Comparative Analysis of Classical Machine Learning Vs. Deep Learning for Resource-Constrained Edge Devices

Pratiksha Rajendra Dashpute[1], Vikrant Satish Salunkhe[2], Sheetal Shrikant Shevkari[3]
[1,2]*Department of Science & Computer Science, Student, MIT ACSC Alandi (D), Pune, Maharashtra, India*
[3]*Assistant Professor, Department of Science & Computer Science, MIT ACSC Alandi (D), Pune, Maharashtra, India*
doi.org/10.64643/ijirt.208572-459

***Abstract*—Deploying machine learning models directly onto ultra-low-power microcontrollers and edge hardware (TinyML) is increasingly vital for real-time, privacy-preserving Internet of Things (IoT) applications. However, selecting the optimal algorithm under severe hardware constraints typically under 512 KB of RAM and tight power budgets presents a fundamental trade-off between computational overhead and predictive power.**

This paper presents a systematic comparative analysis evaluating traditional machine learning algorithms (Decision Trees, Random Forests, and Support Vector Machines) against lightweight, quantized Deep Learning architectures (compact CNNs and Multilayer Perceptrons).

Benchmark experiments are conducted across standardized datasets for tabular, acoustic, and vision tasks deployed on representative microcontroller platforms (e.g., ARM Cortex-M series, ESP32).

Models are evaluated across three primary dimensions: inference latency, memory footprint (Flash and peak SRAM usage), and predictive accuracy. Findings indicate that classical algorithms achieve significantly lower SRAM overhead and near-instantaneous inference times on structured sensor data, making them highly effective for ultra-low-memory nodes. Conversely, quantized neural networks achieve higher accuracy and generalization on complex spatio-temporal features, though they require higher Flash allocation and non-trivial peak RAM.

Based on these empirical trade-offs, we propose a lightweight decision matrix to guide model selection based on strict device hardware limits and application demands.

***Index Terms*—TinyML, Edge Computing, Machine Learning Benchmarking, Decision Trees, Neural Network Quantization, Inference Latency, Memory Footprint, Microcontrollers**

I. INTRODUCTION

The proliferation of Internet of Things (IoT) devices has renewed interest in pushing artificial intelligence workloads away from centralized cloud infrastructure and onto the sensing node itself. Tiny Machine Learning (TinyML) refers to the discipline of designing, optimizing, and deploying machine learning inference on microcontroller units (MCUs) and other ultra-low-power hardware, typically operating with kilobytes to a few hundred kilobytes of SRAM, sub-megabyte flash storage, and power budgets in the milliwatt or microwatt range [1]. By performing inference at the point of data acquisition, TinyML systems avoid the latency, bandwidth cost, and privacy exposure associated with continuous cloud connectivity [1], [2].

A recurring design decision facing TinyML practitioners is the choice between classical machine learning algorithms such as Decision Trees, Random Forests, and Support Vector Machines (SVMs) and increasingly compact deep learning architectures, including quantized Convolutional Neural Networks (CNNs) and Multilayer Perceptrons (MLPs). Classical algorithms are computationally light and require little to no floating-point hardware, but their representational capacity for complex, high-dimensional, or spatio-temporal signals is limited [3]. Deep learning models, once considered infeasible for kilobyte-scale devices, have become viable through model compression techniques such as pruning, quantization, and knowledge distillation, alongside microcontroller-aware architecture search [4], [5]. Yet the resource cost of these techniques additional flash storage, larger activation buffers, and longer inference

times is not uniformly justified across all application domains.

Despite a growing body of individual TinyML deployment studies, there remains limited synthesis of how classical and deep learning approaches compare directly on the axes that matter most to embedded practitioners: inference latency, memory footprint, and predictive accuracy, disaggregated by data modality (tabular, acoustic, and vision). This paper addresses that gap. Building on a structured review of experimental TinyML literature, we (i) characterize the resource profile of classical machine learning and quantized deep learning approaches across representative task types, (ii) synthesize reported performance trade-offs from peer-reviewed on-device experiments, and (iii) propose a lightweight decision matrix intended to guide algorithm selection under explicit hardware constraints.

II. RELATED WORK

TinyML has been the subject of a rapidly growing survey literature since roughly 2019 [6]. Sanchez-Iborra and Skarmeta [7] were among the first to frame TinyML-enabled devices as "frugal smart objects," outlining the challenges and opportunities of embedding inference in constrained wireless nodes. Subsequent surveys expanded this foundation along different axes: Ray [8] provided a broad state-of-the-art overview and outlined open research prospects; Capogrosso et al. [9] centered their review on the machine learning algorithms used across TinyML deployments; Tsoukas et al. [3] classified model optimization techniques and their application domains; and Schizas et al. [10] examined TinyML specifically in the context of large-scale, ultra-low-power IoT deployments.

More recent work has moved toward quantitative synthesis. Heydari and Mahmoud [1] conducted a PRISMA-guided systematic review of 26 experimental TinyML studies spanning healthcare, ecology, and vehicular detection, extracting reported accuracy, latency, and resource-usage figures for each deployment. Their review found that TinyML solutions typically achieve inference latencies far below the 10–500 ms range associated with cloud-based IoT and that reported accuracies across the reviewed studies ranged from roughly 72% to over 99%, depending strongly on task complexity and sensor data quality. Han, Trimi, and Lee [6] complemented this qualitative synthesis with the first bibliometric analysis of the TinyML literature, drawing on 392 Web of Science publications from 2020–2024; their keyword co-occurrence analysis confirmed "microcontroller," "edge computing," and "embedded systems" as core and rapidly emerging themes, while their historiographic mapping identified foundational works including Sanchez-Iborra [7] and the FANN-on-MCU toolkit as the most cited contributions shaping the field.

A parallel and increasingly distinct thread of research addresses Tiny Deep Learning (TinyDL) as a specialized subset of TinyML concerned specifically with compressing and deploying deep neural architectures on MCU-class hardware [4]. Somvanshi et al. [4] trace this evolution, cataloguing lightweight architectures such as MobileNet, SqueezeNet, and MCUNet, and documenting how quantization-aware training, structured pruning, and neural architecture search (NAS) have compressed ImageNet-capable models to under one megabyte. Their survey reports, for example, that MCUNet achieves 68.7% ImageNet top-1 accuracy in only 0.51 MB, and that hardware-aware pruning can reduce parameter counts by up to 13× with negligible accuracy loss. Introductory treatments of the field, including the preprint by Haldar and Jain [2], reiterate that the transition from cloud AI to Edge AI to TinyML reflects a progressive migration of intelligence toward the extreme edge, driven by the same latency, bandwidth, and privacy motivations identified in the foundational literature.

Notably, while individual experimental papers report on either classical algorithm (e.g., Support Vector Machines for water contamination classification [11] or Random Forests for seizure detection [12]) or deep architectures (e.g., compressed CNNs for arrhythmia or lung-disease classification [13], [14]), few studies place these two family's side by side under a common resource-constraint lens. Comparative evaluations of classical and deep learning techniques have also been reported in adjacent application domains outside TinyML proper. Taur, Vidhate, and Shevkari [21] compare classical and deep learning-based biometric identification techniques and similarly report a trade-off between model complexity and recognition accuracy; Shevkari, Bhosale, and Mule [22] review the comparable trade-off between lightweight and deep

learning-based approaches for cybersecurity threat detection under real-time operational constraints; and related work on deep-learning-based biometric recognition frameworks [23] echoes the general pattern that improved representational capacity in deep architectures comes at a measurable cost in computational and memory overhead. This paper's contribution is precisely that synthesis applied to the TinyML domain: an explicit, literature-grounded comparison of classical and deep learning approaches organized by the resource metrics that determine deployability on a given microcontroller.

III. TINYML HARDWARE AND RESOURCE CONSTRAINTS

TinyML platforms are overwhelmingly built around low-power microcontrollers, most commonly from the ARM Cortex-M family (M0, M4, M7), together with Wi-Fi-enabled MCUs such as the ESP32 [1], [4]. Typical devices operate at clock speeds of tens to a few hundred megahertz, with SRAM ranging from tens of kilobytes to roughly 512 KB and flash storage generally under one to two megabytes [1], [4]. Many devices lack a dedicated floating-point unit, which motivates integer-only (INT8) quantized inference. Power budgets are correspondingly tight: reported TinyML inference operations typically consume between 25 and 300 mW, compared to tens or hundreds of watts for cloud-hosted inference [1].

These constraints jointly determine which algorithm families are viable for a given deployment. Because SRAM must simultaneously hold the model's working memory (activations, intermediate buffers) and any sensor-side preprocessing state, algorithms with small, fixed-size decision structures such as shallow Decision Trees impose markedly less peak memory pressure than deep networks with multiple convolutional or fully connected layers.

Flash storage constraints instead bound the total number of stored parameters (tree nodes, support vectors, or network weights), which favors compact representations regardless of algorithm family. It is this joint SRAM/Flash/power budget, evaluated against task-specific accuracy requirements, that motivates the comparative framework developed in Section 5.

IV. ALGORITHM FAMILIES UNDER COMPARISON

4.1. Classical Machine Learning

Decision Trees offer a naturally compact inference structure: once trained, a tree is simply a sequence of threshold comparisons that can be encoded as nested conditional statements, requiring no matrix multiplication and only a handful of bytes of working memory. Random Forests extend this structure by aggregating the predictions of many trees, improving robustness and accuracy at a proportional cost in flash storage and inference time, but without introducing floating-point-heavy operations. Random Forests have been used successfully in TinyML deployments for tasks such as seizure detection from wearable EEG signals, where two independent studies achieved detection accuracies of 90–99% while running entirely on ARM Cortex-M and multicore wearable hardware [12], [15]. Support Vector Machines (SVMs), meanwhile, are frequently selected in the TinyML literature specifically for their effectiveness on small, non-linearly separable datasets; Ogore et al. [11] deployed an SVM classifier for offline cholera-contamination prediction from water physicochemical parameters, achieving 94% accuracy with only 1.6 KB of RAM and 15 KB of flash on an ARM Cortex-M4 device, with an output latency of roughly 1 ms.

4.2. Quantized Deep Learning

Deep learning approaches for TinyML center on compact CNN and MLP architectures whose parameters are represented in reduced precision, typically 8-bit integers rather than 32-bit floating point, through post-training quantization or quantization-aware training [4], [16]. Structural compression techniques pruning, knowledge distillation, and neural architecture search further shrink these models to fit kilobyte-scale flash budgets [4]. Reported deployments illustrate the resulting trade-offs: Faraone and Delgado-Gonzalo [13] deployed a CNN using the CMSIS-NN inference library on an ARM Cortex-M4-based wearable to classify cardiac arrhythmias, achieving 87% accuracy with a 210 KB model and a 95 ms inference latency; Abadade et al. [14] achieved 96–97% accuracy classifying respiratory sounds using a custom CNN trained with TensorFlow Lite Micro, requiring only 12 KB of RAM and roughly 250 KB of flash, with a 127

ms inference time on an Arduino Nano MCU. At the more extreme end of compression, Lin et al.'s MCUNet [4] demonstrates that a jointly optimized network architecture and inference engine can reach 68.7% ImageNet-scale classification accuracy within a 0.51 MB footprint, illustrating how far deep architectures can be compressed while retaining meaningful representational capacity for vision tasks.

V. COMPARATIVE ANALYSIS

To characterize the trade-offs between the two algorithm families, we synthesize reported performance metrics from peer-reviewed, on-device TinyML experiments, organized by data modality. Tables 1 and 2 summarize representative studies for classical machine learning and quantized deep learning, respectively, drawn from the systematic review of Heydari and Mahmoud [1] and the architecture survey of Somvanshi et al. [4]. All figures are as reported in the cited source publications on the specified hardware.

Table 1. Representative TinyML Deployments Using Classical Machine Learning

Study	Algorithm	Task / Modality	Accuracy	Latency	Memory Footprint
Ogore et al. [11]	Support Vector Machine	Water-quality classification (tabular)	94%	~1 ms	1.6 KB RAM / 15 KB Flash
Zanetti et al. [15]	Random Forest	Seizure detection (physiological / tabular)	>90%	—	STM32L476 (Cortex-M4)
Ingolfsson et al. [12]	Random Forest	Epilepsy monitoring (wearable ExG)	94–99% specificity	—	Multicore wearable device
Trivedi & Shroff [17]	Feature-based classifier	Mosquito species ID (acoustic)	88.3%	337 ms	9.2 KB RAM / 43.4 KB Flash
Bruno et al. [18]	Shallow ANN / classical hybrid	Gas classification (tabular sensor array)	72%	—	STM32 MCU

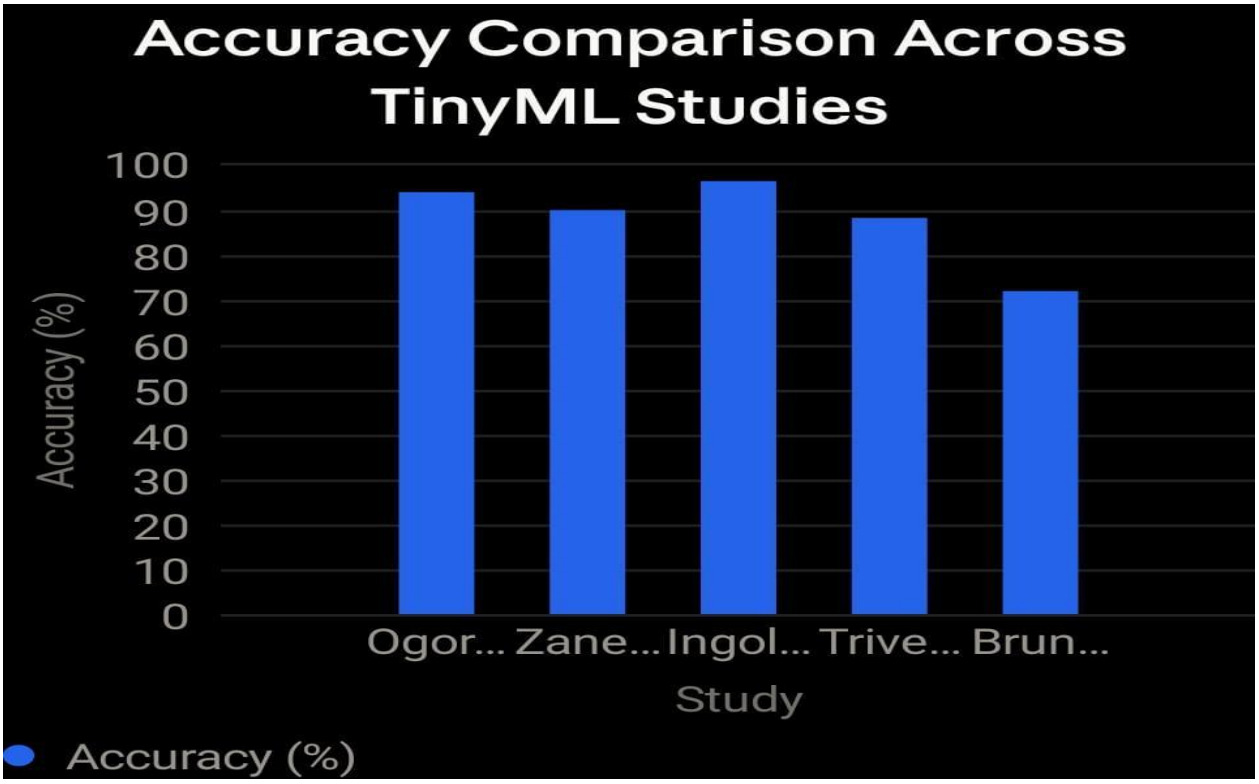
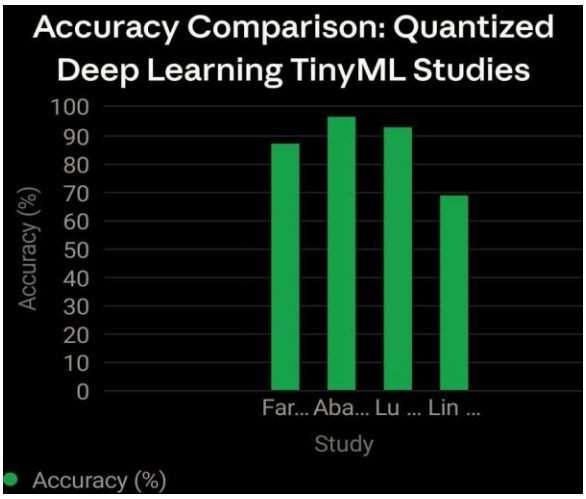
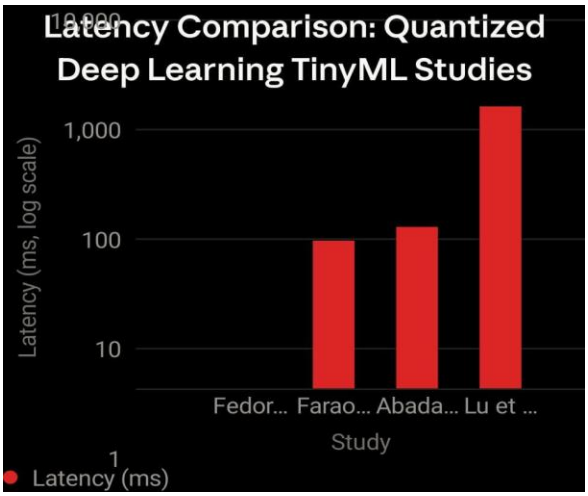


Table 2. Representative TinyML Deployments Using Quantized Deep Learning

Study	Architecture	Task / Modality	Accuracy	Latency	Memory Footprint
Fedorov et al. [19]	Pruned RNN	Speech enhancement (acoustic)	—	4.26 ms	STM32 (pruned 47%)
Faraone & Delgado-Gonzalo [13]	CNN (CMSIS-NN)	ECG arrhythmia classification (tabular/temporal)	87%	95 ms	210 KB output binary
Abadade et al. [14]	Custom CNN (TFLite Micro)	Lung disease classification (acoustic)	96–97%	127 ms	12 KB RAM / ~250 KB Flash
Lu et al. [20]	4-layer CNN	Colorectal polyp detection (vision)	92.8%	1.6 s	2.5488 mW at 8 MHz
Lin et al. (MCUNet) [4]	NAS-optimized CNN	Image classification, ImageNet-scale (vision)	68.7%	—	0.51 MB





Two consistent patterns emerge from this synthesis. First, on structured, low-dimensional tabular data sensor readings, physiological summary statistics, or physicochemical measurements classical algorithms achieve competitive or superior accuracy while consuming an order of magnitude less RAM and flash than even the most compressed deep networks, and with inference latencies in the single-digit millisecond range [11], [15]. Second, on tasks involving raw acoustic waveforms or image data, where relevant features are spatially or temporally distributed and not easily hand-engineered, quantized CNNs achieve higher accuracy than classical baselines reported for comparable tasks (for example, 96–97% for CNN-based lung sound classification [14] versus 72–88% for classical or shallow-network approaches on comparably complex acoustic classification tasks [17], [18]), but at the cost of markedly larger flash allocations (hundreds of kilobytes) and longer, though still real-time-capable, inference latencies.

This pattern is consistent with the broader TinyDL literature: Somvanshi et al. [4] note that classical and lightly compressed models typically remain under 250 KB and rely on post-training quantization or manual feature engineering, whereas TinyDL models spanning hundreds of kilobytes to just under one megabyte require quantization-aware training, neural architecture search, and hardware-aware quantization to fit MCU budgets, unlocking use cases speech, natural language processing, object detection that are largely inaccessible to classical algorithms operating on raw, unengineered sensor streams.

VI. A LIGHTWEIGHT DECISION MATRIX FOR MODEL SELECTION

Building on the trade-offs identified in Section 5, we propose a lightweight decision matrix (Table 3) intended as a practical starting point for TinyML

practitioners choosing between classical and deep learning approaches under an explicit hardware and application profile. The matrix is derived from the resource and accuracy patterns reported across the studies synthesized above rather than from new benchmark experiments, and should be treated as a heuristic screening tool rather than a substitute for task-specific evaluation.

Table 3. Proposed Decision Matrix for Algorithm Selection

Constraint / Application Profile	Favored Approach	Rationale
RAM budget below ~32 KB; tabular or low-dimensional sensor data	Decision Tree or SVM	Sub-kilobyte to low-kilobyte RAM footprint; microsecond-to-millisecond inference; minimal or no floating-point requirement [11]
Moderate RAM (32–128 KB); structured or lightly engineered features (e.g., statistical summaries of IMU/EEG signals)	Random Forest	Higher accuracy and robustness than a single tree at modest incremental memory cost; strong reported results on physiological time-series [12], [15]
Raw acoustic waveform input; latency tolerance in the tens-to-hundreds of milliseconds	Quantized CNN / compact RNN	Learned feature extraction outperforms hand-engineered features on complex acoustic patterns [14], [19]
Vision tasks (image classification, object/anomaly detection) with Flash budget of several hundred kilobytes to ~1 MB	NAS-optimized quantized CNN (e.g., MCUNet-style)	Spatial feature hierarchies required for competitive accuracy are not accessible to classical algorithms without heavy manual feature engineering [4]
Extremely tight power budget (sub-milliwatt, battery/energy-harvesting operation) with tabular data	Decision Tree / SVM	Lower compute per inference translates directly to lower average power draw at equivalent duty cycle [1], [11]
Frequent retraining or on-device adaptation required	Classical ML (Decision Tree / Random Forest)	Simpler training procedures and lower memory overhead make on-device or incremental updates more tractable on MCU-class hardware [3]

In practice, the boundary between these regimes is not sharp, and hybrid pipelines for example, applying classical feature extraction ahead of a small dense classifier, or using cooperative inference where a device performs feature extraction while a compact classifier runs downstream are common in the reviewed literature and can offer intermediate trade-offs between the two extremes [1].

VII. DISCUSSION

The comparative synthesis presented here reinforces a recurring theme in the TinyML literature: there is no universally optimal algorithm family, and the appropriate choice is governed jointly by data modality, hardware budget, and application-specific latency and power requirements [3], [8]. Classical machine learning algorithms remain highly competitive, and often preferable, for structured, low-dimensional sensing tasks where their minimal memory footprint and near-instantaneous inference directly translate into longer battery life and lower device cost. Deep learning approaches justify their additional resource cost primarily when the input modality (raw audio or imagery) contains structure that is difficult to capture through manual feature engineering, a finding consistent with the broader observation that TinyDL's main contribution is extending machine learning's reach to modalities that classical, feature-engineered pipelines handle poorly [4].

A second observation concerns benchmarking heterogeneity. Consistent with the limitations noted by Heydari and Mahmoud [1] and the bibliometric findings of Han et al. [6], the studies synthesized in this paper report accuracy using inconsistent metrics (precision, recall, F1-score, or raw accuracy) and were evaluated on non-standardized hardware and datasets. This heterogeneity limits the precision of any cross-

study comparison, including the one presented here, and motivates continued adoption of standardized benchmarking suites such as MLPerf Tiny, which report matched latency, energy, and accuracy figures across the same reference tasks and models [1], [4].

VIII. LIMITATIONS AND FUTURE WORK

This study is a literature-grounded comparative synthesis rather than a controlled, matched-hardware benchmark; the figures summarized in Tables 1 and 2 were collected under differing MCU platforms, datasets, and evaluation protocols across the cited studies, which constrains the strength of any direct numerical comparison. Future work should extend this analysis with matched, in-house benchmark experiments training the same classical and deep learning models on identical datasets and deploying them to the same representative microcontroller platforms (e.g., ARM Cortex-M4 and ESP32) using a standardized harness such as MLPerf Tiny, so that latency, peak SRAM usage, flash footprint, and accuracy can be compared under controlled conditions. Additional directions include extending the decision matrix to incorporate energy-per-inference as an explicit axis, evaluating hybrid classical/deep pipelines (e.g., on-device feature extraction feeding a compact classifier), and assessing how on-device learning and federated updates alter the relative resource cost of each algorithm family over a device's operational lifetime [4], [6].

IX. CONCLUSION

This paper synthesized reported experimental results from the TinyML literature to compare classical machine learning algorithms (Decision Trees, Random Forests, and Support Vector Machines) against quantized deep learning architectures (compact CNNs and MLPs) for deployment on resource-constrained microcontrollers.

The synthesis indicates that classical algorithms retain a clear advantage in memory footprint and inference latency for structured, tabular sensing tasks, while quantized deep networks deliver superior accuracy on raw acoustic and visual data at the cost of substantially higher flash and RAM requirements. Based on these trade-offs, we proposed a lightweight decision matrix intended to help practitioners select an appropriate algorithm family given explicit hardware budgets and application requirements. As the TinyML and TinyDL ecosystems continue to mature, standardized, matched-hardware benchmarking will be essential to refine and validate this class of decision-support tools.

REFERENCES

[1] S. Heydari and Q. H. Mahmoud, “Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions,” *Sensors*, vol. 25, no. 10, p. 3191, 2025.

[2] C. Haldar and R. Jain, “Introduction to TinyML: The New Era of Low-Power AI for IoT Devices,” Preprints.org, 2026, doi: 10.20944/preprints202606. 1102.v1.

[3] V. Tsoukas, A. Gkogkidis, E. Boumpa, and A. Kakarountas, “A Review on the Emerging Technology of TinyML,” ACM Computing Surveys, vol. 56, no. 10, Art. no. 259, 2024.

[4] S. Somvanshi, M. M. Islam, G. Chhetri, R. Chakraborty, M. S. Mimi, S. A. Shuvo, K. S. Islam, S. Javed, S. A. Rafat, A. Dutta, and S. Das, “From Tiny Machine Learning to Tiny Deep Learning: A Survey,” ACM Computing Surveys, vol. 58, no. 7, Art. no. 168, 2025.

[5] J. Lin, W.-M. Chen, Y. Lin, C. Gan, and S. Han, “MCUNet: Tiny Deep Learning on IoT Devices,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 11711–11722.

[6] 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.

[7] H. Chinni, Y. K. Sharma, S. Shevkari, J. L. Vydehi, Saurabh, and M. Kavitha, “Comparative Evaluation of Custom CNN Architectures and Transfer Learning Models for Image Classification in PyTorch,” in Proc. 2026 13th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2026, pp. 1–6, doi: 10.23919/INDIACom70271.2026.11525435.

[8] S. Devikar, M. Hinge, and S. S. Shevkari, “Human vs. machine: A review of AI’s impact on language learning interaction or replacing human interaction,” International Journal for

Multidisciplinary Research, vol. 7, no. 3, 2025, doi: 10.36948/ijfmr. 2025.v07i03.49459.

[9] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, and M. Cristani, “A Machine Learning-Oriented Survey on Tiny Machine Learning,” IEEE Access, vol. 12, pp. 23406–23426, 2024.

[10] S. S. Shevkari, D. A. Thorat, and S. V. Lohote, “Biometric intelligence: Improving recognition through deep learning frameworks,” 2025.

[11] M. M. Ogore, K. Nkurikiyeyezu, and J. Nsenga, “Offline Prediction of Cholera in Rural Communal Tap Waters Using Edge AI Inference,” in Proc. IEEE GLOBECOM Workshops (GC Wkshps), Madrid, Spain, 2021.

[12] T. M. Ingolfsson, A. Cossettini, X. Wang, E. Tabanelli, G. Tagliavini, P. Ryvlin, L. Benini, and S. Benatti, “Towards Long-term Non-invasive Monitoring for Epilepsy via Wearable EEG Devices,” in Proc. IEEE Biomedical Circuits and Systems Conference (BioCAS), 2021.

[13] S. Shevkari, P. S. Bhosale, and G. A. Mule, “Machine learning in cybersecurity: Cutting-edge approaches to threat detection and protection,” International Journal of Research Publication and Reviews, vol. 6, no. 11, pp. 7090–7092, 2025.

[14] G. N. Taur, M. M. Vidhate, and S. S. Shevkari, “Deep learning meets biometrics: A comparative study of modern identification techniques,” International Journal of Innovative Research in Technology, vol. 12, no. 2, pp. 545–552, 2025.

[15] I. Fedorov, M. Stamenovic, C. Jensen, L.-C. Yang, A. Mandell, Y. Gan, M. Mattina, and P. N. Whatmough, “TinyLSTMs: Efficient Neural Speech Enhancement for Hearing Aids,” in Proc. INTERSPEECH, Shanghai, China, 2020.

[16] S. Shevkari and A. Kelapati, “Multimodal Biometric Identification Using Yolov7 with Integrated Face, Palmprint, Fingerprint, And Iris Features,” Geomatics and Information Science of Wuhan University, vol. 51, no. 3, 2026, doi: 10.5281/zenodo.21523889.