TinyML: Tools, Applications, Challenges, and Future Research Directions

Rakhee Kallimani[1], Krishna Pai[2], Prasoon Raghuwanshi[3] Sridhar Iyer[4], and Onel L. A. López[5]
[1]Department of Electrical and Electronics Engineering,
KLE Technological University Dr MSSCET, Belagavi, Karnataka, India - 590008.
Email: rakhee.kallimani@klescet.ac.in
[2]Department of Electronics and Communication Engineering,
KLE Technological University Dr. MSSCET, Belagavi, Karnataka, India - 590008.
Email: krishnapai271999@gmail.com
[3]Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland- 90014.
Email: Prasoon.Raghuwanshi@oulu.fi
[4]Department of Artificial Intelligence,
KLE Technological University Dr. MSSCET, Belagavi, Karnataka, India - 590008.
Email: sridhariyer1983@klescet.ac.in
[5]Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland- 90014.
Email: onel.alcarazlopez@oulu.fi

In recent years, Artificial Intelligence (AI) and Machine learning (ML) have gained significant interest from both, industry and academia. Notably, conventional ML techniques require enormous amounts of power to meet the desired accuracy, which has limited their use mainly to high-capability devices such as network nodes. However, with many advancements in technologies such as the Internet of Things (IoT) and edge computing, it is desirable to incorporate ML techniques into resource-constrained embedded devices for distributed and ubiquitous intelligence. This has motivated the emergence of the TinyML paradigm which is an embedded ML technique that enables ML applications on multiple cheap, resource- and power-constrained devices. However, during this transition towards appropriate implementation of the TinyML technology, multiple challenges such as processing capacity optimisation, improved reliability, and maintenance of learning models' accuracy require timely solutions. In this article, various avenues available for TinyML implementation are reviewed. Firstly, a background of TinyML is provided, followed by detailed discussions on various tools supporting TinyML. Then, state-of-art applications of TinyML using advanced technologies are detailed. Lastly, various research challenges and future directions are identified.

***Index Terms*—TinyML, embedded AI, edge computing, IoT.**

I. Introduction

THE Internet of Things (IoT) leverages edge computing to enable the seamless processing of data from millions of interconnected sensors and other devices. The IoT devices are deployable at the network edge and incur a very low memory footprint and processing capacity [1], [2]. The IoT ecosystems increasingly depend on the edge platforms to collect and transmit the data [3]. In fact, within the IoT ecosystem, edge devices gather sensor data, which is then transmitted to a remote cloud or a nearby location for processing [4]. The edge computing technology performs computations and stores original data, simultaneously providing the infrastructure to support distributed computing [5], [6]. Additionally, edge computing provides (i) effective privacy, security, and reliability to end-users, (ii) lower delay, (iii) higher throughput, and availability and effective response to services and applications [7], [8]. Notably, by using a collaborative technique among the sensors, the edge devices, and the cloud, data processing may be conducted (at least partially) at the network edge rather than at the cloud. This may facilitate quality data management, effective service delivery, data persistence, and content caching [9]. Further, for implementation in various applications such as human-to-machine (H2M) interaction and smart healthcare, edge computing provides an opportunity to significantly improve the network services which are automated simultaneously ensuring that the network back-haul is less burdened [10], [11].

Recent research on IoT edge computing is in the spotlight as it facilitates the implementation of Machine Learning (ML) techniques in many use cases. However, hardware to be deployed at the network edge is severely constrained in resources such as memory capacity, power consumption, and compatibility, limiting the provisioning of high-end complex services [11]. In fact, the current IoT edge scenario is undermined as it does not exactly depict the envisioned cloud-to-embedded paradigm [12]. In effect, edge computing, though expected to do so in the future, does not yet offer significant power saving and high transmission capacity [13]. This is mainly due to the existing differences between hardware and software technologies, which lead to heterogeneous systems. Therefore, holistic and harmonious infrastructures are required, especially for training, updating, and deploying the ML models [14], [15]. Also, the architectures designed for embedded systems depend on the type of hardware and software, which in turn represents

Corresponding author: K. Pai (email: krishnapai271999@gmail.com).

a hindrance to developing a standard ML architecture for all edge IoT networks. Additionally, majority existing processors which are embedded permit only processing of sensor data in a generalized manner and applications of the software.

Currently, the large amount of data generated by multiple devices is sent to the cloud for processing due to the computationally intensive nature of existing network implementations. Indeed, operating advanced ML models such as deep neural networks (DNNs) and deep learning (DL) demand graphics processing units (GPUs) and dedicated hardware application specific integrated circuits (ASICs), which require large energy amounts and capacity of memory. Hence, there is currently significant interest in optimizing ML algorithms to make them more energy-efficient. Concurrently, there is also a growing demand to miniaturize low power embedded devices. These aspects have paved the way for the introduction of Tiny Machine Learning (TinyML), which implements ML algorithms on tiny devices such as edge IoT devices. TinyML enables signal processing at these devices while provisioning embedded intelligence, thus constituting a paradigm shift from cloud intelligence. Indeed, TinyML is a rapidly evolving edge computing concept which links ML and embedded systems [16]–[18]. All in all, TinyML may enable ultra low power and cost systems demonstrating efficiency and privacy [19]. Further, in cases of inadequate connectivity, TinyML may provide on-premise analytic(s), undoubtedly appealing for IoT services.

A. Article Motivation

The research on TinyML is in the early stages and requires appropriate alignments to accommodate the existing edge IoT frameworks [12]. Although initial research has demonstrated that TinyML is key to the development of smart IoT applications, several questions remain unanswered and solutions to these timely issues are required. The key issues which require immediate attention include:

- Key requirements and applications of TinyML.
- Capability of TinyML to implement DNNs at the edge.
- Power consumption versus accuracy trade-offs appropriate for and attainable by TinyML.

However, continued progress has been limited by lack of widely accepted benchmarks for TinyML-enabled systems. Specifically, bench-marking will allow measuring and thereby systematically comparing, evaluating, and improving the system's performance. This is essential to the progress of TinyML research. All in all, there are still multiple gaps in TinyML research, requiring timely solutions. This survey article lists several open research questions, outlines the potential obstacles in research on TinyML, and suggests possible directions for efficient solutions.

B. Our Contribution

Although an extensive body of literature has discussed TinyML aspects, aspects related to supporting tools and applications are not often seriously addressed. In this article, we detail the main applications motivating TinyML research and list the corresponding supporting tools. We then detail the key TinyML enablers and advances while performing a state-of-art survey. Lastly, several relevant research challenges are presented followed by corresponding research directions.

The main contributions of this article are as follows:

- Presenting an intuitive understanding of TinyML and providing detailed insights regarding the related fundamentals.
- Detailing the existing TinyML technology supporting tool-sets that are used for training the model to be deployed at edge.
- Discussing the key TinyML enablers and multiple use-cases/applications of TinyML.
- Presenting and discussing various current and future challenges in research, and related practical solutions with an outlook towards furthering research on TinyML.

Table I compares the contribution of this survey article with respect to the most recent articles on TinyML.

C. Article Outline

The rest of the paper is structured as follows. **Section II** overviews the TinyML technology and details the various tools that support TinyML. enablers. In **Section III**, we detail the multiple state-of-art applications of TinyML enabled by advanced technologies. **Section IV** identifies the various challenges and also proposes future research directions. Finally, **Section V** concludes the survey.

Figure. 1 shows the devised taxonomy which represents the survey on TinyML presented in this article.

II. TinyML: Overview and Related Tools

TinyML can be seen as an ML tool/technique with the capability to perform on-device analytic(s) for multiple sensing modalities such as vision, audio, and speech. TinyML incurs very low power/energy consumption, thus suitable for embedded edge devices that are battery operated. Further, TinyML is appropriate to be implemented for large-scale applications within the IoT network framework [16], [26].

Currently, cloud-enabled ML systems suffer a number of difficulties, including high power consumption and security, privacy, dependability, and latency issues. As a result, pre-installed models on hardware-software platforms are currently implemented (e.g., edge impulse) [27]. Raw data, which simulates the physical world, is gathered by sensors and subsequently processed at a CPU/microprocessor unit (MPU). The MPU aids in catering to the ML-aware analytic support enabled by specific edge aware ML networks. Notice that edge ML communicates with any remote cloud ML for transfer of knowledge. The incorporation of TinyML into system will make the physical world significantly smarter compared to any current scenario [28]. Indeed, such a system can help edge devices to undertake key decisions even without assistance from edge AI or cloud AI. Notably, the system performance may improve over various fronts such as energy efficiency, effective data privacy, and delay.

All in all, TinyML is envisioned as an amalgamation of hardware, software, and algorithms. Concerning hardware, IoT

TABLE I
SUMMARY OF RECENT SURVEYS RELATED TO TINYML.

Reference	Key Contribution	Limitations w.r.t our survey
[12]	• Intuitive review regarding the possibilities for TinyML. • Background and toolsets to support TinyML. • Key enablers to improve TinyML systems and state-of-art architectures for TinyML. • Key challenges and future road-map to mitigate numerous research issues of TinyML.	• Recent case studies of TinyML. • Detailed directions to highlight key research findings.
[19]	• Overview and review of TinyML studies. • Analysis of ML models types used for TinyML. • Details of datasets and devices types and characteristics. • Available resource constraints such as hardware platforms, and supporting platforms.	• Specific application.
[20]	• Definition of TinyML. • Background information on various related technologies. • TinyML as a service. • Role of 5G for TinyML IoT. • Recent progress in TinyML research. • Future oppportunities and challenges.	• Need for standardization.
[21]	• Cross-layer design flow as a key aim of TinyML. • TinyML applications, frameworks and benchmarking. • TinyML as a service by implementing it via efficient hardware and software design. • Challenges, opportunities, and vision for the road ahead on TinyML.	• Supporting platforms and available library/framework for specific applications.
[22]	• Resource optimization challenges of TinyML. • Present state of TinyML frameworks, libraries, development environments, and tools. • Benchmarking of TinyML devices. • Emerging techniques and approaches to boost/expand TinyML process, and improve data privacy and security. • Future development of TinyML.	• Specific state-of-art use-cases/applications.
[23]	• Systematic review of TinyML research. • Relevant TinyML literature on hardware, framework, data sets, use cases, and algorithms/models. • Roadmap to understand literature on TinyML.	• Existing challenges concerning multiple constraints, and future directions for research on TinyML.
[24]	• Review of the contribution of TinyML in healthcare applications at the edge. • Requirement of integration of ML followed by generated solutions. • Optimization of Neural Networks by TinyML.	• Use-case related to healthcare.
[25]	• Current landscape of TinyML. • Challenges and directions to develop fair and useful hardware benchmark for TinyML workloads. • Benchmarks and selection methodology.	• Possible solutions to future research challenges.
Our Survey	• Supporting software/library and platforms and associated targeted applications. • Key existing challenges with respect to different constraints and possible directions.	–

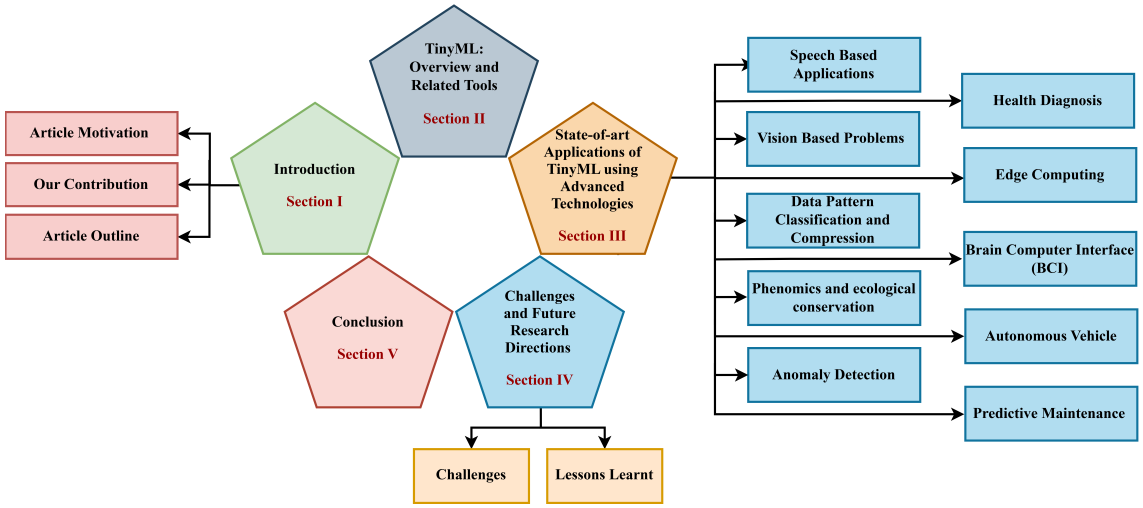

Fig. 1. Taxonomy of the survey in this article.

devices, which may or may not comprise hardware accelerators, may require analog and memory computing to provide an effective learning experience. Regarding software, TinyML applications can be implemented over cloud-enabled software or on varieties of platforms such as Linux/embedded Linux, etc. Lastly, Tiny ML systems must be supported by new algorithms requiring exceptionally low memory-sized models to avoid excessive memory consumption. Overall, the TinyML systems must optimize ML with a compact design of software in the presence of high-quality data. Then, this data must be flashed via binary files generated through the models which have been trained over a much larger machine [29], [30].

Additionally, compact software is required for small power consumption supporting TinyML implementation. Hence, systems enabled by TinyML must operate under rigid constraints while still providing high accuracy. In many cases, TinyML may rely on energy harvesting at the edge devices to support its operation and/or enable battery-operated embedded edge devices. TinyML's fundamental requirements include (a) providing scalability to billions of cheap embedded devices, and (b) storing codes within a limited few KBs over on device RAM [27], [31]. Lastly, it has been demonstrated that TinyML

can also be used within a standard pipeline over edge which could be changed when required by cross-section data [32].

TinyML frameworks are continuously under development by multiple industries, specific developers, and research groups. The study conducted by [33] shows that many of the architectures are available to the public with the exception of Fraunhofer IMS and Cartesiam-developed AlfES and NanoEdge AI Studio. Further, the majority of these frameworks support the ARM Cortex group whereas, others support the ESP8266, ESP32 group hardware, and few support Arduino and Raspberry Pi. It is evident that C and C++ are the most used languages via these architectures whereas, multiple external libraries such as TensorFlow Lie, TensorFlow, etc., can be used with the frameworks.

Several frameworks are currently being introduced by multiple research groups globally in view of implementing the TinyML models over resource-constrained devices. The study by [34] focuses on the deployment of TinyML models over edge devices to reduce the delay and improve privacy. The authors have presented a parallel ultra low power (PULP) architecture for implementation in IoT-enabled processors. PULP permits the running of non-neural ML kernels which demonstrates higher accuracy in comparison to the neural networks. PULP is compared to the PULP-OPEN hardware and it is demonstrated that PULP is 12.87x faster in comparison to ARM Cortex-M4 MCU.

An open-source toolkit namely, fast artificial neural network (FANN)-on-MCU, which runs on PULP, is developed for reducing energy consumption by [35]. The proposed toolkit is enabled via FANN library to run the architecture, implemented in the InfiniWolf prototype, over lightweight IoT-aware devices. The authors have compared FANN-on-MCU with RISC-V octa-core processor and have demonstrated that FANN-on-MCU incurs an increased speedup by 22x and consumes 69 percent less energy.

In [36], the authors have presented hls4ml TinyML architecture for reducing energy consumption. The proposed framework enables the acceleration of ML-aware FPGA and ASIC implementation in a feasible and easy manner and provides Python-based APIs to harness scientific benefits from the framework. Further, it also provides quantization and pruning-aware training for low-power embedded devices.

In [37], PhiNets, a scalable backbone architecture for DNNs is detailed which is designed for providing image processing application support for resource-constrained edge IoT devices. The proposed framework is developed over the inverted residual blocks to decouple cost, memory, and over-processing. The results demonstrated that PhiNets reduces the count of parameters by 85–90- in comparison to existing architectures.

In [38], to address the hardware/software co-design, the HANNAH framework is detailed which aims at automating co-optimization steps of a NN framework for efficient end-to-end DNN training and use over edge devices. HANNAH is implemented in 3 steps with the Ultra-Trail NN unit.

Following [19], each industry has unique software and ML model to make the embedded systems board compatible with TinyML applications. However, with TinyML this may present significant issues. TinyML is a technique for using extremely compact computer programs for tasks such as voice recognition and motion detection. However, it will be challenging to utilize TinyML over several devices without compromising on accuracy since each industry has its own software. Hence, it will be crucial to provide a standardized approach for implementing TinyML. This necessitates the development of a common framework that can run on various hardware manufactured by various industries. Once successful, this will ensure that multiple commonplace applications can be provisioned by TinyML.

From the above, it can be inferred that among the multiple resource constraints viz., data set generation, execution time, etc., hardware platforms present the key constraints for TinyML's high performance. Hence, there is a need to encourage the design and training of any TinyML model using specific software/library/framework and deploy the trained model to the supporting hardware platform.

In Table II, we list the details of the available hardware platforms supporting the design environment i.e., frameworks/libraries. Further, we also list the software/libraries which can be integrated with the related hardware platforms to provision specific application(s)/use-case(s). Our survey reveals that approximately (i) 59 % of participants use TensorFlow, (ii) 17 % of participants use PyTorch, and (iii) 24 % of participants use other frameworks. Further, the top 3 data types used by ML users are vision data, motion data, and sound data. In regard to the hardware boards, the most used for developing TinyML projects include (i) Raspberry Pi, (ii) Arduino Nano 33 BLE Sense, (iii) ESP32, and (iv) Raspberry Pi Pico and NVIDIA Jetson Nano.

III. State-of-art Applications of TinyML using Advanced Technologies

There are several applications of TinyML, including speech and vision-based applications, data pattern classification and compression, health diagnosis, edge computing, brain-control interface, autonomous vehicles, phenomics, and ecology monitoring. This section details the state-of-art applications of TinyML using various advanced technologies.

A. Speech-Based Applications

1) Speech Communications

Semantic communication emerged as an alternative to conventional communication. In the latter, all the data matters and is transmitted, while in the case of the former, only the context/meaning of the data is transmitted to the receiver. Notably, semantic communication can be implemented by employing the TinyML methodologies [51]. Speech detection and recognition, online teaching/learning, and goal-oriented communication as shown in Figure 2, are popular applications in the current scenario and require high data and high-power consumption on the host device. To overcome these drawbacks, TinySpeech library has been introduced to build a low computational architecture with a low storage facility using deep convolutional networks [19].

In view of Speech Enhancement [52], the authors addressed sizing of the speech enhancement model as it was subjected to

TABLE II
SUPPORTING PLATFORMS FOR TINYML.

Reference	Software/ Library / Framework	Developer	Supporting Platform	Hardware Platforms	Targeted Applications
[39]	TensorFlow Lite (TFL)	Google Brain Team	Android, iOS, Embedded Linux, Microcontrollers	Arduino Nano 33 BLESense, Sparkfun Edge, STM32F746 Discovery Kit , Adafruit Edgebadge, Adafruit TensorFlow Lite for Microcontrollers Kit, Adafruit Circuit Playground Bluefruit, Espressif ESP32-Devkitc, Espressif ESP-EYE, Wio Terminal: ATSAMD51, Himax WE-I Plus EVB Endpoint AI Development Board, Synopsys Designware ARC EM Software Development Platform, Sony Spresense	Image and Audio Classification, Object Detection, Pose Estimation, Speech and Gesture Recognition, Segmentation, Video Classification, Text Classification, Reinforcement Learning, On Device Training, Optical Character Recognition
[40]	Utensor	ARM	Android, iOS, Embedded Linux, Microcontrollers	Mbed, ST K64 ARM Boards	Image Classification, Gesture Recognition, Acoustic Detection and Motion Analysis
[41]	Edge Impulse	Zach Shelby And Jan Jongboom	Android, iOS, Embedded Linux, Microcontrollers	Arduino Nano 33 BLE Sense, Arduino Nicla Sense ME, Arduino Nicla Vision, Arduino Portenta H7 + Vision Shield, Espressif ESP32, Himax WE-I Plus, Nordic Semi Nrf52840 DK, Nordic Semi Nrf5340 DK	Asset Tracking and Monitoring, Human Interfaces, Predictive Maintenance
[42]	Nanoedge AI Studio	Cartesiam	Android, Linux	STM32 Boards	Anomaly Detection, Predictive Maintenance, Condition Monitoring, Asset Tracking, People Counting, Activity Recognition
[43]	Pytorch Mobile	Meta AI (Facebook)	Android, iOS, Linux CPU	NNAPI (Android), Coreml (iOS), Metal GPU (iOS), Vulkan (Android)	Computer Vision and Natural Language Processing
[44]	Embedded Learning Library (ELL)	Microsoft	Windows, Ubuntu Linux, Mac OS X	Raspberry Pi, Arduino, Micro: Bit	Image And Audio Classification
[45]	STM32Cube.AI	STMicroelectronics	Android, Linux	STM32 ARM CORTEX Boards	Anomaly Detection, Predictive Maintenance, Condition Monitoring, Asset Tracking People Counting, Activity Recognition
[46]–[48]	Autoflow	Daniel Konegen And Marcus Rüb	Android, iOS, Embedded Linux, Microcontrollers,	MCU, FPGA Boards, Raspberry Pi	Image Classification, Object Detection, Pose Estimation, Speech Recognition, Gesture Recognition
[49]	Apache Mxnet	Apache Software Foundation (ASF)	Linux	Raspberry Pi, NVIDIA Jetson	Image Classification, Object Detection, Pose Estimation, Speech and Gesture Recognition
[50]	ML Kit for Firebase	Google	Android, iOS	Mobile Devices	Facial Detection, Bar-Code Scanning, Object Detection

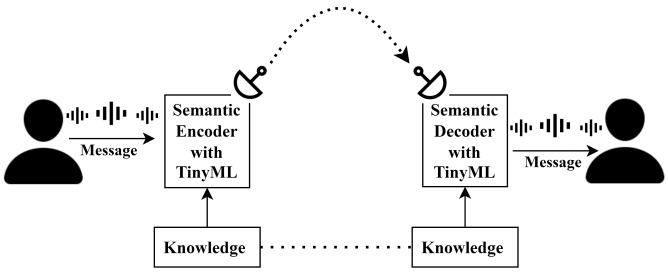

Fig. 2. Speech recognition and response.

hardware resource constraints. The study employed structured pruning and integer quantization for the Recurrent Neural Network (RNN) speech enhancement model. The results suggested reducing the model size by 11.9x and operations by 2.9x. The study also demonstrated the usefulness of hearing aid products enabled by neural speech enhancement methods with better battery life. Resources must be utilized efficiently for energy-constrained edge devices executing voice-recognition applications, as demonstrated by the authors in [53]. Therefore, the study aimed to partition the process and proposed a co-design for the TinyML based voice-recognition. The authors used windowing operation to partition hardware and software such that raw voice data would be pre-processed. The results demonstrated a decrease in energy consumption on the hardware. Lastly, the authors proposed optimized partitioning among hardware and software co-design as future scope.

Authors in [54] proposed a phone-based transducer for the speech recognition system. However, the study involved replacing the LSTM predictor with the conv1d layer for reducing the computations on the edge device. The results revealed that the Singular Value Decomposition (SVD) technology had successfully compressed the model. While the Weighted Finite State Transducers (WFST) based decoding allowed the flexible bias in the model improvement. A similar study on speech recognition systems was conducted in [55] by introducing integer quantization in the LSTM neural network topology to reduce memory consumption and computation latency. The results demonstrated that the proposed model achieves good accuracy even on a data set that consists of long utterances.

Live captioning, virtual assistants, and voice commands are all prominent applications of SR and all of them require ML for their work. The current SR technologies (such as Siri and Google Assistant) have to ping the cloud every time they receive data, which creates concerns regarding data security. The solution for this problem is to perform on-device SR, which is where TinyML comes into play. Authors in [56] proposed Tiny Transducer, an SR model for the on-device scenario, which uses a deep feed-forward sequential memory block (DFSMN) layer on the encoder side and one Conv1d layer on the predictor side in place of LSTM layers to bring down both network parameters and computation.

2) Hearing Aid (HA)

It is common that most people in their later half experience hearing loss. This health issue constitutes a serious problem for countries dealing with population aging, e.g., Japan, South Korea, and China, opening a business opportunity for the HA industry. However, a critical problem must still be resolved and it is that currently, HA devices amplify all of the input sounds, thus making it difficult for a person to distinguish the desired sound in a noisy environment. According to [57], TinyML can provide a solution to this problem. Therein, the authors proposed a TinyLSTM-based speech enhancement (SE) algorithm for HA devices, which performs the denoising operation over the input sounds and extracts the speech signal. When tested on STM32F746VE MCU and trained with the CHiME2 WSJ0 data-set [58], the algorithm shows a computational latency of $2.39ms$, which is far less than the $10ms$ target.

B. Vision-Based Applications

To process computer vision based data-sets, TinyML can play a crucial role as these processes need to be performed on the edge platform to generate faster outputs. Authors in [59] addressed the practical challenges in training the model using the OpenMV H7 micro-controller board. They proposed an architecture for detecting alphabets within American Sign Language over ARM Cortex-M7 microcontroller with only 496 KB of frame-buffer RAM. In effect, the authors addressed the major challenge of convolutional networks (CNNs) with high generalization error, including large test and training accuracy. However, these did not generalize effectively to images within new cases and backgrounds with noise. The authors employed interpolation augmentation; results show 98.80% accuracy in test and 74.59% accuracy in generalization. It was observed that interpolation augmentation reduced drop in accuracy during quantization in hand sign classification. However, it improved classification generalization (with a 185 KB post quantization model) and inference speed (to 20 fps). The authors also proposed the future scope to improve accuracy in generalization model training on data from highly varied sources and testing it over hardware to attain the ambition of portable watch-like device. We know that word level vocabulary and non manual features require identification of facial expressions, mouth, tongue, and body pose. By extending the study on CNN, the authors in [60] deployed CNN architecture on a resource-constrained device. They developed framework to detect medical face masks over resource constrained ARM Cortex M7 micro-controller using TensorFlow lite with extremely low memory footprints. The results demonstrated 138 KB model size post quantization and 30 frames per second inference speed on the targeted board. The authors also proposed the research scope on developing (i) quantization schemes for reducing the precision from float32 to int8, (ii) data sets with heterogeneous sources, and (iii) experiments with smaller precision networks.

Authors in [61] presented a case study aiming to design a gesture recognition device that could be clamped to an existing cane to be used by the visually impaired. The design constraints considered were low cost, accurate gesture detection, and battery design. Further, the data was collected using a gestures data set, and the ProtoNN model was trained with a classification algorithm. Finally, as a scope for future research, the authors mentioned the necessity of understanding gestures and their associated safety and the integration of android and developed devices. In [62], the authors addressed challenges, such as resource scarcity and on-board computation, faced in scaling the autonomous driving to mini-vehicles. The authors introduced a TinyCNN-based closed-loop learning flow and proposed an online predictor model which takes into account the recently captured image at the run-time. The major challenge observed in the design of autonomous driving was the decision model developed for offline data, which may not be robust for online data. For such applications, the authors stated that the model design should be able to adapt to real-time data, and this motivated the authors' current study. The authors performed experiments on GAP8, STM32L4, and NXP k64f. It was demonstrated through comparative results that GAP8 outperforms in terms of energy consumption and latency with online data. As a future study, CNN on-chip can be trained for continuous learning considering real-time applications.

The study on NAS was conducted by authors in [63], where NAS was implemented in image classification and object detection problems. The authors addressed the real-time challenges such as deploying architectures of synthesized CNN, and the hardware-aware NAS was proposed as a solution. In addition, the study involved a detailed survey of the challenges and limitations of existing approaches, and their categorization with respect to acceleration techniques, cost estimation of hardware, search space, and search strategy was also available.

C. Data Pattern Classification and Compression

The challenge of adapting a trained TinyML model to the online data has attracted attention from the research

community. The authors in [64] proposed a novel system, namely, TinyML with Online Learning (TinyOL), for introducing training with incremental online learning on MCU and enabling updating the model online on edge devices of IoT. The implementation was performed using C++ language, and an additional layer was added to the TinyOL. Further, the study was performed on the auto-encoder of Arduino Nano 33 BLE sense board [39], and the model was trained to classify new data patterns. The research scope mentioned included the design of efficient and optimized algorithms for the neural network to support online device training patterns. Authors in [65] mentioned the number of activation layers as a major issue for memory-constrained AI edge devices. As a result, Tiny-Transfer-Learning (TinyTL) was introduced to efficiently utilize the memory over an edge device and avoid using the intermediate layers as activation. In addition, to uphold the adaptation capabilities and allow the feature extractor to discover the small residual feature maps, a bias module known as the 'lite residual module' was also introduced. Compared to the full network fine-tuning, the results showed TinyTL reduced the memory overhead by 6.5x with a moderate accuracy loss. While compared to the case when the last layer was fine-tuned, TinyTL displayed a 34.1% of accuracy improvement once again with a moderate accuracy loss.

Authors conducted a detailed study on data compression in [66], emphasizing that data compression algorithms must manage extensive collected data in a portable device. The authors developed Tiny Anomaly Compressor (TAC) and demonstrated that TAC outperforms Swing Door Trending (SDT) and Discrete Cosine Transform (DCT) algorithms. Furthermore, TAC achieved a maximum compression rate of 98.33% and outperformed both SDT and DCT in terms of peak signal-to-noise ratio.

D. Health Diagnosis

With the spread of COVID-19, it is now required to continuously detect cough-related respiratory symptoms. The authors in [67] have presented a scalable CNN enabled model namely, Tiny RespNet which operates over multi-modal settings. These settings are deployed over Xilinx Artix-7 100 t FPGA which provides the parallel processing facility with low power consumption and high energy efficiency. Further, Tiny RespNet framework is able to input audio recordings, speech of patient, and information of demography to ensure classification. The cough detection related respiratory symptoms are classified via three data sets.

The authors in [68] conducted a study on deep learning computations on edge devices. The study aimed to implement a TinyML model named TinyDL on wearable devices for health diagnosis and carried out the performance accuracy analysis to reduce latency, bandwidth, and power consumption. A multi-layer LSTM model was designed and trained for a wearable device with the collected accelerated data as the input. The model exhibited an accuracy of 75-95% accuracy per gesture and was able to analyze only the off-device data. This limitation was due to compatibility issues of the framework, which authors corrected in [19] by providing a detailed study of TinyML's significance, and related role in IoT. The ML models on edge devices showed the potential solution to the existing challenges of IoT. The authors also provided a detailed review of models, device types, and the data sets used in TinyML, in addition to a detailed survey of the existing and supporting framework on platforms.

Another application includes the estimation of body pose which is vital to monitor health of the elderly. In [69], platform agnostic framework is proposed to enable validation and rapid fostering of model to platform adaptations. It implements face landmarks and body pose estimation algorithms and implements composite fields to detect spatiotemporal body pose in real time. The platform used is Nvidia Jetson NX consisting of GPUs and DL hardware accelerators.

E. Edge Computing

With the massive increase in IoT devices connecting to the global network, there is an urgent need for setting up edge devices to reduce the load on the cloud. These edge devices carry individual data centres capable of high-level computing, which results in high security, and reduced cloud dependency, latency, and bandwidth. The edge devices enriched with TinyML algorithms will help in fulfilling the power, memory, and computing time constraints as shown in Figure 3. In [70], the authors detailed the energy efficiency problems faced during the practical implementation of an edge Unmanned Aerial Vehicle (UAV) device. The goal was to implement an energy-efficient device with low latency by interfacing TinyML on the MCU, which acts as a host controller for the UAV. For various edge computing activities, there is a need for sensors for data acquisition. Edge sensors such as blood pressure sensors, accelerometers, glucose sensors, Electrocardiogram (ECG) sensors, motion sensors, and Electroencephalogram (EEG) sensors are widely used for the data gathering process during edge computing [71].

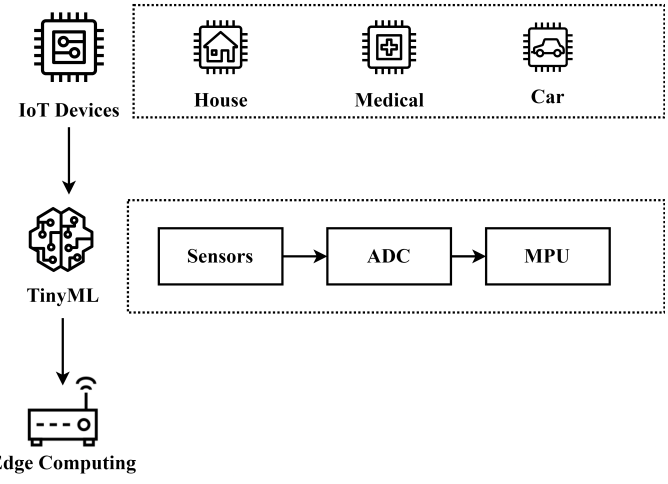

Fig. 3. Edge devices combined with TinyML for Edge Computing.

F. Brain-Computer Interface (BCI)

Within the healthcare sector, TinyML can contribute significantly; to, e.g., tumor and cancer detection, emotional

intelligence, and health predictions using EEG and ECG signals [72] as shown in Figure 4. With the aid of TinyML technology, Adaptive Deep Brain Stimulation (aDBS) [73] has the potential to demonstrate breakthroughs in successful clinical adaptations. aDBS helps in the identification of disease-specific bio marks and their respective symptoms through invasive recordings of the brain signals. Further, as healthcare mainly includes a collection of enormous data and then processing it to reach specific solutions for the early cure of the patient, it is necessary to build a system that is extremely accurate and highly secure. Such a system in the medical science field, when combined with IoT and TinyML, is termed as the Healthcare Internet of Things (H-IoT) [74]. The major applications of H-IoT are monitoring, diagnosis, spread control, logistics, and assistive systems. To detect a patient's health state remotely there is a need to develop a highly reliable system with extremely low latency and global accessibility. This system can be developed by integrating H-IoT with TinyML and 6G-enabled Internet services [75].

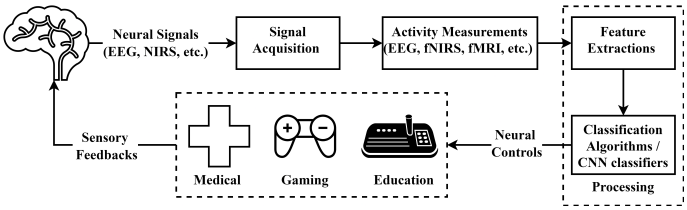

Fig. 4. BCI Methodology.

G. Autonomous Vehicle

Autonomous vehicles are utilized during multiple emergency cases such as military, human tracking, and industrial applications. Such vehicles demand smart navigation to ensure efficient identification of the object(s) being searched. Currently, autonomous driving is a complex task especially, when lower scale such as mini-vehicles are desired. The TinyML technology can be implemented to improve autonomous driving of the mini vehicles as shown in [76] where it has been deployed over the GAP8 MCI which is enabled by the framework of convolution neural network (CNN). Further, the same is tested over STM32L4 and NXP k64f platforms. The results demonstrated that such an integration reduces the processing delay by 13 times and simultaneously provides an improvement in consumed energy of approximately 90%. In addition, automatic traffic scheduling has been investigated in recent years to possibly integrate it with the TinyML technology in view of improving real time traffic system [77]. The proposed method includes piezoelectric sensors which are embedded over the multiple lanes of a specific road, and the two point time ratio technique is used for detecting vehicle by using data from piezo-sensors. Further, vehicle classification includes prediction of green light timings. The study implements the random forest regressor for predicting signal duration depending on the count of input vehicles over every lane. The implementation is over an Arduino Uno which is supported by the m2gen library using Scikit Learn requiring only 1.754 KB for the algorithm.

H. Phenomics and conservation of ecology

Phenomics is defined as study of phenotypes related to genome wide changes in an organism's lifespan. Specifically, among the entire germplasm set, plant phenomics utilizes improvements in genes to discriminate key germplasm. The study by [78] performed a phenomics image analysis which is based on tomato leaf disease and spider mites classification. The method utilizes Plant-Village tomato data set in conjunction with YOLO3 algorithm which is enabled by DarkNet-53 architecture to automatically detect tomato leaves. The study further implements SegNet algorithm to segment the images in a pixel-wise manner. Lastly, the study investigates multiple other commonly used data analysis tools for validating phenomics use with TinyML. In recent years, ecological conservation analytics enabled by AI techniques has witnessed tremendous growth. The study by [79] has deployed ecology conservation step using TinyML within small payload satellites (SmallSats). Specifically, study focuses on improvement of sea turtles' conservation using advanced real time vision based TinyML. Another crucial application of TinyML includes environment monitoring. In [80], a deep tiny NN is detailed for prediction of weather. The proposed framework utilizes STM32 MCU and X-CUBE-AI tool chain over the Miosix operating system. The framework requires 45.5 KB of flash and 480 Bytes onboard RAM for running deep tiny NN architecture.

I. Anomaly detection

Anomaly is defined as an event that varies from majority mass of events. In [81], an investigation is conducted for finding appropriateness of TinyML to detect anomalies of related tasks. A generic ANN, auto-encoder, and variational auto-encoder are used over Arduino Nano 33 BLE sense module, and top load washing machine Kenmore model is implemented to detect anomalies over unbalanced spin dry cycle. The results demonstrated approximately 90% precision and accuracy.

J. Predictive Maintenance

Predictive Maintenance has emerged as a promising maintenance paradigm in which models perform the prediction of equipment failure [82]. Currently, the majority of the predictive maintenance systems have been used either over cloud or via powerful computers. The data utilized by such models are mostly generated by tiny sensor devices and hence, the current approach requires data to be aggregated and transmitted over network for processing. TinyML can be used to implement an alternative to cloud-based predictive maintenance systems. This will require the optimization of the entire TinyML pipeline in an industrial setting which if achieved, will demonstrate immense potential for input optimization in view of achieving predictive maintenance using TinyML.

TABLE III
SUMMARY OF MULTIPLE EXISTING CHALLENGES ON TINYML RESEARCH AND SUGGESTED DIRECTIONS TO OBTAIN SOLUTIONS [12].

Sl. No.	Constraint	Existing Challenges	Proposed Directions
1.	Resource	Limited power availability at the edge devices is a critical challenge to maintaining the algorithm's accuracy.	Design the edge devices with a co-design approach to meet the power management challenges.
		Limited memory size is another challenge as deploying the model needs a higher peak of memory.	Quantise the model and use an appropriate converter to convert from float type to integer type and use the memory on edge hardware appropriately.
2.	Hardware	Existing benchmarks need to be re-framed before deploying TinyML on the resource constraint hardware	Redesign the benchmark to balance the resource constraint and data heterogeneity data of the system.
		Heterogeneity is a challenge as there are various embedded hardware available over a wide area of application; an extremely heterogeneous ecosystem demands a different type of micro-controllers and the model generated is expected to work efficiently on the targeted embedded hardware	Develop a generalized model so that it can work efficiently with heterogeneous systems
3.	Data-set	The architecture of TinyML systems do not support the existing data set. This is a challenge to all edge devices as the data collected from external sensors need resolution and the devices are energy and power constrained. Thus the existing data set cannot be used directly to train the TinyML models.	Train the TinyML model with a standard and optimized data set.
		The lack of popularly accepted Models is another challenge in the research domain	Develop a model which can be readily adopted by the system and improve the TinyML ecosystem
		The heterogeneous data type is a major concern, especially for data and Network Management	An intelligent network is needed as current data and networks are not managed for the data.
4.	Existing edge infrastructure	Edge computing infrastructure faces a challenge as the resources are changing dynamically, in turn affecting the TinyML ecosystem.	The need for enablers is key to providing support to system and leveraging the existing infrastructure
		Currently, the edge platform suffers from the issue with dynamic resource allocation due to which there is a need for techniques/ algorithms for the analysis of dynamic data.	Employing optimization techniques could be one of the directions to support dynamic edge resource allocation.
5.	Design of ML Models	The response time of ML models is another issue discussed in the research community widely.	Models need to be co-designed to provide good and quick responses for all the edge devices with model pruning and quantization being a part of model design.

IV. CHALLENGES AND FUTURE RESEARCH DIRECTIONS

In this section, we present the various challenges and issues in research related to the multiple TinyML applications. Specifically, we summarise key leanings on TinyML from the detailed discussions throughout the paper. Next, we have also detailed the possible solutions to provide future scope to the researchers for contributing towards providing solutions to the various issues and challenges.

A. Challenges

TinyML encounters major hurdles hindering the related growth pattern. The key challenges include the following:

- Currently, for the embedded edge IoT devices battery power consumption is expected to be over a period of ten years [83]. For e.g., in ideal conditions, a battery capacity of 2Ah is expected to have a life cycle of more than ten years considering that the power consumption is less than 12μA. However, when a simple edge IoT device's circuit is considered with a combination of MCU, temperature sensor, and a Wi-Fi Module, the aggregate current consumption is approximately 176.4 mA [84]. This will invariably reduce the life cycle of the 2Ah battery to approximately 11 hours and 20 minutes. This aspect is a critical challenge in the TinyML ecosystem.
- Majority of edge devices operate at clock speeds between 10–1000 MHz which is restrictive in the effective execution of complex learning models at the edge [27].
- Memory is limited. Indeed, existing TinyML edge platforms operate with lesser than 1 MB onboard flash memory [27]. This restricts the performance of models and presents a significant challenge in view of accommodating the MCU.
- The cost of large-scale deployments can be significant even when the cost per device is low. Thus, for the success of low-cost edge platforms, monetary issues must be addressed [85].

B. Lessons Learnt

Through our comprehensive survey, we identified that the key issue in research over TinyML includes low availability of power within edge devices, hence, calling for energy-efficient TinyML system designs. In this regard, efficient energy harvesting techniques must be implemented for powering smart devices [86], so that an appropriate amount of energy is dedicated to ML-related tasks. Limited memory is another factor that hinders the growth of TinyML. Thus, research must be focused on the low memory footprint of edge hardware for the TinyML systems. In terms of problems related to clock speeds, much research attention must be focused on providing the optimal solution to address the issues related to

the capacity of the processor. Also, running complicated ML algorithms on MCU is difficult due to the low CPU capability.

The heterogeneity in the infrastructures of hardware/software poses significant challenge to TinyML systems in view of adopting a specific learning mechanism and deployment strategy. Also, existing domain related to edge computation is in an early stage which does not allow for the adoption of resources that change dynamically within the edge devices. Hence, it will be required to include device mobility and reliability factors during the deployment of ML models. In regard to reliability, the significant issues will be in terms of variations in process, hard and soft errors, and aging. Hence, it will be important to ensure that a specific edge device undergoes the reliability assessment before it is deployed for any application.

With the implementation of TinyML, new ML models will have to be formulated for introduction within the TinyML ecosystem. Techniques such as federated learning, transfer learning, and reinforcement learning can be used for the model design which must provide real-time solutions. In regard to edge-based solutions, the edge infrastructure must be based on techniques of virtual optimization for supporting multiple level dynamics at edge. Considering software for the edge, such a design will require specialized skill sets. The merging of edge devices and software is an additional challenge to be addressed. Overall, the edge intelligence framework must provision advanced applications such as 5G/6G wireless networking, data and cooperative intelligence, management of energy efficiency, ML as a service, etc.

Finally, the lack of bench-marking tools, data sets, and accepted models also presents a key challenge and must be resolved for TinyML research to move forward. The absence of standardization is one of the main problems due to which it is challenging for developers to generate cross-platform compatible solutions.

Table III lists the issues and challenges which have been identified, and also presents the possible solutions in terms of the proposed directions.

V. Conclusion

The development of ML techniques has led to a paradigm shift in the IoT ecosystem. Indeed, the integration of ML at the edge devices may ensure that the IoT systems can take intelligent decisions. As these edge devices are resource constrained, there is immense interest from the research community to implement low-complexity ML techniques, i.e., TinyML. The TinyML technology allows the tiny devices to be optimized, thereby ensuring accuracy and efficiency. In this article, we have surveyed the emerging growth of TinyML in the field of edge and energy computing IoT devices. The implementation of TinyML requires training the models, performing quantization techniques, and deploying the trained model on the hardware. The ML models to be deployed on edge devices have multiple research challenges due to the involved complexities, including the selection of hardware and compatibility of the framework. This article provides a detailed study of the available hardware platforms and software frameworks to encourage the growth potential of TinyML. We have also presented future research directions to various existing challenges in view of spurring future research on TinyML.

Despite the multiple difficulties, TinyML has immense potential to revolutionize multiple sectors including manufacturing, transportation, agriculture, and healthcare. We anticipate future creative uses of the TinyML technology as further studies are conducted and advanced use cases/applications emerge. Finally, through this comprehensive survey, we hope to have extended research on key issues related to TinyML and aid successful implementations of multiple TinyML applications.

References

[1] M. Goudarzi, M. S. Palaniswami, and R. Buyya, "A distributed deep reinforcement learning technique for application placement in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, p. 1–1, 2021.

[2] G. Muhammad and M. S. Hossain, "Emotion recognition for cognitive edge computing using deep learning," *IEEE Internet of Things Journal*, vol. 8, no. 23, p. 16894–16901, Dec 2021.

[3] W. Li, W. Deng, R. She, N. Zhang, Y. Wang, and W. Ma, "Edge computing offloading strategy based on particle swarm algorithm for power internet of things," in *IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. IEEE, Mar 2021, p. 145–150.

[4] J. Liu, C. Liu, B. Wang, G. Gao, and S. Wang, "Optimized task allocation for iot application in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 13, p. 10370–10381, Jul 2022.

[5] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, "A survey on cloudlets, mobile edge, and fog computing," in *8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE, Jun 2021, p. 139–142.

[6] J. Ying, J. Hsieh, D. Hou, J. Hou, T. Liu, X. Zhang, Y. Wang, and Y.-T. Pan, "Edge-enabled cloud computing management platform for smart manufacturing," in *IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*. IEEE, Jun 2021, p. 682–686.

[7] D. Wu, X. Huang, X. Xie, X. Nie, L. Bao, and Z. Qin, "Ledge: Leveraging edge computing for resilient access management of mobile iot," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, p. 1110–1125, Mar 2021.

[8] W. Bao, C. Wu, S. Guleng, J. Zhang, K.-L. A. Yau, and Y. Ji, "Edge computing-based joint client selection and networking scheme for federated learning in vehicular iot," *China Communications*, vol. 18, no. 6, p. 39–52, Jun 2021.

[9] J. Singh, Y. Bello, A. R. Hussein, A. Erbad, and A. Mohamed, "Hierarchical security paradigm for IoT multiaccess edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 7, p. 5794–5805, Apr 2021.

[10] C. Ding, A. Zhou, X. Ma, N. Zhang, C.-H. Hsu, and S. Wang, "Towards diversified iot services in mobile edge computing," *IEEE Transactions on Cloud Computing*, p. 1–1, 2021.

[11] N. Mahmood, O. López, O. Park, I. Moerman, K. Mikhaylov, E. Mercier, A. Munari, F. Clazzer, S. Böcker, and H. Bartz *(Eds.)*, "White paper on critical and massive machine type communication towards 6G [white paper]," *6G Research Visions*, vol. 11, 2020. [Online]. Available: http://urn.fi/urn:isbn:9789526226781

[12] P. P. Ray, "A review on TinyML: State-of-the-art and prospects," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 4, p. 1595–1623, Apr 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1319157821003335

[13] C. Guleria, K. Das, and A. Sahu, "A survey on mobile edge computing: Efficient energy management system," in *2021 Innovations in Energy Management and Renewable Resources(52042)*. IEEE, Feb 2021, p. 1–4.

[14] T. Ogino, "Simplified multi-objective optimization for flexible IoT edge computing," in *4th International Conference on Information and Computer Technologies (ICICT)*. IEEE, Mar 2021, p. 168–173.

[15] W. Ren, Y. Sun, H. Luo, and M. Guizani, "A demand-driven incremental deployment strategy for edge computing in IoT network," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 2, p. 416–430, Mar 2022.

[16] F. Johnny and F. Knutsson Arm, "CMSIS-NN & Optimizations for Edge AI," 2021.

[17] "Home | tinyml foundation." [Online]. Available: https://www.tinyml.org/

[18] P. Warden and D. Situnayake, *TinyML: machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers*. O'Reilly, 2020. [Online]. Available: https://books.google.com/books/about/TinyML.html?id=sB3mxQEACAAJ

[19] 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, Jun 2022. [Online]. Available: https://www.mdpi.com/2072-666X/13/6/851

[20] D. L. Dutta and S. Bharali, "TinyML Meets IoT: a comprehensive survey," *Internet of Things*, vol. 16, p. 100461, Dec 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660521001025

[21] M. Shafique, T. Theocharides, V. J. Reddy, and B. Murmann, "TinyML: current progress, research challenges, and future roadmap," in *58th ACM/IEEE Design Automation Conference (DAC)*, Dec 2021, p. 1303–1306.

[22] R. Immonen and T. Hämäläinen, "Tiny machine learning for resource-constrained microcontrollers," *Journal of Sensors*, vol. 2022, p. 1–11, Nov 2022. [Online]. Available: https://www.hindawi.com/journals/js/2022/7437023/

[23] H. Han and J. Siebert, "TinyML: a systematic review and synthesis of existing research," in *International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Feb 2022, p. 269–274.

[24] V. Tsoukas, E. Boumpa, G. Giannakas, and A. Kakarountas, "A review of machine learning and tinyml in healthcare," in *25th Pan-Hellenic Conference on Informatics*, ser. PCI 2021. New York, NY, USA: Association for Computing Machinery, Feb 2022, p. 69–73. [Online]. Available: https://doi.org/10.1145/3503823.3503836

[25] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, D. Patterson, D. Pau, J.-s. Seo, J. Sieracki, U. Thakker, M. Verhelst, and P. Yadav, "Benchmarking TinyML systems: Challenges and direction," 2020. [Online]. Available: https://arxiv.org/abs/2003.04821

[26] "TinyML as a service and machine learning at the edge - Ericsson." [Online]. Available: https://www.ericsson.com/en/blog/2019/12/tinyml-as-a-service

[27] E. Gousev, "Recent progress on tinyml technologies and opportunities." [Online]. Available: https://sites.google.com/g.harvard.edu/tinyml/lectures?authuser=0#h.839rbio9569w

[28] P. Jain, "Edgeml: Algorithms for tinyml." [Online]. Available: https://sites.google.com/g.harvard.edu/tinyml/lectures?authuser=0#h.5hc2tcel4ikp

[29] B. Turnquist and R. Dockter Boon Logic, "Amber: A complete, ML-based, anomaly detection pipeline for microcontrollers," 2020.

[30] C. Xu Eta Compute, "Enabling neural network at the low power edge: A neural network compiler for hardware constrained embedded system," 2020.

[31] S. Krstulovic, "Data collection design for real world tinyml." [Online]. Available: https://sites.google.com/g.harvard.edu/tinyml/lectures?authuser=0#h.5aj7gww1ta6s

[32] A. Eroma, "Unsupervised collaborative learning technology at the edge for industrial machine learning," Apr 2020. [Online]. Available: https://cms.tinyml.org/wp-content/uploads/talks2020/tinyML_Talks_Alexander_Eroma_200428.pdf

[33] R. Sanchez-Iborra and A. F. Skarmeta, "TinyML-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits and Systems Magazine*, vol. 20, no. 3, pp. 4–18, 2020.

[34] E. Tabanelli, G. Tagliavini, and L. Benini, "DNN is not all you need: Parallelizing non-neural ML algorithms on ultra-low-power IoT processors," 2022.

[35] X. Wang, M. Magno, L. Cavigelli, and L. Benini, "FANN-on-MCU: An open-source toolkit for energy-efficient neural network inference at the edge of the Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4403–4417, 2020.

[36] F. Fahim, B. Hawks, C. Herwig, J. Hirschauer, S. Jindariani, N. Tran, L. P. Carloni, G. D. Guglielmo, P. Harris, J. Krupa, D. Rankin, M. B. Valentin, J. Hester, Y. Luo, J. Mamish, S. Orgrenci-Memik, T. Aarrestad, H. Javed, V. Loncar, M. Pierini, A. A. Pol, S. Summers, J. Duarte, S. Hauck, S.-C. Hsu, J. Ngadiuba, M. Liu, D. Hoang, E. Kreinar, and Z. Wu, "hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices," 2021.

[37] F. Paissan, A. Ancilotto, and E. Farella, "PhiNets: a scalable backbone for low-power AI at the edge," *ACM Trans. Embed. Comput. Syst.*, vol. 21, no. 5, dec 2022. [Online]. Available: https://doi.org/10.1145/3510832

[38] O. Bringmann, W. Ecker, I. Feldner, A. Frischknecht, C. Gerum, T. Hämäläinen, M. A. Hanif, M. J. Klaiber, D. Mueller-Gritschneder, P. P. Bernardo, S. Prebeck, and M. Shafique, "Automated HW/SW co-design for edge AI: State, challenges and steps ahead," in *Proceedings of the 2021 International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES/ISSS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 11–20. [Online]. Available: https://doi.org/10.1145/3478684.3479261

[39] "TensorFlow Lite inference." [Online]. Available: https://www.tensorflow.org/lite/guide/inference

[40] "microtensor." [Online]. Available: https://utensor.github.io/website/

[41] "Edge impulse." [Online]. Available: https://www.edgeimpulse.com/

[42] "Home - NanoEdgeTM AI Studio." [Online]. Available: https://cartesiam.ai/

[43] "Home | PyTorch." [Online]. Available: https://pytorch.org/mobile/home/

[44] "The embedded learning library - Embedded Learning Library (ELL)." [Online]. Available: https://microsoft.github.io/ELL/

[45] "Introduction to STM32Cube.AI - STMicroelectronics." [Online]. Available: https://www.st.com/content/st_com/en/support/learning/stm32-education/stm32-moocs/Introduction_to_STM32CubeAI_MOOC.html

[46] D. Sun, D. Vlasic, C. Herrmann, V. Jampani, M. Krainin, H. Chang, R. Zabih, W. T. Freeman, and C. Liu, "Autoflow: Learning a better training set for optical flow," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10 088–10 097.

[47] "tinyML Talks: AutoFlow – an open source Framework to automatically implement neural networks on embedded devices | tinyML Foundation." [Online]. Available: https://cms.tinyml.org/wp-content/uploads/talks2022/tinyML_Talks_Daniel_Konegen_and_Marcus_Rub_220405.pdf

[48] "AutoFlow: learning a better training set for optical flow." [Online]. Available: https://autoflow-google.github.io/

[49] "Apache MXNet | A flexible and efficient library for deep learning." [Online]. Available: https://mxnet.apache.org/versions/1.9.1/

[50] "ML kit for firebase | firebase documentation." [Online]. Available: https://firebase.google.com/docs/ml-kit

[51] S. Iyer, R. Khanai, D. Torse, R. J. Pandya, K. M. Rabie, K. Pai, W. U. Khan, and Z. Fadlullah, "A survey on semantic communications for intelligent wireless networks," *Wireless Personal Communications*, vol. 129, no. 1, p. 569–611, Mar 2023. [Online]. Available: https://link.springer.com/10.1007/s11277-022-10111-7

[52] 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 *Interspeech 2020*. ISCA: ISCA, Oct 2020, p. 4054–4058. [Online]. Available: http://arxiv.org/abs/2005.11138

[53] J. Kwon and D. Park, "Hardware/software co-design for tinyml voice-recognition application on resource frugal edge devices," *Applied Sciences*, vol. 11, no. 22, p. 11073, Nov 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/22/11073

[54] Y. Zhang, S. Sun, and L. Ma, "Tiny transducer: A highly-efficient speech recognition model on edge devices," Jan 2021. [Online]. Available: http://arxiv.org/abs/2101.06856

[55] J. Li and R. Alvarez, "On the quantization of recurrent neural networks," Jan 2021. [Online]. Available: http://arxiv.org/abs/2101.05453

[56] Y. Zhang, S. Sun, and L. Ma, "Tiny transducer: A highly-efficient speech recognition model on edge devices," in *ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6024–6028.

[57] 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," *arXiv preprint arXiv:2005.11138*, 2020.

[58] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni, "The second 'chime' speech separation and recognition challenge: Datasets, tasks and baselines," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 126–130.

[59] A. J. Paul, P. Mohan, and S. Sehgal, "Rethinking generalization in american sign language prediction for edge devices with extremely low memory footprint," in *2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*. IEEE, Dec 2020, p. 147–152. [Online]. Available: https://ieeexplore.ieee.org/document/9332480/

[60] P. Mohan, A. J. Paul, and A. Chirania, *A Tiny CNN Architecture for Medical Face Mask Detection for Resource-Constrained Endpoints*. Springer, 2021, p. 657–670. [Online]. Available: https://link.springer.com/10.1007/978-981-16-0749-3_52

[61] S. G. Patil, D. K. Dennis, C. Pabbaraju, N. Shaheer, H. V. Simhadri, V. Seshadri, M. Varma, and P. Jain, "Gesturepod," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, Oct 2019, p. 403–415.

[62] M. de Prado, M. Rusci, A. Capotondi, R. Donze, L. Benini, and N. Pazos, "Robustifying the deployment of tinyml models for autonomous mini-vehicles," *Sensors*, vol. 21, no. 4, p. 1339, Feb 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1339

[63] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," Jan 2021. [Online]. Available: http://arxiv.org/abs/2101.09336

[64] H. Ren, D. Anicic, and T. Runkler, "TinyOL: TinyML with online-learning on microcontrollers," 2021. [Online]. Available: https://arxiv.org/abs/2103.08295

[65] H. Cai, C. Gan, L. Zhu, and S. Han, "Tinytl: Reduce activations, not trainable parameters for efficient on-device learning," 2020. [Online]. Available: https://arxiv.org/abs/2007.11622

[66] 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, Jun 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/12/4153

[67] M. A. Rashid H.A., Ren H. and M. T., "Tiny RespNet: A scalable multimodal TinyCNN processor for automatic detection of respiratory symptoms," 2020.

[68] B. Coffen and M. Mahmud, "Tinydl: Edge computing and deep learning based real-time hand gesture recognition using wearable sensor," in *2020 IEEE International Conference on E-health Networking, Application & Services (HEALTHCOM)*. IEEE, Mar 2021, p. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9399005/

[69] M. Vuletic, V. Mujagic, N. Milojevic, and D. Biswas, "Edge AI framework for healthcare applications," in *Proceedings of the 30th International Joint Conference on Artificial Intelligence, Virtual*, 2021, pp. 19–26.

[70] W. Raza, A. Osman, F. Ferrini, and F. D. Natale, "Energy-efficient inference on the edge exploiting tinyml capabilities for uavs," *Drones*, vol. 5, no. 4, p. 127, Oct 2021. [Online]. Available: https://www.mdpi.com/2504-446X/5/4/127

[71] A. I. Awad, M. M. Fouda, M. M. Khashaba, E. R. Mohamed, and K. M. Hosny, "Utilization of mobile edge computing on the internet of medical things: A survey," *ICT Express*, no. xxxx, May 2022. [Online]. Available: https://doi.org/10.1016/j.icte.2022.05.006

[72] K. Pai, R. Kallimani, S. Iyer, B. U. Maheswari, R. Khanai, and D. Torse, "A survey on brain-computer interface and related applications," Mar 2022. [Online]. Available: http://arxiv.org/abs/2203.09164

[73] T. Merk, V. Peterson, R. Köhler, S. Haufe, R. M. Richardson, and W.-J. Neumann, "Machine learning based brain signal decoding for intelligent adaptive deep brain stimulation," *Experimental Neurology*, vol. 351, no. August 2021, p. 113993, May 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0014488622000188

[74] H. K. Bharadwaj, A. Agarwal, V. Chamola, N. R. Lakkaniga, V. Hassija, M. Guizani, and B. Sikdar, "A review on the role of machine learning in enabling iot based healthcare applications," *IEEE Access*, vol. 9, p. 38859–38890, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9355143/

[75] P. Padhi and F. Charrua-Santos, "6g enabled tactile internet and cognitive internet of healthcare everything: Towards a theoretical framework," *Applied System Innovation*, vol. 4, no. 3, p. 66, Sep 2021. [Online]. Available: https://www.mdpi.com/2571-5577/4/3/66

[76] M. de Prado, M. Rusci, A. Capotondi, R. Donze, L. Benini, and N. Pazos, "Robustifying the deployment of tinyml models for autonomous mini-vehicles," *Sensors*, vol. 21, no. 4, p. 1339, Feb 2021. [Online]. Available: http://dx.doi.org/10.3390/s21041339

[77] A. N. Roshan, B. Gokulapriyan, C. Siddarth, and P. Kokil, "Adaptive traffic control with tinyml," in *Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2021, pp. 451–455.

[78] F. Nakhle and A. L. Harfouche, "Ready, steady, go AI: A practical tutorial on fundamentals of artificial intelligence and its applications in phenomics image analysis," *Patterns*, vol. 2, no. 9, p. 100323, Sep 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666389921001719

[79] D. J. Curnick, A. J. Davies, C. Duncan, R. Freeman, D. M. P. Jacoby, H. T. E. Shelley, C. Rossi, O. R. Wearn, M. J. Williamson, and N. Pettorelli, "SmallSats: a new technological frontier in ecology and conservation?" *Remote Sensing in Ecology and Conservation*, vol. 8, no. 2, p. 139–150, Apr 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rse2.239

[80] F. Alongi, N. Ghielmetti, D. Pau, F. Terraneo, and W. Fornaciari, "Tiny neural networks for environmental predictions: An integrated approach with Miosix," in *IEEE International Conference on Smart Computing (SMARTCOMP)*, 2020, pp. 350–355.

[81] M. Lord, "TinyML, anomaly detection," Ph.D. dissertation, California State University, Northridge, 2021.

[82] E. Jørgensen Njor, J. Madsen, and X. Fafoutis, "A primer for tinyML predictive maintenance: Input and model optimisation," in *Proceedings of 18th International Conference on Artificial Intelligence Applications and Innovations*, vol. 647, 2022, pp. 67–78. [Online]. Available: https://ifipaiai.org/2022/

[83] M. Day, "Programmable power management in the world of IoT," Dec 2022. [Online]. Available: https://embeddedcomputing.com/technology/analog-and-power/batteries-power-supplies/programmable-power-management-in-the-world-of-iot

[84] Omar.unwrap();, "How to estimate your embedded IoT device power consumption," Apr 2022. [Online]. Available: https://dev.to/apollolabsbin/3-simple-steps-to-estimate-your-embedded-iot-device-power

[85] D. Situnayake, "Mlops for tinyml." [Online]. Available: https://sites.google.com/g.harvard.edu/tinyml/lectures?authuser=0#h.m9uxfxjs8d5u

[86] O. L. A. López, H. Alves, R. D. Souza, S. Montejo-Sánchez, E. M. G. Fernández, and M. Latva-Aho, "Massive wireless energy transfer: Enabling sustainable IoT toward 6G era," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8816–8835, 2021.