinyCare", an edge device that monitors high blood pressure using ML models deployed on hardware like the Arduino Uno, ESP32 Wrover, and AdaFruit PyBadge.
- **Hearing Aids:** Noise suppression models built via Recurrent Neural Networks (RNNs) and pruning techniques have been deployed directly on hearing aid hardware to enhance speech hearing in noisy environments.
- **Parkinson's Disease:** Wearables equipped with an ATMega2560 microcontroller use accelerometer signals to detect and recover from Freezing of Gait (FoG) in Parkinson’s patients, achieving 93.58% accuracy.
- **Epilepsy Treatment:** An ultra-low-powered wearable utilizing an STM32L476 ARM Cortex-M4 runs a Random Forest classifier to detect epileptic seizures from EEG data, achieving up to 40.87 hours of continuous monitoring on a single charge.
- **Cardiac Arrhythmia:** Single-lead Electrocardiograms (ECGs) processed by Convolutional Neural Networks on Cortex-M4 cores achieve a power efficiency of 1.64 GOps/s/W for arrhythmia detection. Tiny Reservoir Networks are also used for processing ECGs to recognize pathological conditions [35].
- **Emotion Detection:** Smart wearables recognize emotions through physiological signals (e.g., photoplethysmography, skin conductance, heart rate) using devices like the Shimmer GSR+.
- **Disease Vector Identification:** TinyML prototypes based on the Arduino Nano BLE 33 Sense can identify deadly mosquitoes by classifying audio data of their wing beats, achieving 88.3% accuracy [26].

6.2 Smart Farming and Agriculture

Smart Farming (SF) relies on IoT to monitor crop health, detect diseases, and manage irrigation, utilizing multi-layer architectures (physical-edge-fog-cloud).

- **Crop Disease Detection:** The Plant Village project created the Nuru app to identify plant diseases locally via TensorFlow Lite on mobile phones. In specific hardware applications, a compressed CNN model detects grape leaf esca diseases using an OpenMV Cam

STM32H7 mounted on a moving agricultural vehicle. Another system detects coffee plant diseases using an STM32F746G-disco board connected to an STM32F4DISCAM module inside a specialized box equipped with LEDs [14].

- **Drought Stress and Crop Selection:** A system utilizing a Raspberry Pi Zero W and a Sony IMX219 camera runs a CNN model to detect drought stress in soybeans. Additionally, an STM32F401CC device runs a lightweight neural network to recommend the best crops to grow based on soil nitrogen, phosphorus, potassium, and pH levels.
- **Fruit classification and Counting:** A prototype built on an ESP-32 identifies and classifies fruits and vegetables in real-time using a MobileNet v1 model [30][31]. Furthermore, the FOMO algorithm deployed on an Arduino Portenta H7 helps farmers detect and count strawberries to optimize harvest times.
- **Smart Greenhouse Management:** TinyML models control micro-climates by monitoring temperature, solar illuminance, and CO2 concentrations to automatically perform greenhouse control actions. Other devices detect greenhouse gases (NH3, CH4, N2O) using ANN algorithms compressed via X-CUBE-AI [13].
- **Smart Irrigation:** Low-cost ESP32-CAM devices process images to read numerical water meters and transmit the data using LoRaWAN, scoring an 88% accuracy. Another system uses an Arduino Uno to run Decision Tree models that analyze sensor data to autonomously decide whether to trigger sprinklers ("irrigation" or "fertigation") [28][29].

6.3 Environmental and Biosphere Monitoring

TinyML provides rugged, offline capabilities essential for monitoring the natural environment where power and internet are scarce.

- **Atmospheric Monitoring:** TinyML edge devices plugged into on-board diagnostics systems track vehicular carbon dioxide emissions in real-time, utilizing Typicality and Eccentricity Data Analytics (TEDA) to detect anomalies [10]. Edge computing methodologies on devices like the Arduino Nano 33 BLE Sense also predict temperature and atmospheric pressure.
- **Hydrosphere Monitoring:** Raspberry Pi setups analyze water quality variables (pH, chemical concentrations) to conserve water. Deep learning ANNs evaluate lake toxicity, and edge AI protects water supplies from cholera by functioning offline to monitor physicochemical parameters with 96% accuracy.
- **Underwater Imaging:** Due to the narrow bandwidth of underwater acoustic channels, battery-free cameras use a "fish visual wake word" (fishVWW) model on an STM32L476RG MCU to take and send images only when a fish is detected, saving massive amounts of power.
- **Lithosphere and Geology:** Hyperspectral imaging (HSI) is used to classify rocks and minerals. Global-scale earthquake detection systems rely on TinyML capabilities deployed on stationary MEMS accelerometers driven by Cortex M4 microcontrollers.
- **Wildlife Conservation:** The ElephantEdge project utilizes Edge Impulse Studio to create models that process audio and accelerometer data to detect poacher attacks (via kalashnikov sounds) and monitor elephant physical activity. TinyML is also deployed on SmallSats to track sea turtles.

- **Bio-activity and Soundscapes:** Ultrasonic sensors track bat activity and location. A Wireless Acoustic Sensor Network (WASN) classifies environmental sounds into anthrophony, traffic, biophony, and geophony using an nRF52840 processor and Mel-frequency cepstral coefficients (MFCCs).

6.4 Industrial Anomaly Detection

Anomaly detection refers to identifying patterns that do not conform to expected behaviors.

- **Condition Monitoring:** An STM32H743ZI2 MCU utilizes an auto-encoder model to detect anomalies in rotating machinery by processing vibration signals. ESP32-DEVKIT kits installed in wastewater submersible pumps extract temperature and vibration features to locally train Isolation Forest models to detect failures.
- **Washing Machines:** Autoencoders deployed on an Arduino Nano 33 BLE Sense detect mechanical anomalies, such as imbalanced laundry loads, achieving roughly 92% accuracy [24].
- **Predictive Maintenance:** ESP-WROOM-32 MCUs analyze thermal images of hydraulic refrigeration systems using CNNs to identify anomalies [25].
- **Wind Turbines:** A Block-based Binary Shallow Echo State Network (BBS-ESN) operates as a deeply quantized anomaly detector to identify oil leaks in wind turbines using STMicroelectronics hardware [18].
- **Intelligent Vehicles:** Devices attached to vehicles detect road anomalies like potholes and bumps using unsupervised TEDA algorithms running on an Arduino Nano 33 IoT.

7. Evaluation Metrics and Benchmarking

Standardized benchmarking is required to ensure reproducibility and fair comparisons across highly fragmented hardware platforms.

- **MLPerf Tiny:** Developed by MLCommons, this benchmark defines four core tasks: Keyword Spotting (KWS), Visual Wake Words (VWW), Image Classification, and Anomaly Detection. It evaluates systems based on single-stream inference mode, mimicking real-time sensor workloads.
- **Efficiency Metrics:** In TinyML, performance revolves around Model Size (measured in kilobytes, capped strictly by flash storage), Inference Latency (wall-clock time in milliseconds), Memory Usage (static footprint and dynamic RAM buffers), Operations Count (MACs/FLOPs), and Energy Consumption (measured in millijoules per inference).
- **Accuracy Tradeoffs:** Quantization introduces a balance between memory savings and performance loss. For instance, structured pruning of MobileNetV2 incurs less than a 1% top-1 drop on ImageNet while shrinking model size by 1.9 times. Post-Training Quantization can reduce accuracy by up to 3-4%, though QAT mitigates this by introducing fake quantization nodes during training.

8. Open Challenges in TinyML Deployments

Despite massive progress, several technical and societal limitations must be addressed to fully realize TinyML's potential.

- **Concept Drift and Adaptation:** Current TinyML solutions predominantly rely on offline learning, meaning the models are static once deployed. They cannot adapt to environmental evolution, leading to performance drops known as concept drift.
- **Severe Memory and Compute Limits:** Achieving high accuracy for complex tasks remains difficult when model sizes are capped under 1MB [17]. The memory hierarchy complicates intermediate activation handling, forcing a trade-off between model performance and energy drain.
- **Hardware and Software Heterogeneity:** The fragmentation of the MCU market means that there is no unified standard for TinyML implementation [11]. Different systems require unique software, creating experimental complexity and discrepancies between deployment frameworks.
- **Security and Privacy Vulnerabilities:** While local processing enhances privacy, the physical accessibility of edge devices leaves them vulnerable to extraction and evasion attacks via adversarial examples. Sensors like cameras and microphones can be exploited to collect sensitive private data. Secure model updates require cryptographic validation, which introduces power overheads that contradict the goals of ultra-low-power AI [16].
- **Reliability:** Variations in hardware precision, environmental exposure (like high-energy particles), and hardware aging can modify algorithmic outputs and reduce overall device reliability.

9. Future Directions and Emerging Modalities

As the ecosystem matures, research is pivoting toward making edge models dynamic, collaborative, and biologically inspired.

9.1 On-Device Learning and Reformability

To combat concept drift, frameworks like TinyOL (TinyML with Online-Learning) are being developed to allow MCUs to learn from continuous streaming data in real-time [12]. Furthermore, "Reformable TinyML" allows systems to self-diagnose performance degradation and trigger an adaptation process without requiring a complete manual redeployment.

9.2 Federated Learning

Federated Learning (FL) relies on distributed training models where raw data never leaves the edge devices. Instead, locally computed ML updates (anonymized gradients) are transmitted to a central server, which averages them to improve a shared global model. Lightweight frameworks such as TinyFedTL demonstrate the aggregation of quantized updates directly on-device [34]. The combination of TinyML and FL perfectly addresses privacy and adaptability concerns.

9.3 Transfer Learning

Training deep models from scratch is computationally expensive. Transfer Learning (TL) enables edge devices to download pre-trained base models from centralized repositories and apply them to semantically similar but different tasks via minor local fine-tuning [37]. This modality circumvents the need for massive labeled datasets on the edge.

9.4 Neuromorphic Architectures

Future hardware paradigms include neuromorphic architectures utilizing Spiking Neural Networks. These structures mimic brain-inspired, event-driven processing, offering an alternative computational paradigm designed for always-on inference with absolute minimal power consumption on MCU-scale devices [36].

9.5 Tiny Foundation Models

Researchers are exploring "Tiny Foundation Models," which are miniaturized versions of massive pretrained models compressed via structured pruning and knowledge distillation to fit MCU scales. These general "backbone" models can support multiple lightweight task-specific heads, powered by Edge AutoML for customized fine-tuning.

10. Conclusion

The evolution of machine learning from centralized cloud infrastructures to the extreme edge represents a fundamental paradigm shift in artificial intelligence. TinyML, and increasingly TinyDL, have proven that highly complex, data-driven tasks can be executed on ultra-constrained microcontrollers operating on milliwatts of power. Driven by relentless innovations in hardware accelerators, LPWAN connectivity, and sophisticated software toolchains like TensorFlow Lite Micro and Edge Impulse, these edge devices are transforming global industries.

From securing patient privacy in wearable healthcare tech to enabling battery-free underwater biological monitoring and offline predictive maintenance in massive industrial plants, TinyML directly mitigates the latency, privacy, and power bottlenecks of traditional computing. While substantial challenges remain regarding hardware heterogeneity, concept drift, and adversarial security vulnerabilities, emerging paradigms like Federated Learning, Transfer Learning, and neuromorphic computing promise to make edge intelligence even more autonomous and resilient. Moving forward, the continued co-design of hardware and software will be essential to fully unlock the potential of ubiquitous, intelligent computing at the very edge of our networks.

References

[1] A. Curry, ‘‘The internet of animals,’’ Nature, vol. 562, no. 7727, pp. 322–326, 2018.

[2] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, ‘‘Deep learning-enabled medical computer vision,’’ NPJ Digit. Med., vol. 4, no. 1, pp. 1–9, Jan. 2021.

[3] A. Kumar, S. Goyal, and M. Varma, ‘‘Resource-efficient machine learning in 2 KB RAM for the Internet of Things,’’ in Proc. 34th Int. Conf. Mach. Learn., Sydney, NSW, Australia, 2017, pp. 1935–1944.

[4] Arduino. 2025. Get Started with Machine Learning on Arduino Nano 33 BLE Sense. Retrieved June 11, 2025 from https://docs.arduino.cc/tutorials/nano-33-ble-sense/get-started-with-machine-learning/ (2025)

[5] ARM Ltd. 2025. OctoML: Accelerating ML model deployment. Retrieved June 5, 2025 from https://www.arm.com/ partners/catalog/octoml. (2025). ARM Partner Catalog.

[6] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, ‘‘A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving,’’ J. King Saud Univ. Comput. Inf. Sci., vol. 34, no. 9, pp. 7366–7390, Oct. 2022.

[7] C. Banbury, C. Zhou, I. Fedorov, R. Matas, U. Thakker, D. Gope, V. J. Reddi, M. Mattina, and P. Whatmough, ‘‘MicroNets: Neural network architectures for deploying TinyML applications on commodity microcontrollers,’’ in Proc. Mach. Learn. Syst., San Jose, CA, USA, vol. 3, 2021, pp. 517–532.

[8] C. Liu, M. Jobst, L. Guo, X. Shi, J. Partzsch, and C. Mayr. 2023. Deploying machine learning models to ahead-of-time runtime on edge using MicroTVM. arXiv:2304.04842. Retrieved from https://arxiv.org/abs/2304.04842

[9] D. Gope, G. Dasika, and M. Mattina, ‘‘Ternary hybrid neural-tree networks for highly constrained IoT applications,’’ in Proc. Mach. Learn. Syst. (SysML) Conf., Palo Alto, CA, USA, vol. 1, 2019, pp. 190–200.

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

[11] D. Pau and P. K. Ambrose, ‘‘Automated neural and on-device learning for micro controllers,’’ in Proc. IEEE 21st Medit. Electrotechnical Conf. (MELECON), Jun. 2022, pp. 758–763.

[12] E. Strubell, A. Ganesh, and A. McCallum, ‘‘Energy and policy considerations for modern deep learning research,’’ in Proc. AAAI Conf. Artif. Intell., vol. 34, no. 9, 2020, pp. 13693–13696.

[13] F. A. Ozbay and B. Alatas, ‘‘Fake news detection within online social media using supervised artificial intelligence algorithms,’’ Phys. A, Stat. Mech. Appl., vol. 540, Feb. 2020, Art. no. 123174.

[14] G. Signoretti, M. Silva, P. Andrade, I. Silva, E. Sisinni, and P. Ferrari, ‘‘An evolving TinyML compression algorithm for IoT environments based on data eccentricity,’’ Sensors, vol. 21, no. 12, p. 4153, 2021.

[15] H. Bamoumen, A. Temouden, N. Benamar, and Y. Chtouki, ‘‘How TinyML can be leveraged to solve environmental problems: A survey,’’ in Proc. Int. Conf. Innov. Intell. Informat., Comput., Technol. (ICT), Nov. 2022, pp. 338–343.

[16] H. Han and J. Siebert, ‘‘TinyML: A systematic review and synthesis of existing research,’’ in Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC), Feb. 2022, pp. 269–274.

[17] J. Bai, F. Lu, K. Zhang, et al. 2025. ONNX: Open Neural Network Exchange. GitHub repository. (2025). Retrieved August 17, 2025 from https://github.com/onnx/onnx

[18] J. Duarte, E. Kreinar, J. Ngadiuba, et al. 2021. hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices. arXiv:2103.05579. Retrieved from https://arxiv.org/abs/2103.05579

[19] J. Lin, W.-M. Chen, Y. Lin, C. Gan, and S. Han, ‘‘McuNet: Tiny deep learning on IoT devices,’’ in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 11711–11722.

[20] J. R. Cheng and M. Gen, ‘‘Accelerating genetic algorithms with GPU computing: A selective overview,’’ Comput. Ind. Eng., vol. 128, pp. 514–525, Feb. 2019.

[21] L. Dutta and S. Bharali, ‘‘TinyML meets IoT: A comprehensive survey,’’ Internet Things, vol. 16, Dec. 2021, Art. no. 100461.

[22] LatentAI. 2025. From Cloud-First to Edge-First: The Future of Enterprise AI. White Paper. LatentAI. Retrieved from https://latentai.com/wp-content/uploads/2025/05/Cloud-to-Edge-White-Paper-FINAL.pdf

[23] Liangzhen Lai, Naveen Suda, and Vikas Chandra. 2018. CMSIS-NN: Efficient neural network kernels for arm Cortex-M CPUs. arXiv:1801.06601. Retrieved from https://arxiv.org/abs/1801.06601

[24] M. Al-Rubaie and J. M. Chang, ‘‘Privacy-preserving machine learning: Threats and solutions,’’ IEEE Secur. Privacy, vol. 17, no. 2, pp. 49–58, Mar. 2019.

[25] M. Tschannen, A. Khanna, and A. Anandkumar, ‘‘StrassenNets: Deep learning with a multiplication budget,’’ in Proc. 35th Int. Conf. Mach. Learn., Stockholm, Sweden, 2018, pp. 4985–4994.

[26] Mauro Conti, Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. 2009. (old) distributed data source verification in wireless sensor networks. Information Fusion 10, 4 (2009), 342–353. DOI:http://dx.doi.org/10.1016/j.inffus.2009.01.002

[27] N. Bostrom and E. Yudkowsky, ‘‘The ethics of artificial intelligence,’’ in Artificial Intelligence Safety and Security. Boca Raton, FL, USA: Chapman & Hall/CRC, 2018, pp. 57–69.

[28] N. N. Alajlan and D. M. Ibrahim, ‘‘TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications,’’ Micromachines, vol. 13, no. 6, p. 851, 2022.

[29] N. Rotem, J. Fix, S. Abdulrasool, G. Catron, S. Deng, R. Dzhabarov, N. Gibson, J. Hegeman, M. Lele, R. Levenstein, et al. 2018. Glow: Graph lowering compiler techniques for neural networks. arXiv:1805.00907. Retrieved from https://arxiv.org/abs/1805.00907

[30] 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, no. 12, p. 363, Dec. 2022.

[31] N. Vasilache et al. 2018. Tensor Comprehensions: Framework-agnostic high-performance machine learning abstractions. arXiv:1802.04730. Retrieved from https://arxiv.org/abs/1802.04730

[32] Nebuly Team. 2024. nebullvm: AI runtime optimization library. Retrieved June 5, 2025 from https://pypi.org/project/ nebullvm/. (2024).

[33] Neuton.AI. 2025. Neuton AI User Guide. Retrieved November 6, 2025 from https://neuton.ai/uploads/user-guide.pdf

[34] O. Baclic, M. Tunis, K. Young, C. Doan, and H. Swerdfeger, ‘‘Challenges and opportunities for public health made possible by advances in natural language processing,’’ Canada Communicable Disease Rep., vol. 46, no. 6, pp. 161–168, Jun. 2020.

[35] P. Mohan, A. J. Paul, and A. Chirania, ‘‘A tiny CNN architecture for medical face mask detection for resource-constrained endpoints,’’ in Innovations in Electrical and Electronic Engineering. Singapore: Springer, 2021, pp. 657–670.

[36] P. P. Ray, ‘‘A review on TinyML: State-of-the-art and prospects,’’ J. King Saud Univ. Comput. Inf. Sci., vol. 34, no. 4, pp. 1595–1623, Apr. 2022.

[37] R. R. Curtin, J. R. Cline, N. P. Slagle, W. B. March, P. Ram, N. A. Mehta, and A. G. Gray. 2013. MLPACK: A scalable C++ machine learning library. Journal of Machine Learning Research 14, 1 (2013), 801–805.