



TinyML: Tools, applications, challenges, and future research directions

Rakhee Kallimani¹ · Krishna Pai⁴ · Prasoon Raghuwanshi² · Sridhar Iyer³ · Onel L. A. López²

Received: 25 May 2023 / Revised: 25 July 2023 / Accepted: 31 August 2023 /

Published online: 9 September 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

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, detailed prospects are presented which include various research challenges and identification of future directions.

Keywords TinyML · Embedded AI · Edge computing · IoT

1 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

✉ Krishna Pai
krishnapai271999@gmail.com

Extended author information available on the last page of the article

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 a hindrance to developing a standard ML architecture for all edge IoT networks.

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 optimising ML algorithms to make them more energy-efficient. Concurrently, there is also a growing demand to miniaturise 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 that 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.

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, continued progress has been limited by the 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. Further, several questions remain unanswered and solutions to these timely issues are required. The key issues which require immediate attention include (i) key requirements and applications of TinyML, (ii) capability of TinyML to implement DNNs at the

edge, and (iii) power consumption versus accuracy trade-offs appropriate for and attainable by TinyML. All in all, there are still multiple gaps in TinyML research, requiring timely solutions. This survey article presents detailed prospects including a list of several open research questions, an outline of potential obstacles in research on TinyML, and suggestions of possible directions for efficient solutions.

1.1 Research methodology

In this sub-section, we outline the related literature and the main findings to better understand the context for our survey and identify the relevant gaps. This will help us contribute to the ongoing discussions on TinyML.

To conduct this thorough survey, we follow the guidelines provided by [20–22]. Further, following [23], we frame the specific research questions based on the concept of Population, Intervention, Comparison, Outcomes, and Context. These specific review questions steer the survey within the article. The main aim of the review is to answer the following research questions:

- What is TinyML for?
- Is TinyML capable of running DNNs at the edge?
- How to keep the energy consumption less?
- Is high accuracy achievable by TinyML?

To conduct a detailed survey of studies shedding light on these questions, our search strategy is implemented in three phases: (i) identification of the keywords and defining search strings, (ii) data sources selection, and (iii) search process within the data sources. During the search process, we use web-based resources (e.g., Google Scholar, Scopus, and Web of Science) as the database, and also conduct a manual search for the articles from key authors within the TinyML research domain. The keywords used for searching include TinyML, Tiny Machine Learning, Tiny Deep Learning, Tiny DL, Tiny-DL, tinymL, tiny ML, and tiny machine learning. To conduct a complete search, we use the following six key databases as sources: (i) IEEE (ieee.org), (ii) Springer (springer.com), (iii) Elsevier (sciencedirect.com), (iv) Association for Computing Machinery (acm.org), (v) Wiley-Blackwell (onlinelibrary.wiley.com), and (vi) arXiv (arxiv.org). To obtain a broader insight, the chosen studies contain academic literature, and grey publications such as reports, authentic websites, and working/white papers. The contribution of a specific database to the current article is shown in Fig. 1 (a).

The initial literature search provided a total of 150 sources, and upon reviewing the titles, abstracts, and keywords, and removing the duplicates, 125 unique sources were obtained for further evaluation. Then, upon implementing the inclusion and exclusion criteria during a full-text reading phase, we excluded 15 sources to obtain a total of 115 sources, which comprise of 63 primary studies, 20 reviews, and 29 authentic websites as shown in Fig. 1 (b). For inclusion, we set the criteria that the source's research topic must focus on TinyML and low-power, and experimental applications/use cases. Also, the sources detailing marketed hardware and open-source software, published in key databases, and published between the years 2019 to 2023 are included.

Next, to implement the data extraction and synthesis, the content analysis method is implemented to identify the appearance of specific word(s), topic(s), and/or concept(s) within the text [24]. Through the data synthesis process, we integrated a diverse range of studies within a conceptual map, and use Mendeley and Microsoft Excel spreadsheets to synthesise the data within five TinyML elements viz., software, hardware, framework (detailed

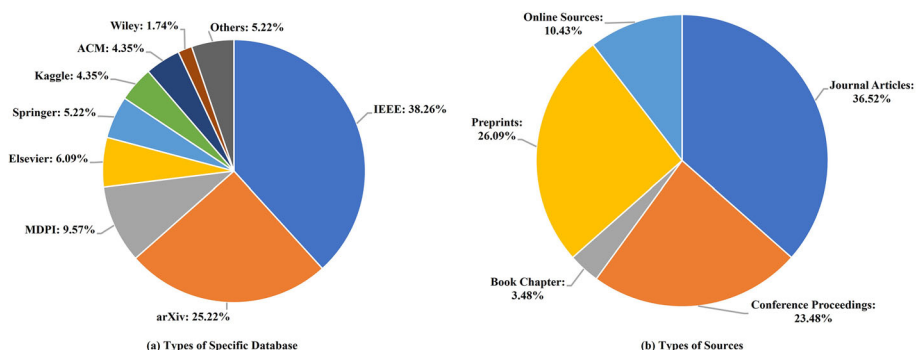


Fig. 1 (a) Percentage of studies contributing to the current article by specific databases. (b) Overview of types of sources considered for our survey

in Section 2), use-cases/applications (detailed in Section 3), and key challenges and future road-map (detailed in Section 4).

Finally, for accessing the quality of the included sources, every source is weighted on the following specific questions [21]: (i) does the source include clear research goals, (ii) did the source achieve the aims stated, and (iii) does the source document the research process aptly and reports sufficient findings related to the goals. As a response, the answer could be (i) ‘1’, implying that the source answered all the questions well; (ii) ‘0.5’, implying that the source partially answered all the questions; and (iii) ‘0’, implying that none of the questions were answered by the source. Following this quality assessment of all the selected primary sources, due to low quality, sources could be excluded. Further, the included sources were identified in relation to the research questions which they answered which enabled the extraction of the applications (detailed in Section 3) and the challenges (detailed in Section 4).

1.2 Contemporaneous survey on TinyML

The above methodology enabled us to identify the most recent articles over-viewing and surveying various aspects of TinyML. Next, we briefly discuss their main contributions, findings, and limitations.

The authors in [12], conducted an intuitive review of the possibilities, background, state-of-art architectures, and tool-sets to support TinyML. They discussed the key techniques to improve TinyML systems while detailing the key challenges and future research road-map. However, recent case studies and detailed directions to highlight the key research findings are not provided. In [19], an overview and review of TinyML studies is presented with respect to different ML models types, data-sets, device types characteristics, available resource constraints (e.g., hardware platforms), and supporting platforms. However, the study does not address any specific application(s) enabled by TinyML. The authors in [25] provided the definition of TinyML and presented background information on the various related technologies. The study introduced TinyML as a service and the role of 5G technology for TinyML IoT. Lastly, recent progress on the TinyML research, together with future opportunities and challenges, are detailed. However, the study fails to address the need for standardisation of TinyML.

In [26], a cross-layer design flow is proposed as the key aim of TinyML. The authors discussed various TinyML applications, frameworks, and highlighted the need for benchmarking by implementing TinyML via efficient hardware and software designs. Further, multiple challenges, opportunities, and vision for the road ahead on TinyML are detailed. The study however does not discuss the supporting platforms and available library/framework for specific applications. The authors in [27] provided the resource optimisation challenges related to TinyML and detailed the recent state of frameworks, libraries, development environments, and tools bench-marking. They discussed the emerging techniques and approaches to boost/expand the TinyML process and improve data privacy and security. However, information related to specific state-of-art use-cases/applications is not provided. In [28] a systematic review of TinyML research is conducted, and relevant literature on TinyML hardware, framework, data sets, use cases, and algorithms/models is provided. The authors detailed the existing literature on TinyML; however, existing challenges concerning the multiple constraints and future directions for research on TinyML is not included in the study.

The authors in [29] reviewed the contribution of TinyML for healthcare applications at the edge. Integration requirements of ML, followed by solutions and optimisation of neural networks (NNs) by TinyML, are discussed in detail. The study does not discuss any use-case(s) related to the use of TinyML in the healthcare domain. In [30], the current landscape, with challenges and directions, to develop a fair and useful hardware benchmark and selection methodology for TinyML workloads is detailed. The study, however, does not present any possible solutions to future research challenges. The authors in [31] surveyed reformable TinyML solutions and proposed the taxonomy to discuss the hierarchical layer's suitability for reformability. The authors also explored the TinyML workflow, and analysed several deployment schemes, limited available benchmarks, and tools. The impact of reformable TinyML on selected industrial applications with related challenges and future directions is also discussed. The study however does not discuss supporting platforms and available libraries/frameworks for the TinyML applications.

In comparison to the above recent studies, our survey in the current article provides details about the supporting software/library and platforms including the related targeted state-of-art applications. Our study also discusses the key existing challenges with respect to different constraints and possible directions. Table 1 compares the contribution of this survey article with respect to the most recent articles on TinyML.

1.3 Scope and contribution of this article

The detailed review of the recent survey articles makes it evident that although there is an extensive body of literature on TinyML, supporting tools and applications-related aspects are not often seriously addressed. Hence, in this article, we detail the main applications motivating the TinyML research and list the corresponding supporting tools. We then discuss the key TinyML enablers and advances while performing a state-of-art survey. Lastly, the prospects are presented by listing several relevant research challenges followed by a future road-map detailing the corresponding research directions.

The main contributions of this article are as follows:

- Presenting an intuitive discussion of TinyML and providing detailed insights regarding the related fundamentals.
- Detailing the existing TinyML technology supporting tool-sets which are used for training the model to be deployed at the edge.

Table 1 Summary of Recent Surveys related to TinyML

Reference	Key Contribution	Limitations w.r.t our survey
Ray [12]	<ul style="list-style-type: none"> • Intuitive review regarding possibilities for TinyML. • Key challenges and future road-map to mitigate numerous research issues of TinyML. 	<ul style="list-style-type: none"> • Recent case studies of TinyML. • Detailed directions to highlight key research findings.
Alajlan and Ibrahim [19]	<ul style="list-style-type: none"> • Overview and review of TinyML studies. • Analysis of ML models types used for TinyML. 	<ul style="list-style-type: none"> • Specific applications.
Dutta and Bharali [25]	<ul style="list-style-type: none"> • Definition of TinyML. • Role of 5G for TinyML IoT. • Recent progress in TinyML research. 	<ul style="list-style-type: none"> • Need for standardisation.
Shafique et al. [26]	<ul style="list-style-type: none"> • TinyML applications, frameworks and bench-marking. • Challenges, opportunities, and vision for the road ahead on TinyML. 	<ul style="list-style-type: none"> • Specific applications.
Immonen and Hämäläinen [27]	<ul style="list-style-type: none"> • Resource optimisation challenges of TinyML. • Future development of TinyML. 	<ul style="list-style-type: none"> • Specific state-of-art use-cases/applications.
Han and Siebert [28]	<ul style="list-style-type: none"> • Relevant TinyML literature on hardware, framework, data sets, use cases, and algorithms/models. • Road-map to understand literature on TinyML. 	<ul style="list-style-type: none"> • Existing challenges concerning multiple constraints, and future directions for research on TinyML.
Tsoukas et al. [29]	<ul style="list-style-type: none"> • Review of the contribution of TinyML in healthcare applications at the edge. • Optimisation of NNs by TinyML. 	<ul style="list-style-type: none"> • Use-case related to healthcare.
Banbury et al. [30]	<ul style="list-style-type: none"> • Current landscape of TinyML. • Challenges and directions to develop fair and useful hardware benchmark for TinyML workloads. 	<ul style="list-style-type: none"> • Possible solutions to future research challenges.
Rajapakse [31]	<ul style="list-style-type: none"> • Survey of reformable TinyML solutions. • Overview of TinyML workflow. • Challenges, and future directions. 	<ul style="list-style-type: none"> • Supporting platforms and available library/framework for every application.
Our Survey	<ul style="list-style-type: none"> • Supporting software/library and platforms and associated targeted applications. • Key existing challenges with respect to different constraints and possible directions. 	–

- Identifying and discussing the key TinyML enablers, metrics, and multiple use cases/applications of TinyML.
- Discussing current and future research challenges and proposing possible solutions with an outlook towards furthering the research on TinyML.

The rest of the paper is structured as follows. Section 2 overviews the TinyML technology and details the various tools that support TinyML enablers. In Section 3, we describe the key metrics applicable to TinyML following which, we detail the multiple state-of-art applications of TinyML enabled by advanced technologies. Section 4 identifies the various challenges and also proposes future research directions. Finally, Section 5 concludes the survey.

2 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, 32]. To provision such applications, TinyML necessitates a seamless integration of both hardware and software components. Following the research methodology detailed in Section 1, we reviewed the most recent studies which address the hardware and software aspects related to TinyML. In specific, from the 115 total sources, after having identified the 63 primary studies, we found 16 studies which detail the requirements of TinyML regarding hardware and software. These studies were identified by using the keywords TinyML hardware and software, Tiny Machine Learning hardware and software, tiny ML hardware and software, and tiny machine learning hardware and software. During this search, synonymous terms related to hardware and software were also found in few studies. As these were related to TinyML, such studies were included in the survey. These studies are detailed next.

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) [33]. 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 [34]. 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 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 optimise 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 [35, 36].

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 [33, 37]. 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 [38].

TinyML frameworks are continuously under development by multiple industries, specific developers, and research groups. The study conducted by [39] 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 Lite, 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 [40] 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 NNs. 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 NN (FANN)-on-MCU, which runs on PULP, is developed for reducing energy consumption by [41]. 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 [42], 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 quantisation and pruning-aware training for low-power embedded devices.

In [43], 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 [44], to address the hardware/software co-design, the HANNAH framework is detailed which aims at automating co-optimisation 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.

Table 2 Supporting platforms for TinyML

Reference	Software/Library / Framework	Developer	Supporting Platform form	Hardware Platforms	Targeted Applications
TensorFlow [45, 46]	TensorFlow (TFL)	Google Team	Android, iOS, Embedded Linux, Micro-controllers	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: ATSAM51, Himax WE-1 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
Coreml [47, 48]	CoreML	Apple	iOS	Mobile and Smart wearable devices with iOS support	Vision, Natural Language, Speech, Sound Analysis
microtensor [49]	Utensor	ARM	Android, iOS, Embedded Linux, Micro-controllers	Mbed, ST K64 ARM Boards	Image Classification, Gesture Recognition, Acoustic Detection and Motion Analysis
Edge [50]	Edge Impulse	Zach Shelby And Jan Jongboom	Android, iOS, Embedded Linux, Micro-controllers	Arduino Nano 33 BLE Sense, Arduino Nicla Sense ME, Arduino Nicla Vision, Arduino Portenta H7 + Vision Shield, Espressif ESP32, Himax WE-1 Plus, Nordic Semi Nrf52840 DK, Nordic Semi Nrf5340 DK	Asset Tracking and Monitoring, Human Interfaces, Predictive Maintenance
Home [51]	Nanoedge AI Studio	Cartesian	Android, Linux	STM32 Boards	Anomaly Detection, Predictive Maintenance, Condition Monitoring, Asset Tracking, People Counting, Activity Recognition

Table 2 continued

Reference	Software/Library / Framework	Developer	Supporting Platform form	Hardware Platforms	Targeted Applications
PyTorch [52, 53] Library [54] STM [55]	Pytorch Mobile	Meta AI (Facebook)	Android, iOS, Linux CPU	NNAPI (Android), Coreml (iOS), Metal GPU (iOS), Vulkan (Android)	Computer Vision and Natural Language Processing
	Embedded Learning Library (ELL)	Microsoft	Windows, Ubuntu Linux, Mac OS X	Raspberry Pi, Arduino, Micro: Bit	Image And Audio Classification
	STM32Cube.AI	STMicroelectronics	Android, Linux	STM32 ARM CORTEX Boards	Anomaly Detection, Predictive Maintenance, Condition Monitoring, Asset Tracking, People Counting, Activity Recognition
Sun et al. [56–58]	Autoflow	Daniel Koneg And Marcus Rüb	Android, iOS, Embedded Linux, Micro-controllers, Linux	MCU, FPGA Boards, Raspberry Pi	Image Classification, Object Detection, Pose Estimation, Speech Recognition, Gesture Recognition
MXNet [59]	Apache Mxnet	Apache Software Foundation (ASF)		Raspberry Pi, NVIDIA Jetson	Image Classification, Object Detection, Pose Estimation, Speech and Gesture Recognition
ML [60]	ML Kit for Firebase	Google	Android, iOS	Mobile Devices	Facial Detection, Bar-Code Scanning, Object Detection

In addition to the above studies identified from the primary sources, we identified 29 websites that provide extensive details regarding the TinyML software (or library or framework), and the corresponding developer and supporting platform. Through the search conducted in a similar manner as for the primary studies, we also identified information on the used hardware platforms and the targeted applications. Specifically, we used a search mechanism on the websites to extract information related to available hardware platforms for supporting design environments i.e., frameworks/libraries. Further, we also searched for the software/libraries which can be integrated with the related hardware platforms to provision specific applications/use cases. The details extracted from these multiple websites are summarised in Table 2.

Following the survey [61], we found that (i) the top two frameworks are Scikit-learn and TensorFlow, (ii) steady growth is observed in the case of PyTorch, and (iii) other frameworks such as Xgboost are now in trend. Further, the top three data types used by ML users are vision data, motion data, and sound data. Regarding the hardware boards, the most used board for developing the TinyML projects include (i) Raspberry Pi, (ii) Arduino Nano 33 BLE Sense, (iii) ESP32, and (iv) Raspberry Pi Pico and NVIDIA Jetson Nano [46, 50, 59].

Further, following [19], each industry has unique software and ML models to make the embedded systems board compatible with the TinyML applications. However, this may present significant issues since TinyML is implemented on extremely compact computer programs for tasks such as voice recognition and motion detection, and thus it will be challenging to utilise TinyML over several devices without compromising accuracy since each industry has its own software. Hence, it is crucial to provide a standardised 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. Overall, 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 an urgent need to encourage the design and training of TinyML models using specific software/libraries/frameworks, and deploy the trained models to the supporting hardware platform.

3 TinyML metrics and applications

3.1 Metrics for TinyML

With the continuous development of TinyML, the models will shrink further to fit the constrained devices thereby resulting in ultra-low power performance. Hence, appropriate metrics must be considered while developing systems enabled via TinyML. This inevitably calls for industry standards to benchmark the TinyML devices. To extract the key metrics for evaluating TinyML models, we conducted the search mechanism detailed in Section 1. Specifically, from the identified 63 primary studies, we found 9 studies that address the metrics applicable to TinyML. These studies were identified by using the keywords TinyML metrics, Tiny Machine Learning metrics, metrics for TinyML, tiny ML metrics. During the search, 2 studies provided details on the benchmarks for TinyML which are also included in the survey. From our survey, we identified that four key metrics are currently relevant for evaluating the TinyML systems. These include accuracy, power consumption, latency, and memory requirements [62, 63].

- Accuracy may be used as a metric for measuring ML models' performance, and larger models tend to outperform their smaller predecessors. Further, in TinyML systems, accuracy will be a critical metric since it will enable models to make real-time decisions based on data from the device's environment. However, a balance with the other metrics will be mandated.
- Power consumption is key to TinyML systems as they are expected to operate for long periods on batteries. In general, the power consumption of TinyML models should be in the order of mW, but the specific values depend on both, specifications of the available hardware instruction sets and underlying software (inference engine) through which the models are run. As an example, though the helium instruction set ARM® Cortex®-M85, significantly improves energy-efficiency compared to the ARM® Cortex®-M7 and the reference kernels, the CMSIS-NN library drastically improves the performance with respect to power consumption [64].
- Latency will be key evaluation parameter for the TinyML systems since systems enabled via TinyML will be expected to demonstrate significantly higher inference speeds. For TinyML systems, in comparison to the cloud-based systems, operation will occur at the endpoint not requiring any cloud connectivity. For applications related to speech and vision, autonomous vehicles, and maintenance, ultra-high inference speeds will be critical. As with the power consumption metric, latency will also depend on both, underlying hardware and software.
- Memory is currently a major hurdle for the TinyML systems since it is difficult to squeeze the ML models to fit into size-constrained micro-controllers (typically, less than 1 MB). In the case of memory, underlying software will play a major role since the developed model can be better optimised if the inference engines are effective. In other words, better memory management and libraries will execute the model layers effectively. Hence, minimising memory needs is an open challenge which requires techniques such as pruning and quantisation during the development of the model [65].

Recently, multiple studies have addressed key issues related to NN pruning in the context of TinyML. Notice that pruning is a NN optimisation technique that sacrifices accuracy for lower computational requirements. In [66], the authors proposed using an auxiliary network acting as an effective interpreter of the intermediate feature maps. The proposed technique demonstrates a parameter reduction of 93% on MLPerfTiny Visual Wakewords (VWW) task, and 28% on the Keyword Spotting (KWS) task with an accuracy cost of 0.65% and 1.06%, respectively. Further, when the proposed technique is evaluated on a Cortex-M0 micro-controller, it minimises the VWW model size by 4.7x and the latency by 1.6x; whereas, it gains 1% accuracy. Also, the KWS model size on Cortex-M0 is minimised by 1.2x and the latency by 1.2x at a cost of 2.21% accuracy. The authors in [67] presented a differentiable structured pruning method for convolutional NNs (CNNs). In this method, a model's MCU-specific resource usage is integrated with parameter importance feedback for obtaining highly compressed and accurate models. The proposed compressed models show high accuracy as they use MCU resource budget, and are generated faster than MCU specialist work. Moreover, the results demonstrate that the proposed methods (a) improve the NN resource usage by 80x; (b) incur negligible overhead, and improve the model training time; (c) generate compressed models with matching or improved resource use up to 1.4x in lesser time when compared to existing MCU-specific model compression methods. In [68], the authors proposed an energy-aware pruning method that quantifies the importance of every filter within the network through nuclear-norm (NN). The proposed technique results in high performance for Top-1

accuracy, FLOPs, and parameter reduction over a wide range of cases with multiple network architectures on CIFAR-10 and ImageNet after fine-grained classification tasks. The proposed method achieves results with 40.4/49.8% of FLOPs and 45.9/52.9% of parameter reduction with 94.13/94.61% in the Top-1 accuracy with ResNet-56/110 on CIFAR-10, respectively. Additionally, the study shows that when multiple pruning settings are considered in terms of data size and data quality, acceleration stability and compression can be emphasised with negligible accuracy loss. The authors in [69] developed a low-cost extension to existing pruning algorithms. An effective sparsity is used as a reference frame to partially reconfirm that random pruning with appropriate sparsity allocation across the layers performs as well or better in comparison to more sophisticated algorithms for pruning at the initial step. Based on this observation, the authors used the analogy of pressure distribution in coupled cylinders for designing layer-wise sparsity quotas, which are shown to outperform all the existing baselines in terms of random pruning.

The above four key metrics are correlated. Indeed, there is an inverse correlation between accuracy and memory, and a positive correlation between memory, latency, and power consumption [70]. Therefore, improving a specific metric affects the others, which requires consideration during the TinyML system development. Also, the TinyML system performance in terms of these metrics will vary in accordance with the application which is developed [71]. As a general thumb rule, it is required first to define the model accuracy requirement in accordance with the application, following which, multiple developed models can be compared against each other with respect to the other three metrics.

In addition to the metrics, benchmarks are also required for setting reproducible standards for comparison with various technologies, architectures, models, and software. The most popular benchmark for TinyML systems is MLCommons, which has been gaining traction as an industry standard since it measures the four metrics discussed above [72]. In addition, benchmarks such as CoreMark, MLMark, ML Perf Inference may also be adopted to balance optimality with portability, and comparability with representativeness [62]. Overall, the benchmarks for TinyML must support multiple options for model deployment while their impact on the performance results must be carefully evaluated.

3.2 State-of-art applications of TinyML

Even though TinyML is in an infant stage, there exist many well-established use cases and applications that implement TinyML for solving multiple issues. To extract the major applications of TinyML, we consulted 115 sources, as shown in Fig. 2. It can be observed that the dominant applications are related to speech and vision, comprising 34.21% of the total studies. They are followed by data pattern classification and compression, health diagnosis, brain computer interface, and sign language recognition applications, which comprise 10.53% of the total studies. Meanwhile, phenomics and conservation of ecology applications represent 7.89% of the total studies. Similarly, edge computing and autonomous vehicle applications comprise 10.52% of the total studies. Lastly, anomaly detection and predictive maintenance applications represent 5.26% of the total studies. In the next sub-section, we detail specific TinyML applications, including speech and vision, data pattern classification and compression, health diagnosis, edge computing, brain-control interface, autonomous vehicles, phenomics and ecology monitoring, anomaly detection, predictive maintenance, and sign language recognition, using various advanced technologies.

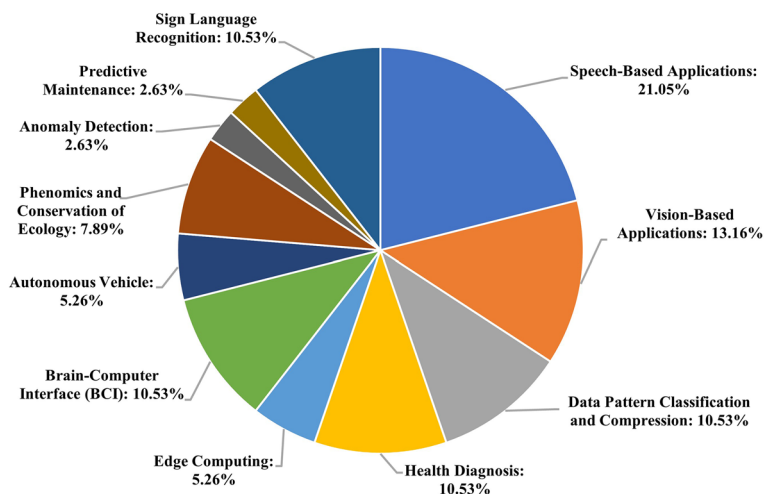


Fig. 2 Overview of surveyed TinyML applications

3.2.1 Speech-based applications

a) 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 [73]. Speech detection and recognition, online teaching/learning, and goal-oriented communication as shown in Fig. 3, 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 [74], the authors addressed sizing of the speech enhancement model as it was subjected to hardware resource constraints. The study employed structured pruning and integer quantisation for the Recurrent NN (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 utilised efficiently for energy-constrained edge devices executing voice-recognition applications, as demonstrated by the authors in [75]. 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 optimised partitioning among hardware and software co-design as future scope.

Authors in [76] 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

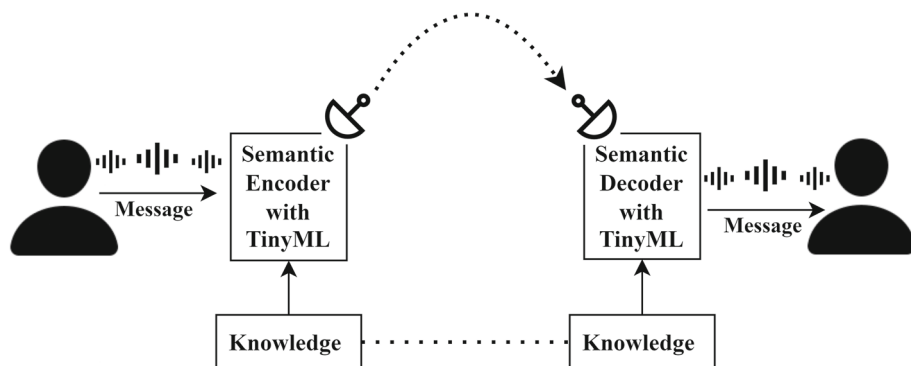


Fig. 3 Speech recognition and response

Finite State Transducers (WFST) based decoding allowed the flexible bias in the model improvement.

A similar study on speech recognition systems was conducted in [77] by introducing integer quantisation in the LSTM NN 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 [78] 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.

b) 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 ageing, 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 [79], 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 [80], the algorithm shows a computational latency of 2.39ms, which is far less than the 10ms target.

3.2.2 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 [81] 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 micro-controller with only 496 KB of frame-buffer

RAM. In effect, the authors addressed the major challenge of CNNs with high generalisation error, including large test and training accuracy. However, these did not generalise effectively to images within new cases and backgrounds with noise. The authors employed interpolation augmentation; results show 98.80% accuracy in the testing and 74.59% accuracy in generalisation. It was observed that interpolation augmentation reduced the drop in accuracy during quantisation in hand sign classification. However, it improved classification generalisation (with a 185 KB post quantisation model) and inference speed (to 20 fps). The authors also proposed the future scope to improve accuracy in generalisation model training on data from highly varied sources and testing it over hardware to attain the ambition of a 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 [82] deployed CNN architecture on a resource-constrained device. They developed a 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 quantisation and 30 frames per second inference speed on the targeted board. The authors also proposed the research scope on developing (i) quantisation schemes for reducing the precision from float32 to int8, (ii) data sets with heterogeneous sources, and (iii) experiments with smaller precision networks.

Authors in [83] 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 [84], 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 [85], where NAS was implemented in image classification and object detection problems. The authors addressed the real-time challenges such as deploying architectures of synthesised 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 categorisation with respect to acceleration techniques, cost estimation of hardware, search space, and search strategy was also available.

3.2.3 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 [86] 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 [45], and the model was trained to classify new data patterns. The research scope mentioned included the design of efficient and optimised algorithms for the NN to support online device training patterns.

Authors in [87] 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 utilise the memory over an edge device and avoid using the intermediate layers as activation. In addition, to up-hold 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 [88], emphasising 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.

3.2.4 Health diagnosis

With the spread of COVID-19, it is now required to continuously detect cough-related respiratory symptoms. The authors in [89] 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 patients, and information on demography to ensure the classification of respiratory symptoms.

The authors in [90] 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 analyse 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 the health of the elderly. In [91], a 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.

3.2.5 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 Fig. 4. In [92], 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 [93].

3.2.6 Brain-computer interface (BCI)

Within the healthcare sector, TinyML can contribute significantly; to, e.g., tumour and cancer detection, emotional intelligence, and health predictions using EEG and ECG signals [94] as shown in Fig. 5. With the aid of TinyML technology, Adaptive Deep Brain Stimulation (aDBS) [95] 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) [96]. The major applications of H-IoT are

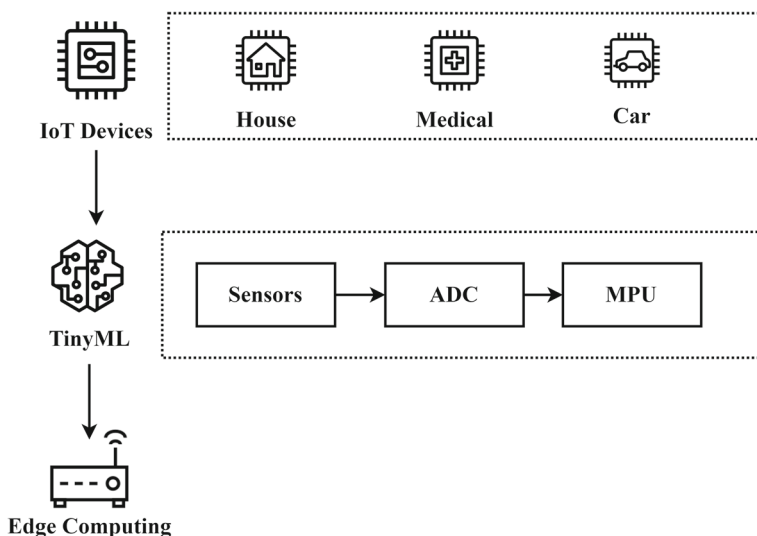


Fig. 4 Edge devices combined with TinyML for Edge Computing

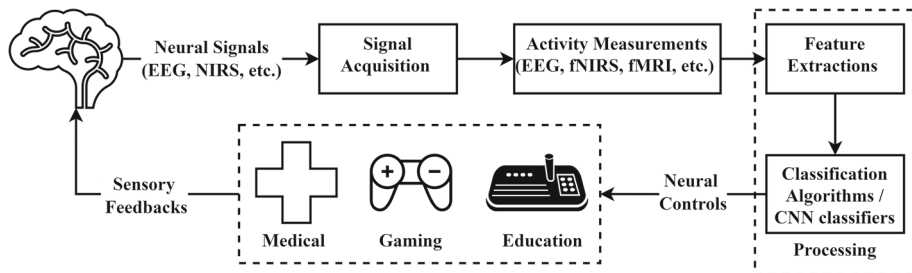


Fig. 5 BCI Methodology

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 [97].

3.2.7 Autonomous vehicle

Autonomous vehicles are utilised 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 [98] where it has been deployed over the GAP8 MCI which is enabled by the framework of 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 [99]. 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.

3.2.8 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 utilises improvements in genes to discriminate key germplasm. The study by [100] performed a phenomics image analysis which is based on tomato leaf disease and spider mites classification. The method utilises 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 [101] 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 [102], a deep tiny NN is detailed for prediction of weather. The proposed framework utilises 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.

3.2.9 Anomaly detection

Anomaly is defined as an event that varies from majority mass of events. In [103], 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.

3.2.10 Predictive maintenance

Predictive Maintenance has emerged as a promising maintenance paradigm in which models perform the prediction of equipment failure [104]. Currently, the majority of the predictive maintenance systems have been used either over cloud or via powerful computers. The data utilised 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 optimisation of the entire TinyML pipeline in an industrial setting which if achieved, will demonstrate immense potential for input optimisation in view of achieving predictive maintenance using TinyML.

3.2.11 Sign language recognition

Sign language recognition is another key application of TinyML. The sign languages comprise multiple hand movements and finger placements which form different gestures. Further, learning these sign languages presents various challenges, especially given so many different sign languages available across the globe. Hence, the interpretation of multiple sign languages imperatively requires a translator. Among the multiple sign languages available, American Sign Language is the most used [105]. Currently, there is a trend to implement machine learning (ML) techniques for building high-performance systems. The smart gloves enabled by ML techniques have demonstrated higher effectiveness in comparison to a camera-based solution. Such smart gloves can be implemented via TinyML using the CNN technique in view of obtaining higher accuracy and low footprint. The survey in [106] presented a comprehensive literature review of various algorithms which can be implemented for sign language recognition, detailed the existing systems, and proposed a system using the Raspberry Pi Pico.

The authors in [107] presented an intelligent electronic glove system that detects sign language numbers to automate the communication process between a deaf-mute person and others. The method used is the translation of hands movement as sign language to an oral

language. The proposed system is integrated into a glove which comprises of flex sensors within every finger for data collection. The collected data is analyzed through the following stages: (i) data balancing with Kennard-Stone (KS), (ii) comparison of prototypes selection between CHC evolutionary algorithm and Incremental Reduction Optimization Procedure3 (DROP3) to define the best data, and (iii) implementation of K-NearestNeighbors (kNN) as the classifier. The results demonstrated a reduction of data amount from first stage within system storage by 98%, and a classification performance of 85% with the implementation of CHC evolutionary algorithm.

In [108], the authors implemented gesture recognition and speech recognition applications on embedded systems with TinyML. Gesture recognition provides an innovative non-verbal communication approach as it used for human-computer interaction and sign language. For the implementation of hand gesture recognition, TinyML model is trained and deployed from EdgeImpulse framework. Then, based on hand movements, Arduino Nano 33 BLE device, with 6-axis IMU, and providing a set of embedded sensors, 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer is used to find the direction of hand movement. For implementing speech recognition, the TinyML model is trained and deployed from EdgeImpulse framework. Then, based on the keyword pronounced by a human, Arduino Nano 33 BLE, with a built-in microphone, glows an RGB LED. The results of various hand motions are perceived and applied as a contribution to a computer. Further, the hand motions which address numbers are changed accordingly to perform related tasks continuously.

3.3 Summary

The survey conducted in this article enable us to answer the specific research questions that are listed in Section 1. Notice that TinyML involves the use of low-complexity hardware and software to reduce the power consumption and cost of intelligent IoT devices. Indeed, TinyML technology is ideal for always-on use-cases and battery-operated devices. Further, the recent studies make it evident that TinyML enables the deployment of small DL and DNN models within a tiny edge device that is constrained in resources such as computation, power, and memory. In such cases, TinyML allows to analyse and interpret the data locally and take real time decisions. Also, via quantisation techniques, it is possible to deploy pre-trained DL models into tiny edge devices simultaneously ensuring minimum accuracy degradations. Redundant network and parameters structures can be removed using pruning methods. Our detailed survey has revealed that even though pruning and quantisation techniques minimise the TinyML model size and processor utilisation, the implementation of these methods decreases the prediction/classification accuracy. The resulting accuracy also depends on the quality of data set and computation capabilities. Additionally, the TinyML technology will not require extensive server infrastructure, thus leading to energy, resource, and cost savings.

4 Prospects

In this section, we present the prospects on TinyML research by detailing the various research challenges and issues related to the multiple TinyML applications, and the potential research road-map. The challenges are extracted after performing the data synthesis step within the survey process detailed in Section 1, through which it is identified that TinyML encounters major hurdles hindering the required growth pattern. The key challenges are detailed in the following sub-section.

4.1 Challenges

Resources such as, power and memory are limited in TinyML. Indeed, existing TinyML edge platforms operate with lesser than 1 MB onboard flash memory [33]. This restricts the performance of models in terms of accuracy and presents a significant challenge for accommodating the MCU. It is difficult to specify a universal power-management module for hardware platforms due to the variation in their designs and pre-processing paths. The limited memory size is another challenge as deploying the model needs a higher peak of memory. Due to the extremely small (e.g., KB) size SRAM and less than 1 MB size flash memory makes the task of ML very difficult to get deployed at the time edge devices [43].

The battery of embedded edge IoT devices is expected to be operational over a period of ten years [109]. In fact, in ideal conditions, a battery capacity of 2 Ah may have a life cycle of more than ten years considering that the power consumption is less than 12 μ A. 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 [110]. This will invariably reduce the life cycle of the 2 Ah battery to approximately 11 hours and 20 minutes. This aspect is a critical challenge in the TinyML ecosystem.

Regarding the hardware, existing benchmarks need to be re-framed before deploying TinyML on the resource constraint hardware. Also, TinyML systems should be operated on low-power micro-controllers boards with extensive hardware extraction [111]. Regarding the software, there are diverse frameworks, while resolving software heterogeneity will allow feasible deployment of TinyML within resource constrained devices [44]. Heterogeneity is a challenge as there are various embedded hardware available over a wide area of application. Notice that an extremely heterogeneous ecosystem demands different types of micro-controllers while the generated model is expected to work efficiently on the targeted embedded hardware [44].

The architecture of TinyML systems does not support the existing data set [12]. 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. Also, the lack of popularly accepted models is a research challenge, while the heterogeneous data type constitutes a major concern, especially for data and network management.

Moreover, in the existing edge computing infrastructure, the resources are changing dynamically, thus affecting the TinyML ecosystem [92]. Thus, there is a need for techniques/ algorithms to analyse dynamic data. Key issues also persist involving device mobility and reliability features during the ML model deployments. Also, 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 [33]. Lastly, 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 [112].

It is also important to incorporate reliability as a key performance indicator when TinyML is adopted for large-scale real-life applications [26]. However, this presents challenges regarding variations in process, errors which are soft in nature, and ageing. An edge device may be assessed for reliability before deployment within the field of applications. Deploying TinyML will require efficient management of the data and the network. Currently, edge networks are limited by their handling of the heterogeneous data types, which results in poor performance [81]. This is in part due limited understanding regarding data life cycle for sensor equipped devices. As a solution, it will be required to investigate the importance of network fault detection and identification. The existing incremental learning methods could be fused with

knowledge extraction techniques for knowledge sharing, and light-weight ML solutions can be incorporated for increasing autonomy and self-adaptivity at the edge network management [113].

Regarding the design of ML models, response time reduction is a key issue to be addressed. Indeed, new ML models providing short-time responses should be investigated, which may exploit techniques such as federated learning, transfer learning, reinforcement learning, and online learning while simultaneously aggregating the knowledge distillation dimension. These new ML models must be co-designed with control and communication aspects to enable real-time solutions. This may require the use of model pruning and quantisation during the model design process, while overall cost is minimised during coordination with the edge devices. Notice that existing literature has proposed ML models for TinyML to solve IoT-related issues, for which the interested reader may refer to [63, 90].

4.2 Future road-map

Through this comprehensive survey, we identified key TinyML research issues. One of them is the low power availability within edge devices, which calls for energy-efficient TinyML system designs. In this regard, efficient energy harvesting techniques must be implemented for powering smart devices [114], 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 and does not support 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. Regarding reliability, the significant issues will be in terms of variations in process, hard and soft errors, and ageing. Hence, it will be important to ensure that a specific edge device undergoes the reliability assessment and min-entropy evaluation [115] before it is deployed for any application.

With the implementation of TinyML, we foresee that new ML models will be incorporated within the TinyML ecosystem, exploiting techniques such as federated learning, transfer learning, and reinforcement learning, and providing real-time solutions. Regarding edge-based solutions, the edge infrastructure in the future will be based on techniques of virtual optimisation for supporting multiple level dynamics at edge. Considering software for the edge, such a design will require specialised skill sets. The merging of edge devices and software will be an additional challenge to be addressed. Overall, the edge intelligence framework will be required to provision advanced applications such as, 5G/6G wireless networking, data and cooperative intelligence, management of energy efficiency, ML as a service, etc. The lack of bench-marking tools, data sets, and accepted models will present a major challenge requiring timely solutions for TinyML research to move forward. The absence of standardisation is and will be remain a key problem due to which it will be challenging for developers to generate cross-platform compatible solutions.

Table 3 Summary of multiple existing challenges on TinyML research and suggested directions to obtain solutions [12]

Sl. No.	Constraint	Existing Challenges	Proposed Directions
1.	Resource	<ul style="list-style-type: none"> • Limited power availability and memory size • Quantise the model, use an appropriate converter, and implement memory on edge hardware appropriately 	<ul style="list-style-type: none"> • Design edge devices via co-design approach
2.	Hardware and Software	<ul style="list-style-type: none"> • Existing benchmarks to be re-framed before deploying TinyML on resource constraint hardware • Resolve software heterogeneity for feasible deployment of TinyML • Heterogeneity needs to be included within the design 	<ul style="list-style-type: none"> • Redesign benchmark to balance resource constraint and data heterogeneity of the system • Develop a generalised model to work efficiently with heterogeneous systems
3.	Data-set	<ul style="list-style-type: none"> • Existing data set cannot be used directly to train TinyML models • Lack of popularly accepted models • Consideration of heterogeneous data type for data and network management 	<ul style="list-style-type: none"> • Develop a ready model for adoption by TinyML ecosystem • Implement an intelligent network
4.	Existing edge infrastructure	<ul style="list-style-type: none"> • Edge computing infrastructure faces a challenge as resources are change dynamically • Edge platform faces issues during dynamic resource allocation 	<ul style="list-style-type: none"> • Implement enablers to support the system and leverage existing infrastructure • Employ optimisation techniques to support dynamic edge resource allocation
5.	Design of ML Models	<ul style="list-style-type: none"> • Response time of ML models is an issue 	<ul style="list-style-type: none"> • Design models to provide efficient and quick responses for all edge devices • Incorporate model pruning and quantisation as part of model design
6.	Reliability	<ul style="list-style-type: none"> • There exist issues of variations in process, errors which are soft in nature, and ageing 	<ul style="list-style-type: none"> • Ensure that an edge device is assessed for reliability before deployment within the field of applications

With TinyML gradually becoming a need and reality to solve decision aware real-life issues, it will soon become mandatory to use TinyML for low power embedded device pools used for multiple applications [65]. The wearable platforms will need to accommodate TinyML orientation for allowing users to experience sophisticated use-cases. In this regard, authors envision that the techno-developers and enterprises will soon begin to incorporate TinyML aware solutions for future generation smart and handheld devices. Currently, there is a strong need to minimise the resource hungry CPU-GPU-TPU interaction to facilitate smart decision-making supports. Hence, we envision that micro-controller manufacturers will begin to emphasise on the integration of TinyML design specifications so that the users are not concerned regarding external alignments to AI/ML. The aim must be to integrated TinyML into the IoT-edge analytics domain so that the application is user-friendly and reliable. Further,

standardised process flow requires major development for helping the developers to realise the ready-to-market deployment scenario. This will require the development of proper data set repository and light-weight bench-marking tools. Moreover, the IndustryX.0 applications must be targeted with TinyML accommodation soon, and the 8-bit micro-controller platforms must be leveraged with such low-memory footprint libraries. Also, delay mitigation at the edge-level implications must be resolved using TinyML. We emphasise upon the use of TinyML in the low-cost and hand-held digital devices for provisioning instantaneous feedback to the users. Lastly, we envision that the unnecessary use and dependency on GPUs, TPUs, and cloud platforms will be reduced by proper utilisation of the TinyML frameworks. Table 3 summarises the various issues and challenges that have been identified, and also presents the possible solutions in terms of the proposed directions.

5 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 optimised, 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 quantisation 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 revolutionise 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.

Author Contributions All the Authors contributed equally to the study and approved the final manuscript

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript

Data Availability Data sharing not applicable to this article as no data-sets were generated or analysed during the current study

Declarations

Conflicts of interest The authors have no conflict of interests to disclose

References

- Goudarzi M, Palaniswami MS, Buyya R (2021) A distributed deep reinforcement learning technique for application placement in edge and fog computing environments, *IEEE Trans Mob Comput*, p 1–1
- Muhammad G, Hossain MS (2021) Emotion recognition for cognitive edge computing using deep learning, *IEEE Internet Things J*, 8(23):16894–16901
- Li W, Deng W, She R, Zhang N, Wang Y, Ma W (2021) 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)*, p 145–150
- Liu J, Liu C, Wang B, Gao G, Wang S (2022) Optimized task allocation for iot application in mobile-edge computing, *IEEE Internet Things J*, 9(13):10370–10381
- Muniswamaiah M, Agerwala T, Tappert CC (2021) 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)*, p 139–142
- Ying J, Hsieh J, Hou D, Hou J, Liu T, Zhang X, Wang Y, Pan Y-T (2021) Edge-enabled cloud computing management platform for smart manufacturing, In *IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*, p 682–686
- Wu D, Huang X, Xie X, Nie X, Bao L, Qin Z (2021) Ledge: Leveraging edge computing for resilient access management of mobile iot, *IEEE Trans Mob Comput*, 20(3):1110–1125
- Bao W, Wu C, Guleng S, Zhang J, Yau K-LA, Ji Y (2021) Edge computing-based joint client selection and networking scheme for federated learning in vehicular iot, *China Commun*, 18(6):39–52
- Singh J, Bello Y, Hussein AR, Erbad A, Mohamed A (2021) Hierarchical security paradigm for IoT multiaccess edge computing, *IEEE Internet Things J*, 8(7):5794–5805
- Ding C, Zhou A, Ma X, Zhang N, Hsu C-H, Wang S (2021) Towards diversified iot services in mobile edge computing, *IEEE Transactions on Cloud Computing*, p 1–1
- Mahmood N, López O, Park O, Moerman I, Mikhaylov K, Mercier E, Munari A, Clazzer F, Böcker S, Bartz H (Eds.) (2020) White paper on critical and massive machine type communication towards 6G [white paper], *6G Research Visions*, vol. 11 [Online]. Available: <http://urn.fi/urn:isbn:9789526226781>
- Ray PP (2022) A review on TinyML: State-of-the-art and prospects, *Journal of King Saud University - Computer and Information Sciences*, 34(4):1595–1623, [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1319157821003335>
- Guleria C, Das K, Sahu A (2021) A survey on mobile edge computing: Efficient energy management system, In *Innovations in Energy Management and Renewable Resources(52042)*. IEEE, p 1–4
- Ogino T (2021) Simplified multi-objective optimization for flexible IoT edge computing, In *4th International Conference on Information and Computer Technologies (ICICT)*. IEEE, p 168–173
- Ren W, Sun Y, Luo H, Guizani M (2022) A demand-driven incremental deployment strategy for edge computing in IoT network, *IEEE Transactions on Network Science and Engineering* 9(2):416–430
- Johnny F, Knutsson Arm F (2021) CMSIS-NN & Optimizations for Edge AI,
- Home | tinymml foundation.” [Online]. Available: <https://www.tinymml.org/>
- Warden P, Situnayake D, TinyML: machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers. O'Reilly, [Online]. Available: <https://books.google.com/books/about/TinyML.html?id=sB3mxQEACAAJ>
- Alajlan NN, Ibrahim DM (2022) TinyML: enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications,” *Micromachines*, 13(6):851, [Online]. Available: <https://www.mdpi.com/2072-666X/13/6/851>
- Carrera-Rivera A, Ochoa W, Larrinaga F, Lasa G (2022) How-to conduct a systematic literature review: A quick guide for computer science research, *MethodsX*, 9:101895, . [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2215016122002746>
- Kitchenham BA, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering, *Keele University and Durham University Joint Report*, Tech. Rep. EBSE 2007-001, 07. Available: https://www.elsevier.com/_data/promis_misc/525444systematicreviewsguide.pdf
- Staples Mm, Niazi, “Experiences using systematic review guidelines, *J Syst Softw*, , 9:1425–1437, sep 2007. [Online]. Available: <https://doi.org/10.1016/j.jss.2006.09.046>
- Petticrew M, Roberts H (2006) *Systematic Reviews in the Social Sciences*. Oxford, UK: Blackwell Publishing Ltd, Jan . [Online]. Available: <http://doi.wiley.com/10.1002/9780470754887>
- Krippendorff K (2018) *Content analysis: An introduction to its methodology*. Sage publications
- Dutta DL, Bharali S (2021) TinyML Meets IoT: A comprehensive survey, *Internet of Thing*, 16:100461. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2542660521001025>
- Shafique M, Theocharides T, Reddy VJ, Murmann B (2021) TinyML: current progress, research challenges, and future roadmap, In *58th ACM/IEEE Design Automation Conference (DAC)*, p. 1303–1306

27. Immonen R, Hämäläinen T (2022) Tiny machine learning for resource-constrained microcontrollers, *J Sens*, (2022)1–11. [Online]. Available: <https://www.hindawi.com/journals/js/2022/7437023/>
28. Han H, Siebert J (2022) TinyML: A systematic review and synthesis of existing research, In *International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, p 269–274
29. Tsoukas V, Boumpa E, Giannakas G, Kakarountas A (2022) 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, p 69–73. [Online]. Available: <https://doi.org/10.1145/3503823.3503836>
30. Banbury CR, Reddi VJ, Lam M, Fu W, Fazel A, Holleman J, Huang X, Hurtado R, Kanter D, Lokhmotov A, Patterson D, Pau D, Seo J-s, Sieracki J, Thakker U, Verhelst M, Yadav P (2020) Benchmarking TinyML systems: Challenges and direction, [Online]. Available: <https://arxiv.org/abs/2003.04821>
31. Rajapakse V, Karunanayake I, Ahmed N (2023) Intelligence at the extreme edge: A survey on reformatable tinyml, *ACM Comput Surv*, Just Accepted. [Online]. Available: <https://doi.org/10.1145/3583683>
32. 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>
33. Goussev E (2020) Recent progress on tinyml technologies and opportunities. [Online]. Available: <https://sites.google.com/g.harvard.edu/tinyml/lectures?authuser=0#h.839rbio9569w>
34. Jain P (2020) Edgeml: Algorithms for tinyml. [Online]. Available: <https://sites.google.com/g.harvard.edu/tinyml/lectures?authuser=0#h.5hc2tcel4ikp>
35. Turnquist B, Dockter Boon Logic R (2020) Amber: A complete, ML-based, anomaly detection pipeline for microcontrollers
36. Xu Eta Compute C (2020) Enabling neural network at the low power edge: A neural network compiler for hardware constrained embedded system
37. Krstulovic S (2020) Data collection design for real world tinyml. [Online]. Available: <https://sites.google.com/g.harvard.edu/tinyml/lectures?authuser=0#h.5aj7gww1ta6s>
38. Eroma A (2020) Unsupervised collaborative learning technology at the edge for industrial machine learning, [Online]. Available: https://cms.tinyml.org/wp-content/uploads/talks2020/tinyML_Talks_Alexander_Eroma_200428.pdf
39. Sanchez-Iborra R, Skarmeta AF (2020) TinyML-enabled frugal smart objects: Challenges and opportunities. *IEEE Circ Syst Mag*, 20(3):4–18
40. Tabanelli E, Tagliavini G, Benini L (2022) DNN is not all you need: Parallelizing non-neural ML algorithms on ultra-low-power IoT processors
41. Wang X, Magno M, Cavigelli L, Benini L (2020) FANN-on-MCU: An open-source toolkit for energy-efficient neural network inference at the edge of the Internet of Things. *IEEE Internet Things J* 7(5):4403–4417
42. Fahim F, Hawks B, Herwig C, Hirschauer J, Jindariani S, Tran N, Carloni LP, Guglielmo GD, Harris P, Krupa J, Rankin D, Valentin MB, Hester J, Luo Y, Mamish J, Orgrenci-Memik S, Aarrestad T, Javed H, Loncar V, Pierini M, Pol AA, Summers S, Duarte J, Hauck S, Hsu S-C, Ngadiuba J, Liu M, Hoang D, Kreinar E, Wu Z (2021) hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices
43. Paissan F, Ancilotto A, Farella E (2022) PhiNets: A scalable backbone for low-power AI at the edge, *ACM Trans Embed Comput Syst*, 21(5), [Online]. Available: <https://doi.org/10.1145/3510832>
44. Bringmann O, Ecker W, Feldner I, Frischknecht A, Gerum C, Hämäläinen T, Hanif MA, Klaiber MJ, Mueller-Gritschneider D, Bernardo PP, Prebeck S, Shafique M (2021) 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, p 11–20. [Online]. Available: <https://doi.org/10.1145/3478684.3479261>
45. TensorFlow Lite inference. [Online]. Available: <https://www.tensorflow.org/lite/guide/inference>
46. Adi SE, Casson AJ (2021) Design and optimization of a tensorflow lite deep learning neural network for human activity recognition on a smartphone, In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp 7028–7031
47. Coreml. [Online]. Available: <https://docs.developer.apple.com/documentation/coreml>
48. Avola D, Cinque L, Fagioli A, Foresti GL, Marini MR, Mecca A, Pannone D (2022) Medicinal Boxes Recognition on a Deep Transfer Learning Augmented Reality Mobile Application. Cham: Springer International Publishing, 13231:489–499. [Online]. Available: https://link.springer.com/10.1007/978-3-031-06427-2_41
49. microtensor. [Online]. Available: <https://utensor.github.io/website/>
50. Edge impulse. [Online]. Available: <https://www.edgeimpulse.com/>
51. Home - NanoEdgeTM AI Studio. [Online]. Available: <https://cartesian.ai/>

52. Home | PyTorch. [Online]. Available: <https://pytorch.org/mobile/home/>
53. Dai X, Spasić I, Chapman S, Meyer B (2020) The state of the art in implementing machine learning for mobile apps: A survey. In 2020 SoutheastCon, pp 1–8
54. The embedded learning library - Embedded Learning Library (ELL). [Online]. Available: <https://microsoft.github.io/ELL/>
55. 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
56. Sun D, Vlasic D, Herrmann C, Jampani V, Krainin M, Chang H, Zabih R, Freeman WT, Liu C (2021) Autoflow: Learning a better training set for optical flow. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 10,088–10,097
57. 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
58. AutoFlow: learning a better training set for optical flow. [Online]. Available: <https://autoflow-google.github.io/>
59. Apache MXNet | A flexible and efficient library for deep learning. [Online]. Available: <https://mxnet.apache.org/versions/1.9.1/>
60. ML kit for firebase | firebase documentation. [Online]. Available: <https://firebase.google.com/docs/ml-kit>
61. Mooney P (2022) kaggle machine learning & data science survey. [Online]. Available: <https://kaggle.com/competitions/kaggle-survey-2022>
62. Banbury CR, Reddi VJ, Lam M, Fu W, Fazel A, Holleman J, Huang X, Hurtado R, Kanter D, Lokhmotov A, Patterson D, Pau D, sun D, sun Seo J, Sieracki J, Thakker U, Verhelst M, Yadav P (2021) Benchmarking tinyml systems: Challenges and direction,
63. López OA, Rosabal OM, Ruiz-Guirola D, Raghuwanshi P, Mikhaylov K, Lovén L, Iyer S (2023) Energy-sustainable iot connectivity: Vision, technological enablers, challenges, and future directions
64. Delnevo G, Prandi C, Mirri S, Manzoni P (2021) Evaluating the practical limitations of tinyml: an experimental approach. In IEEE Globecom Workshops (GC Wkshps), p 1–6
65. Taheri Tajar A, Ramazani A, Mansoorizadeh M (2021) A lightweight tiny-yolov3 vehicle detection approach. *Journal of Real-Time Image Processing*, 18(6):2389–2401. [Online]. Available: <https://link.springer.com/10.1007/s11554-021-01131-w>
66. De Leon JD, Atienza R (2022) Depth pruning with auxiliary networks for tinyml. In ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), p 3963–3967
67. Liberis E, Lane ND (2023) Differentiable neural network pruning to enable smart applications on micro-controllers. *Proc ACM Interact Mob Wearable Ubiquitous Technol*, 6(4) [Online]. Available: <https://doi.org/10.1145/3569468>
68. Yeom S-K, Shim K-H, Hwang J-H (2022) Toward compact deep neural networks via energy-aware pruning
69. Vysogorets A, Kempe J (2021) Connectivity matters: Neural network pruning through the lens of effective sparsity. *J Mach Learn Res*, 24:99:1–99:23
70. Khajooei A, Jamshidi MB, Shokouhi SB (2023) A super-efficient tinyml processor for the edge meta-verse. *Information*, 14(4):235. [Online]. Available: <https://doi.org/10.3390/info14040235>
71. Sudharsan B, Salerno S, Nguyen D-D, Yahya M, Wahid A, Yadav P, Breslin JG, Ali MI (2021) Tinyml benchmark: Executing fully connected neural networks on commodity microcontrollers. In IEEE 7th World Forum on Internet of Things (WF-IoT), pp 883–884
72. Mlcommons. Mar 2023. [Online]. Available: <https://mlcommons.org/>
73. Iyer S, Khanai R, Torse D, Pandya RJ, Rabie KM, Pai K, Khan WU, Fadlullah Z (2023) A survey on semantic communications for intelligent wireless networks. *Wirel Pers Commun*, 129(1):569–611 Mar 2023. [Online]. Available: <https://link.springer.com/10.1007/s11277-022-10111-7>
74. Fedorov I, Stamenovic M, Jensen C, Yang L-C, Mandell A, Gan Y, Mattina M, Whatmough PN (2020) Tynylstms: Efficient neural speech enhancement for hearing aids. In Interspeech 2020. ISCA: ISCA, Oct 2020, p 4054–4058. [Online]. Available: [arXiv:2005.11138](https://arxiv.org/abs/2005.11138)
75. Kwon J, Park D (2021) Hardware/software co-design for tinyml voice-recognition application on resource frugal edge devices. *Appl Sci*, 11(22):11073 Nov 2021. [Online]. Available: <https://www.mdpi.com/2076-3417/11/22/11073>
76. Zhang Y, Sun S, Ma L (2021) Tiny transducer: A highly-efficient speech recognition model on edge devices. Jan 2021. [Online]. Available: [arXiv:2101.06856](https://arxiv.org/abs/2101.06856)
77. Li J, Alvarez R (2021) On the quantization of recurrent neural networks. Jan 2021. [Online]. Available: [arXiv:2101.05453](https://arxiv.org/abs/2101.05453)


78. Zhang Y, Sun S, Ma L (2021) Tiny transducer: A highly-efficient speech recognition model on edge devices, In ICASSP IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp 6024–6028
79. Fedorov I, Stamenovic M, Jensen C, Yang L-C, Mandell A, Gan Y, Mattina M, Whatmough PN (2020) TinyLSTMs: efficient neural speech enhancement for hearing aids, [arXiv:2005.11138](https://arxiv.org/abs/2005.11138)
80. Vincent E, Barker J, Watanabe S, Le Roux J, Nesta F, Matassoni M (2013) The second ‘chime’ speech separation and recognition challenge: Datasets, tasks and baselines, In IEEE International Conference on Acoustics, Speech and Signal Processing, pp 126–130
81. Paul AJ, Mohan P, Sehgal S (2020) Rethinking generalization in american sign language prediction for edge devices with extremely low memory footprint, In IEEE Recent Advances in Intelligent Computational Systems (RAICS). IEEE, Dec 2020, p 147–152. [Online]. Available: <https://ieeexplore.ieee.org/document/9332480/>
82. Mohan P, Paul AJ, Chirania A (2021) A Tiny CNN Architecture for Medical Face Mask Detection for Resource-Constrained Endpoints. Springer, p 657–670. [Online]. Available: https://link.springer.com/10.1007/978-981-16-0749-3_52
83. Patil SG, Dennis DK, Pabbaraju C, Shaheer N, Simhadri HV, Seshadri V, Varma M, Jain P (2019) 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
84. de Prado M, Rusci M, Capotondi A, Donze R, Benini L, Pazos N (2021) Robustifying the deployment of tinyml models for autonomous mini-vehicles, *Sensors*, 21(4):1339 Feb 2021. [Online]. Available: <https://www.mdpi.com/1424-8220/21/4/1339>
85. Benmeziane H, Maghraoui KE, Ouarnoughi H, Niar S, Wistuba M, Wang N (2021) A comprehensive survey on hardware-aware neural architecture search, Jan 2021. [Online]. Available: [arXiv:2101.09336](https://arxiv.org/abs/2101.09336)
86. Ren H, Anicic D, Runkler T (2021) TinyOL: TinyML with online-learning on microcontrollers. [Online]. Available: [arXiv:2103.08295](https://arxiv.org/abs/2103.08295)
87. Cai H, Gan C, Zhu L, Han S (2020) Tinytl: Reduce activations, not trainable parameters for efficient on-device learning. [Online]. Available: [arXiv:2007.11622](https://arxiv.org/abs/2007.11622)
88. Signoretti G, Silva M, Andrade P, Silva I, Sisinni E, Ferrari P (2021) An evolving TinyML compression algorithm for IoT environments based on data eccentricity, *Sensors*, 21(12):4153. Jun 2021. [Online]. Available: <https://www.mdpi.com/1424-8220/21/12/4153>
89. MA Rashid HA, Ren H MT (2020) Tiny RespNet: A scalable multimodal TinyCNN processor for automatic detection of respiratory symptoms
90. Coffen B, Mahmud M (2021) 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/>
91. Vuletic M, Mujagic V, Milojevic N, Biswas D (2021) Edge AI framework for healthcare applications, In Proceedings of the 30th International Joint Conference on Artificial Intelligence, Virtual, pp 19–26
92. Raza W, Osman A, Ferrini F, Natale FD (2021) Energy-efficient inference on the edge exploiting tinyml capabilities for uavs, *Drones*, 5(4):127, Oct 2021. [Online]. Available: <https://www.mdpi.com/2504-446X/5/4/127>
93. Awad AI, Fouda MM, Khashaba MM, Mohamed ER, Hosny KM (2022) 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.ict.2022.05.006>
94. Pai K, Kallimani R, Iyer S, Uma Maheswari B, Khanai R, Torse D (2023) A Survey on Brain-Computer Interface and Related Applications. Bentham Science Publishers, May 2023, p 210–228. [Online]. Available: <https://www.eurekaselect.com/node/216769>
95. Merk T, Peterson V, Köhler R, Haufe S, Richardson RM, Neumann W-J (2022) Machine learning based brain signal decoding for intelligent adaptive deep brain stimulation, *Exp Neurol*, 351(2021):113993. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0014488622000188>
96. Bharadwaj HK, Agarwal A, Chamola V, Lakkaniga NR, Hassija V, Guizani M, Sikdar B (2021) A review on the role of machine learning in enabling iot based healthcare applications, *IEEE Access*, 9:38859–38890, [Online]. Available: <https://ieeexplore.ieee.org/document/9355143/>
97. Padhi P, Charrua-Santos F (2021) 6g enabled tactile internet and cognitive internet of healthcare everything: Towards a theoretical framework, *Applied System Innovation*, 4(3):66. [Online]. Available: <https://www.mdpi.com/2571-5577/4/3/66>
98. de Prado M, Rusci M, Capotondi A, Donze R, Benini L, Pazos N (2021) Robustifying the deployment of tinyml models for autonomous mini-vehicles, *Sens*, 21(4):1339, Feb 2021. [Online]. Available: <https://doi.org/10.3390/s21041339>

99. Roshan AN, Gokulapriyan B, Siddarth C, Kokil P (2021) Adaptive traffic control with tinyml, In Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp 451–455
100. Nakhle F, Harfouche AL (2021) Ready, steady, go AI: A practical tutorial on fundamentals of artificial intelligence and its applications in phenomics image analysis, *Patt*, 2(9): 100323, Sep 2021. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2666389921001719>
101. Curmick DJ, Davies AJ, Duncan C, Freeman R, Jacoby DMP, Shelley HTE, Rossi C, Wearn OR, Williamson MJ, Pettoirelli N (2022) SmallSats: A new technological frontier in ecology and conservation? *Remote Sensing in Ecology and Conservation*, 8(2):139–150 Apr 2022. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1002/rse2.239>
102. Alongi F, Ghielmetti N, Pau D, Terraneo F, Fornaciari W (2020) Tiny neural networks for environmental predictions: An integrated approach with Miosix, In IEEE International Conference on Smart Computing (SMARTCOMP), p 350–355
103. Lord M (2021) TinyML, anomaly detection, Ph.D. dissertation, California State University, Northridge
104. Jørgensen Njor E, Madsen J, Fafoutis X (2022) A primer for tinyML predictive maintenance: Input and model optimisation, In Proceedings of 18th International Conference on Artificial Intelligence Applications and Innovations, 647:67–78. [Online]. Available: <https://ifipaiai.org/2022/>
105. Quer J, Steinbach M (2019) Handling sign language data: The impact of modality, *Front Psych*, 10:483 Mar 2019. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fpsyg.2019.00483/full>
106. B SK, P R, Hiremath RB, Ramadurgam VS, Shaw DK (2022) Survey on implementation of tinyml for real-time sign language recognition using smart gloves, In Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT), pp 1–7
107. Rosero-Montalvo PD, Godoy-Trujillo P, Flores-Bosmediano E, Carrascal-García J, Otero-Potosi S, Benítez-Pereira H, Peluffo-Ordóñez DH (2018) Sign language recognition based on intelligent glove using machine learning techniques, In 2018 IEEE Third Ecuador Technical Chapters Meeting (ETCM), pp 1–5
108. V V, C RA, Prasanna R, Kakarla PC, PJ VS, Mohan N (2022) Implementation of tiny machine learning models on arduino 33 ble for gesture and speech recognition
109. Day M (2022) 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>
110. Omar.unwrap (2022) How to estimate your embedded IoT device power consumption, Apr 2022. [Online]. Available: <https://dev.to/apollo-labsbin/3-simple-steps-to-estimate-your-embedded-iot-device-power>
111. Reddi VJ, Plancher B, Kennedy S, Moroney L, Warden P, Agarwal A, Banbury C, Banzi M, Bennett M, Brown B, Chitlangia S, Ghosal R, Grafman S, Jaeger R, Krishnan S, Lam M, Leiker D, Mann C, Mazumder M, Pajak D, Ramaprasad D, Smith JE, Stewart M, Tingley D (2021) Widening access to applied machine learning with tinyml
112. Situnayake D (2020) Mlops for tinyml. [Online]. Available: <https://sites.google.com/g.harvard.edu/tinyml/lectures?authuser=0#h.m9uxfxjs8d5u>
113. Schizas N, Karras A, Karras C, Sioutas S (2022) Tinyml for ultra-low power ai and large scale iot deployments: A systematic review, *Future Internet*, 14(12):363 Dec 2022. [Online]. Available: <https://doi.org/10.3390/fi14120363>
114. López OLA, Alves H, Souza RD, Montejo-Sánchez S, Fernández EMG, Latva-Aho M (2021) Massive wireless energy transfer: Enabling sustainable IoT toward 6G era. *IEEE Internet Things J*, 8(11):8816–8835
115. Li H, Zhang J, Li Z, Liu J, Wang Y (2023) Improvement of min-entropy evaluation based on pruning and quantized deep neural network. *IEEE Trans Inf Forensic Sec*, 18:1410–1420

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Authors and Affiliations

Rakhee Kallimani¹ · Krishna Pai⁴ · Prasoon Raghuwanshi² · Sridhar Iyer³  · Onel L. A. López²

Rakhee Kallimani
rakhee.kallimani@klescet.ac.in

Prasoon Raghuwanshi
Prasoon.Raghuwanshi@oulu.fi

Sridhar Iyer
sridhariyer1983@klescet.ac.in

Onel L. A. López
onel.alcarazlopez@oulu.fi

¹ Department of Electrical and Electronics Engineering, KLE Technological University Dr. M.S. Sheshgiri Campus, 590008 Belagavi, Karnataka, India

² Faculty of Information Technology and Electrical Engineering, University of Oulu, Oulu 90014, Finland

³ Department of CSE(AI), KLE Technological University Dr. M.S. Sheshgiri Campus, 590008 Belagavi, Karnataka, India

⁴ Department of Electronics and Communication Engineering, KLE Technological University Dr. M.S. Sheshgiri Campus, 590008 Karnataka, Belagavi, India