Received 6 June 2023, accepted 7 July 2023, date of publication 10 July 2023, date of current version 12 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3294111

SURVEY

A Comprehensive Survey on TinyML

YOUSSEF ABADADE[1], ANAS TEMOUDEN[2], HATIM BAMOUMEN[2], (Student Member, IEEE), NABIL BENAMAR[2,3], YOUSRA CHTOUKI[2], AND ABDELHAKIM SENHAJI HAFID[4]

[1]System Engineering Laboratory, National School of Applied Sciences, Ibn Tofail University, Kenitra 14000, Morocco
[2]School of Sciences and Engineering, Al Akhawayn University, Ifrane 53000, Morocco
[3]Department of Computer Engineering, School of Technology, Moulay Ismail University, Meknes 50050, Morocco
[4]University of Montreal, Montreal, QC H3C 3J7, Canada

Corresponding author: Nabil Benamar (n.benamar@aui.ma)

ABSTRACT Recent spectacular progress in computational technologies has led to an unprecedented boom in the field of Artificial Intelligence (AI). AI is now used in a plethora of research areas and has demonstrated its capability to bring new approaches and solutions to various research problems. However, the extensive computation required to train AI algorithms comes with a cost. Driven by the need to reduce the energy consumption, the carbon footprint and the cost of computers running machine learning algorithms, TinyML is nowadays considered as a promising AI alternative focusing on technologies and applications for extremely low-profile devices. This paper presents the results of a literature survey of all TinyML applications and related research efforts. Our survey builds a taxonomy of TinyML techniques that have been used so far to bring new solutions to various domains, such as healthcare, smart farming, environment, and anomaly detection. Finally, this survey highlights the remaining challenges and points out possible future research directions. We anticipate that this survey will motivate further discussions on the various fields of applications of TinyML and the synergy of resource-constrained devices and edge intelligence.

INDEX TERMS TinyML, embedded machine learning, deep learning, edge intelligence.

I. INTRODUCTION

Recent advances in computational technologies have enabled an unprecedented boom in the field of Artificial Intelligence (AI). AI has become ubiquitous in our daily life. Ranging from game playing [1] to healthcare monitoring [2], [3] and passing by Natural Language Processing (NLP) [4], computer vision (CV) [5], social media [6], and Autonomous Driving [7], [8], more AI applications continue to expand as more people embrace this technology. However, the extensive computation needed to reach such exciting results in AI-based research and projects comes with a cost. Financially, the price of dedicated hardware to run AI algorithms is increasing. The cloud alternative does not always overcome the cost issue since the cloud compute time is proportional to the number of processes being executed [9]. This also impacts the environment, as a consequence of non-renewable energy supplied to modern tensor processing hardware. Authors in [9] evaluated the estimated cost of training some benchmark Deep Learning (DL) algorithms, in terms of CO2 emissions and cloud compute time. The results show that as the trained models become larger and larger so does their carbon footprint. To overcome the side effect of AI on the ecosystem and to make it more affordable for researchers and practitioners, tiny machine learning (TinyML) has recently emerged as a promising field of AI. Driven by the need to reduce the energy consumption and the CO2 emissions of computers running machine learning algorithms, TinyML is an AI alternative that exclusively uses extremely low-profile devices to process AI algorithms [10].

TinyML is a big shift in AI. It pushes the intelligence to the edge and makes use of tiny devices such as microcontollers to execute AI algorithms. TinyML enpowers low-latency, consumes low power, and uses low bandwidth. In modern computing technologies, standard CPUs consume generally up to 85 watts and standard GPUs consume between 200 watts to 500 watts [11], [12]. A typical microcontroller consumes thousand times less power, in the order of milliwatts or microwatts. Such low energy consumption of TinyML devices enables running them unplugged for weeks or even years.

In AI, training large ML models requires a significant amount of data that should be processed in the cloud network. AI researchers are nowadays more concerned about

The associate editor coordinating the review of this manuscript and approving it for publication was Gianluigi Ciocca.

the ethical aspect of machine learning than the accuracy of the model [13]. This has led to extensive works on privacy-preserving machine learning techniques for big data analysis [14]. There is a growing interest from the ML community in leveraging a plethora of cryptographic techniques to secure data both in the training phase and the testing phase. However, by deploying TinyML where the source of data is created and since this data does not leave the device, privacy concerns are largely addressed.

There are different promising domains where TinyML can bring a significant impact. In industry, anomaly detection is highly critical and can help reducing the downtime for repairs and thus increase the efficiency of the production process. By deploying ML algorithms in the edge, one can detect and analyze continuously the sound emitted by the machine during the production process, which can inform of a possible break down in that device. Analyzing different metrics such as sounds or vibrations in real time will help saving time in correcting or replacing any defective device without unnecessary delays. In environment [15], one of the recent research fields, where sensors have been extensively deployed, is the Internet of animals [16], [17]. Understanding animal behavior continues to be a gruelling task for a majority of researchers. It is indeed not an easy routine to track animals for hours or even days in their living place to document their behavior. Internet of Things (IoT) and especially TinyML can make this arduous work largely superfluous. It can help getting more detailed insights on animals life and predict possible threats. In the elephant TinyML project [18], collars are attached to an elephant and capture its real-time movements using GPS. The embedded sensors take surrounding images that are continuously processed and analyzed by TinyML and can predict events around each animal such as the presence of human predators. Other ML models can also be applied to understand and detect the mood of the elephant, while an accelerometer is used to further determine the movement of the elephant. TinyML brings new insights and unlocks new possibilities for sustainable development. By reducing the latency, TinyML enables real time applications to be deployed in the source of data, such as in the case of image and speech recognition. TinyML models can also run even when there is no internet connection, which can not be realized in the cloud context. In TinyML, the processed data do not need to leave the device, which significantly improves user privacy and thus complies with data protection regulations.

The scope of this paper is to present TinyML as a major candidate for the support of machine learning in small and constrained devices such as microcontrollers, by decomposing various related technologies and revealing research challenges and directions. This paper also highlights the power that TinyML brings to the research field of AI in general and Deep learning in particular. The current study shows to the AI researchers worldwide, who may have limited access to the high technology of dedicated servers and datacenters, that TinyML is an affordable alternative. Specifically, the contributions of this paper can be summarised as follows:

1) We provide a new classification of TinyML applications and techniques by surveying all published papers on TinyML until 2023.
2) We study in depth the main advantages of using TinyML compared to other existing approaches
3) We highlight how TinyML unlocks new advances for sustainable development.
4) We highlight the remaining challenges in TinyML towards its worldwide deployment in different research fields.

The rest of this paper is organized as follows. Section I discusses existing surveys, highlights their contributions, and the main gap we fill with the current survey. It also presents the scope of this survey and the research methodology. Section II presents an overview of TinyML. Section III presents environmental TinyML applications. Section IV focuses on recent TinyML applications in healthcare. Section V covers recent advances of TinyML in smart farming. Section VI presents TinyML use for anomaly detection. Section VI presents the remaining challenges in TinyML, open research questions, and future research directions. Finally, Section VII concludes the paper. For ease of use, the acronyms are summarized in Table 5.

A. EXISTING SURVEYS

In this section, we explore existing survey specific publications to TinyML. As of the writing of this paper, there are a few surveys exploring TinyML. This is mainly due to the fact that TinyML is relatively a new research topic. According to [19], the year 2019 was the year where TinyML was discussed for the first time in research publications. In Table 1, we order the existing surveys by publication year. The table shows a comparison of existing surveys in terms of the topics covered: Benefits of TinyML, model compression techniques for TinyML, TinyML frameworks and inference engine, TinyML Hardware, taxonomy of the main TinyML applications, TinyML in the environment, smart farming, anomaly detection, healthcare and challenges. The table lists the depth in which each topic was addressed (covered, partially covered). It also includes the number of references and their year of publication, cited in each survey paper. We can categorize the main topics surveyed as follows: surveys that discuss how TinyML can optimize ML models to bring intelligence and autonomy to devices in specific field such as, healthcare [20] and embedded vision [21]. Those that focused on general aspects of TinyML implementations. These include, 1) the benefits, 2) use cases in TinyML, 3) frameworks and inference engine, 4) hardware, 5) model compression techniques, 6) tools, and challenges and future roadmap [22], [23], [24], [25], [26]. A survey presented the challenges and direction of benchmarking in TinyML [27]. The work in [21] explored existing TinyML engineering, workflow and its challenges specific to IoT embedded vision. In [28], the authors covered reformable TinyML, and listed existing workflows, deployments schemes and sectors affected by reformable TinyML.

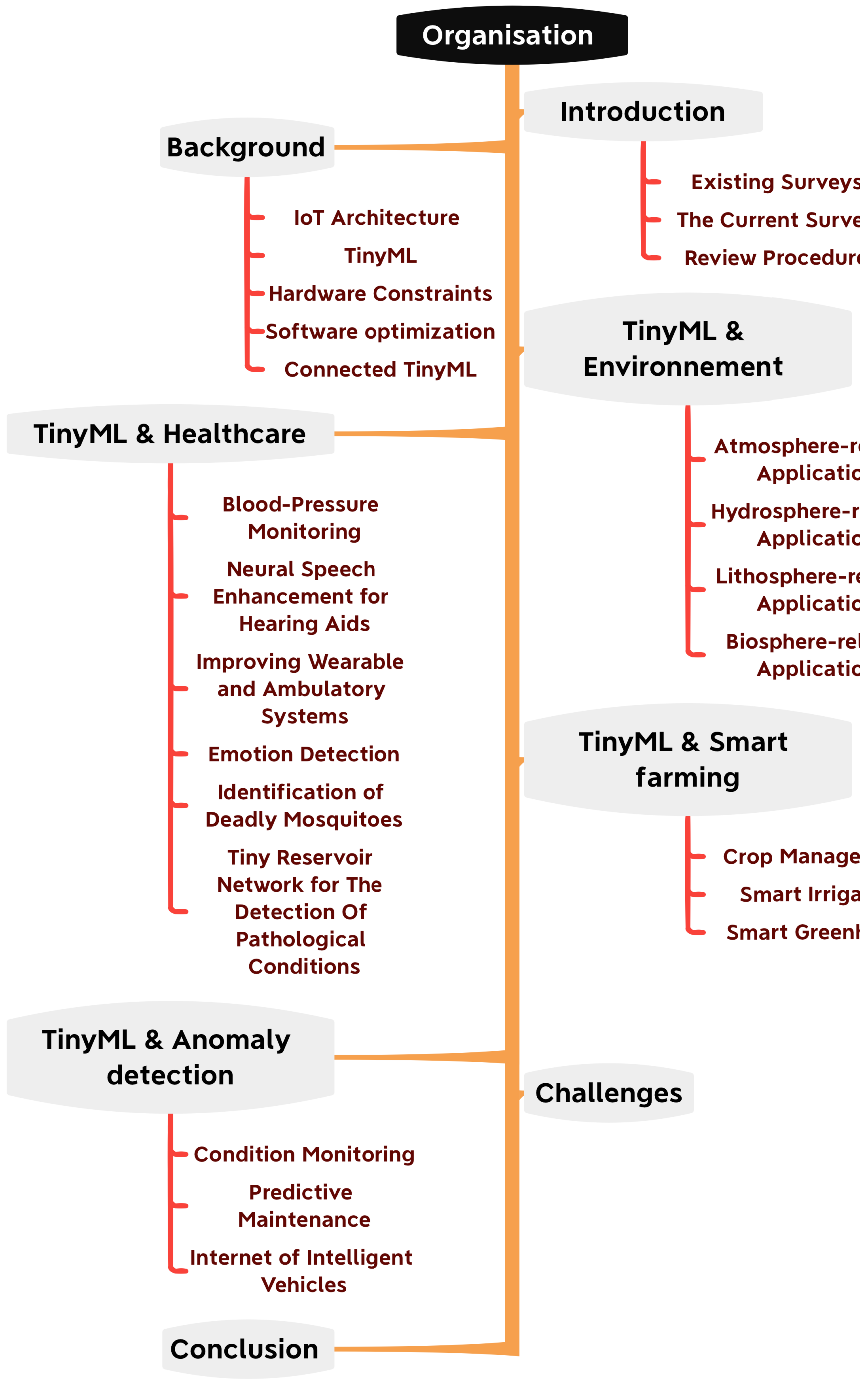

Organisation
Introduction
Existing Surveys
The Current Survey
Review Procedure
Background
IoT Architecture
TinyML
Hardware Constraints
Software optimization
Connected TinyML
TinyML & Environnement
Atmosphere-related Applications
Hydrosphere-related Applications
Lithosphere-related Applications
Biosphere-related Applications
TinyML & Healthcare
Blood-Pressure Monitoring
Neural Speech Enhancement for Hearing Aids
Improving Wearable and Ambulatory Systems
Emotion Detection
Identification of Deadly Mosquitoes
Tiny Reservoir Network for The Detection Of Pathological Conditions
TinyML & Smart farming
Crop Management
Smart Irrigation
Smart Greenhouse
TinyML & Anomaly detection
Condition Monitoring
Predictive Maintenance
Internet of Intelligent Vehicles
Challenges
Conclusion

FIGURE 1. Organization of the survey.

The study in [29] presented a clear and complete closed-loop workflow for the development and deployment of ML models for MCUs. It outlined each step in the workflow and provided both qualitative and numerical insights.

All of these surveys have reported the same promising benefits of TinyML including energy efficiency, low cost, data integrity, privacy, security, and latency. They also reported the same remaining challenges including limitation of existing benchmarks, hardware limitations when it comes to large amount of data, software and hardware co-design which is specifically addressed in [25]. However, existing surveys have not focused on the sectors where TinyML has a major impact, such as healthcare, environment, smart farming, and anomaly detection. Most of the surveys demonstrated the implementation of TinyML in different use cases, such as gesture recognition, speech recognition, keyword spotting, image recognition, and visual wake words. Our survey attempts to address these limitations which we discuss in detail in the following section.

B. THE CURRENT SURVEY

Our survey defines TinyML and presents its potential benefits in different sectors. In addition, we report the current advances and challenges in applying tinyML. We explore in depth existing contributions to critical sectors where the use of TinyML brings a significant impact. To the best of our knowledge, this survey is the first to include a taxonomy of the main TinyML applications. The main sectors that we survey are, the environment, smart farming, healthcare and anomaly detection. The choice of these sectors was driven by the amount of existing publications in these fields. The potential of TinyML revolutionizing these specific sectors as well as others can be sensed by the detailed overview of the existing work that we share in this survey. We believe this survey is an important added value to the research field as well as a valuable reference to researchers.

C. REVIEW PROCEDURE

The research procedure used in this survey is a Systematic Literature Review (SLR) [30], which consists of the following steps: defining the research questions, retrieving the literature, evaluating the literature, extracting the data, and finally presenting and discussing the results. Our aim is to present a fair evaluation of existing works on TinyML. To this aim, we used different digital libraries: IEEE Xplore Digital Library, Science Direct, and Springer Link. We also used Google Scholar for bibliographic databases. Different words and acronyms have been used in our research, as : TinyML, Tiny-ML, embedded, tiny machine learning, embedded machine learning, edge intelligence, and TinyDL. We noted the existence of some research papers dealing with TinyML as the main topic without mentioning this term in the title neither in the keywords [31], [32], [33].

The scope of our work in the current survey was guided by answering the three following research questions:

1) What are the new AI related publications that use non traditional computing resources and use mainly tiny devices?
2) What are the smart solutions proposed by TinyML developers and researchers that mitigate the limitations of existing AI based approaches?
3) What are the trending topics in TinyML, remaining challenges, and future research directions?

II. BACKGROUND

A. IoT ARCHITECHURE

The term IoT has been introduced for the first time back in 1999 by Kevin Ashton at MIT's Auto-ID Center to refer to a network that not only connects computers but extends this connection to any other device or thing [34]. Today, IoT is a huge network that connects various devices including computers, smartphones, drones to wearable and autonomous vehicles. IoT devices are getting more complex as machine learning models are integrated, allowing them to be more intelligent. Because of the vast quantity of data that can be acquired from IoT components, the coupling of machine learning with IoT leads to overall development and better intelligence of IoT devices. The IoT architecture can be broken down into four fundamental layers [35]: 1) the perception layer: it is composed of sensors that gather data and physical measurements, and actuators that execute tasks or actions based on sensor data; 2) the network or transport layer: it comprises the infrastructure for internet gateways and data acquisition systems, to transmit and gather data from different devices to an on-premise location; 3) the middleware or processing layer; it includes high-performance machines for data analysis and data storage; and 4) the application or service/interface layer; it grants users access to services and presents them through interfaces or APIs. Nevertheless, when implementing tasks such as machine learning models, this architecture may encounter difficulties such as high energy consumption. It might also need to maintain a steady connection at the network layer in order to preserve communication with devices and the cloud.

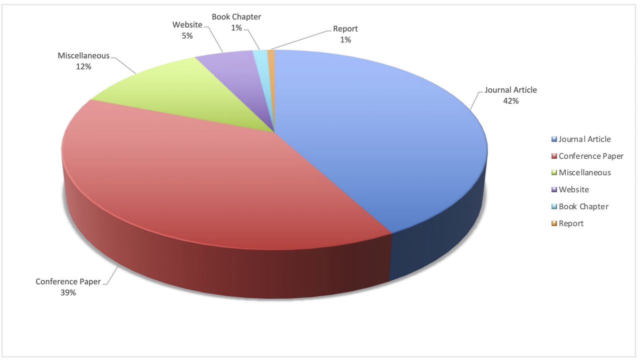


FIGURE 2. Types of used references.

The IoT architecture may be separated into three tiers based on computing capacity, according to [36]. This hierarchical separation increases efficiency and dependency among layers while also allowing for cost and resource optimization [37]. Cloud computing in the top layer is intended for high-level data processing and storage. However, the distance between cloud servers and the data source causes challenges such as sluggish decision-making and latency. In 2011, Fog computing has been introduced to overcome the increasing number of IoT devices, as a decentralized computer layer between the cloud and the sensing components [38]. This layer performs data collection, medium processing, and decision-making, decreasing the quantity of information coming to the cloud layer. Fog computing permits to cope with the huge amount of data to be processed in the cloud by performing task offloading. The lowest layer is edge

TABLE 1. Comparison of related survey papers. Annotations: "✓" indicates that the topic is covered, "≈" indicates that topic is partially covered, "✗" indicates that the topic is not covered.

Year of Publication	2020	2020	2021	2021	2021	2021	2022	2022	2022	2022	2022	2023
Reference number	[22]	[27]	[20]	[26]	[23]	[25]	[24]	[28]	[21]	[24]	[29]	Our Survey
Number of References	65	41	42	20	137	118	172	101	33	172	263	146
Year Range of Publication References	2013-2020	2016-2020	2015-2021	2015-2020	2019-2021	2014-2020	2015-2022	2012-2022	2015-2021	2015-2022	2011-2022	2015-2023
Benefits of TinyML	✓	✗	≈	≈	✓	✓	✓	✓	≈	✓	✓	✓
Model compression in TinyML	✗	✗	≈	✓	✓	✓	✗	✓	✗	≈	✓	✓
Connection and TinyML	≈	✗	✗	✗	✗	≈	✓	✗	✗	✓	✗	✓
TinyML Frameworks and Inference Engines	✓	≈	≈	✓	✓	✓	✓	≈	≈	✓	✓	✓
TinyML Hardwares	✗	≈	✗	✗	✓	✗	✓	≈	≈	✓	≈	✓
Taxonomy of the main TinyML applications	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
TinyML in Environment	✗	✗	✗	✗	≈	✗	≈	≈	✗	≈	✗	✓
TinyML in Smart Farming	≈	✗	✗	✗	✗	✗	≈	≈	✗	≈	✗	✓
TinyML in Anomaly detection	≈	✗	✗	≈	✗	✗	≈	≈	✗	≈	≈	✓
TinyML in Healthcare	≈	✗	✓	≈	≈	✗	≈	≈	✗	≈	≈	✓
TinyML Challenges	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
Number of Topics covered in the Survey	3 covered 3 partially mentioned	1 covered 2 partially mentioned	1 covered 2 partially covered	1 covered 2 partially mentioned	5 covered 2 partially mentioned	4 covered 1 partially mentioned	5 covered 4 partially mentioned	3 covered 3 partially mentioned	1 covered 3 partially mentioned	5 covered 4 partially covered	4 covered 3 partially covered	11 covered

computing, which puts data storage and computation closer to the source of the data (components of the sensing layer) [39]. The fog and edge layers increase bandwidth and reduce latency, allowing autonomous decision-making. These two layers improve the security and the privacy of the user data since it does not need to be sent outside the local network for further processing and storage. The edge layer is more dependable than the fog layer since the core tasks are handled locally, lowering device and network costs while providing quicker reaction time and offline availability. The computing paradigm is currently shifting from cloud computing to end-edge-cloud computing, which also supports AI evolving from a centralized AI deployment to a distributed artificial intelligence (DAI). This new paradigm is empowered by end-edge-cloud computing (EECC), where the heterogeneous computing capabilities of on-devices, edge, and cloud servers are managed to meet the requirements raised by resource-intensive and distributed AI computation [40].

B. TinyML

Artificial intelligence is one of the sectors touched by the new IoT paradigm, which advocates bringing intelligence to the edge layer. Edge AI evolved as a response to the limits of cloud-based AI, which is not necessarily appropriate for real-time applications and devices with limited processing power and bandwidth [41]. The term has become more popular in recent years as IoT and the number of linked devices have expanded, along with the requirement for low-latency and real-time decision making [23]. Mobile Machine Learning is one example of the use of intelligence on the edge layer, notably on mobile devices such as smartphones and tablets. Mobile ML techniques are meant to perform effectively on mobile devices' limited computing resources (few gigabytes of RAM memory) and battery restrictions. A Neural Processing Unit (NPU) was incorporated in current mobiles to enable the execution of ML algorithms. It is a customized processor designed to speed neural network workload calculation. NPU is substantially quicker than traditional processors such as CPU or GPU [42] in performing matrix multiplications, which are the essential operations of neural networks.

TinyML focuses on deploying compressed and optimized machine learning models on tiny, low-power devices such as battery-powered microcontrollers, and embedded systems. TinyML was inspired by Mobile ML's features [43] (low latency, resource limits, moderate cost) and its development grew as a result of the technical breakthrough in the field of IoT and MCUs. TinyML is successfully applied in various application areas(see Fig. 3) e.g., healthcare, agriculture, industrial IoT and environment. TinyML technology is driven by the necessity to integrate intelligence in a wide range of applications that were previously not viable owing to the high power and computing needs of standard ML models as shown in Fig. 4:

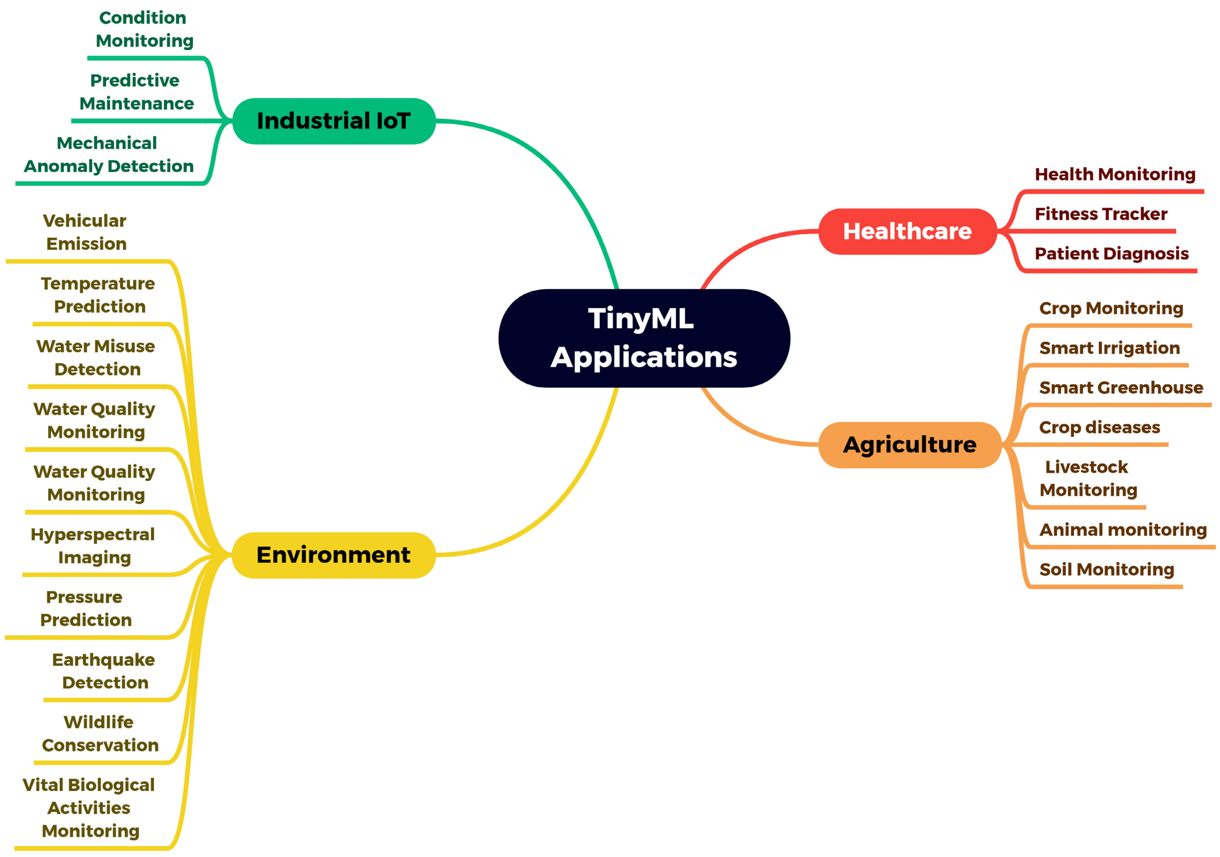


FIGURE 3. **Taxonomy of the main TinyML applications.**

1) REDUCING LATENCY

TinyML models can run on the device itself; thus the response time is much faster than sending the data to the cloud for processing. This is critical for applications that require real-time decisions, such as image and speech recognition. Figure 5 illustrates a comparison between the latency and accuracy of a ResNet-based model for Image Classification deployed on two TinyML-supported boards (NDP9120-EVL Board and GAP9 EVK Board), a traditional cloud-based deployment on Google Cloud, and Amazon servers. To conduct the benchmark test, MLPerf [44] and Stanford DAWN [45] were utilized. The results presented in Fig. 5 demonstrate that the deployment of models on TinyML systems significantly reduces latency, with a range of 0 to 5 ms, as compared to cloud-based machine learning models. Furthermore, TinyML systems offer high accuracy in their models, with only a slight decrease from 95% to 85% due to compression and optimization for use on restricted devices. These findings highlight the potential of TinyML to minimize latency while maintaining model accuracy, making it an attractive option for deploying machine learning models on resource-constrained devices.

2) OFFLINE CAPABILITY

TinyML models can run even when there is no Internet connection, whereas cloud ML models require such connectivity. This is important for applications that need to be deployed in areas with poor or no Internet access.

3) IMPROVING PRIVACY AND SECURITY

TinyML keeps the data on the device; thus, sensitive information does not have to be sent to the cloud for processing. This feature drastically changes the AI ethics picture by ensuring the user data privacy, which complies with data protection regulations.

4) LOW ENERGY CONSUMPTION

One significant part of decreasing energy usage in TinyML is to reduce the quantity of data that has to be transported and processed. Furthermore, TinyML algorithms are frequently intended to be computationally efficient, which can help in lowering the power consumption of the device on which they are executed. Other methods for reducing energy usage in TinyML systems include employing low-power components, designing for low voltage and battery-powered operations, and adopting sleep or idle modes.

5) REDUCING COST

TinyML models can save on costs associated with sending data to the cloud for processing and storage, such as bandwidth and storage costs. The low energy consumption is also

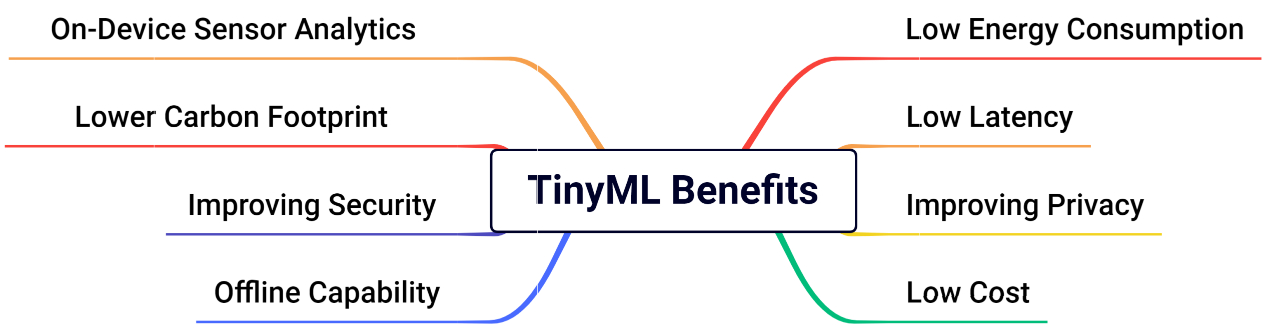


FIGURE 4. TinyML benefits.

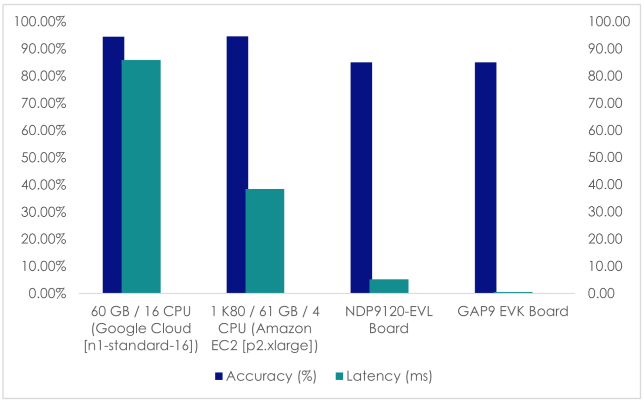


FIGURE 5. Latency and accuracy comparison.

an additional factor that implies the reduction of costs of TinyML.

TinyML enables IoT devices to become intelligent by allowing them to perform data analysis at the edge, leading to faster decision-making. Additionally, it can be applied to a variety of use cases to provide standalone ML services. Fig 6 describes the integration of TinyML in IoT workflow. In the IoT workflow, TinyML refers to the use of ML models that are small and resource-efficient enough to run on devices with limited computational capabilities, such as MCU found in IoT devices. These models are trained to perform specific tasks, such as image recognition, audio classification, or sensor data analysis. The data is collected by sensors on the IoT device and passed through the TinyML model for processing. The output of the model is then used for various purposes such as controlling the device, sending data to the edge, fog node or cloud for further analysis.

There are various proposed techniques in the literature that address the challenges associated with cloud-based systems, including alternative mechanisms for data analysis such as fog and edge computing. These technologies have been used to extend cloud services to the local area network, where data processing occurs at an IoT gateway or a fog node located close to the extreme edge device. Typically, each IoT device is registered onto a gateway or fog node to enable the extraction and processing of multi-sensor data. Although fog and edge computing provide low latency, the major drawback is that the wireless connection between the IoT device and the gateway or fog node must remain active for complex operations.

Table 2 compares TinyML with cloud, fog, and edge computing in terms of latency, data privacy, accuracy, power consumption, reliability, and memory consumption. The values presented in the table are derived from the benchmark conducted by MLCommons [44], an open engineering consortium dedicated to advancing machine learning innovation and fostering the development of cutting-edge ML models and systems. Benchmarking plays a crucial role in allowing ML enthusiasts to measure, compare, evaluate, and potentially enhance learning systems. It is important to note that the TinyML benchmark is still in the early stages of development, with the latest version being 1.0 [44]. The table shows that TinyML has advantages over other technologies in terms of latency, privacy, power consumption, memory consumption and reliability. However, it seems to have lower accuracy compared to these technologies due to the model optimization for deployment on low-constraint devices.

C. HARDWARE CONSTRAINTS

The growth of the Internet of Things industry has been fueled by advancements in artificial intelligence and machine learning, as well as improved connectivity. The growing demand for portable devices and the increasing use of microcontrollers for edge computing drive the microcontroller Market. According to verified market research [46], the microcontroller market size was valued at USD 348.83 million in 2021 and is expected to reach USD 992.80 million by 2030. A microcontroller is a compact integrated circuit designed to control a specific function in an embedded system and comprises a processor ranging from small and simple 4-bit processors to complex 32-bit, memory, input/output (I/O) peripherals, potentially with hardware accelerators, sensors, and connectivity. MCUs are generally small, low-cost and energy-saving.

The market has seen an emergence of the new 32bits generation of IoT-ready microcontrollers [46]. This emergence has

TABLE 2. Comparison of TinyML performances against existing technologies.

Technologies	Latency	Privacy considera-tion	Accuracy	Power Consumption	Reliability	Memory Consumption
Cloud Computing	10-500 ms	Very Low	86-94%	50-1000 W	Depends on the uninterrupted internet connectivity	GBs to TBs
Fog Computing	5-400 ms	Low	85-93%	10-100 W	Depends on active wireless connectivity	MBs to GBs
Edge computing	0.70-350 ms	Low	80-90%	1-10 W	Depends on active wireless connectivity	Few KB
TinyML	0.18-300 ms	Very High	80-90%	25-300 mW	Does not relay on network connectivity	Few KB

drustically transformed edge computing. Because of support for single instruction multiple data (SIMD) and digital signal processing (DSP) instructions, Cortex-M-based devices may now perform previously unachievable tasks. MCUs also include on-chip SRAM and embedded Flash; thus models that can fit within the memory limits are free from the costly DRAM accesses that limit classical ML. The broad acceptance and implementation of TinyML relies on the capabilities of these platforms. While general-purpose MCUs provide flexibility, specialized hardware provides the best TinyML performance efficiency. These customized devices can attain performance in the one micro Joule per inference level, pushing the limits of machine learning to the ultra-low power end of TinyML processors [27].

Deep learning hardware accelerators are specialized chips or circuits designed to improve the performance and efficiency of neural networks. These hardware accelerators are crucial for deploying deep learning models on TinyML as they provide parallel processing capabilities and optimized data flow to reduce computation time, energy consumption, and memory usage. Authors in [47] present a convolutional neural network accelerator (HBDCA) that uses a high-accuracy block random access memory (BRAM)-aware FPGA structure. For quantization, the HBDCA incorporates TensorFlow Lite (TFL) with 8-bit per-layer activation. The toolchain enables BRAM content to be updated without the need for re-synthesis or re-implementation. The HBDCA supports multiple kernel-level parallelism and uses spatial and temporal mechanisms to minimize memory access. Keras TensorFlow Lite is used to train the toolchain, and FPGA resources are used to find the best hardware configuration. The toolchain's final output is a memory map information file, which is generated at the end of the flow. The toolchain is designed for TinyML environments and provides a high-accuracy workflow that does not require re-implementation. The GrAI One accelerator which is a platform for sparsity-aware computing in neural networks that attempts to decrease computation overload and improve energy usage. The platform focuses on connection, space, time, and activation sparsity. The GrAI One accelerator processes network events on chip and saves weights in local SRAM. The neurons carry out basic neural and arithmetic operations, and their states are stored in local SRAM. Based on a synapse table, the platform generates events and neural model values. The sparsity-aware technique is similar to TinyML models and has the potential to save energy in neural network calculations [23].

Additionally, another technique for optimizing machine learning workloads is based on the usage of Tensor Processing Units (TPU's). A TPU is a custom-built AI accelerator designed by Google to perform highly efficient matrix calculations: the foundation of many machine learning algorithms. TPUs may greatly accelerate ML model training and inference, lowering the time and expense necessary to create correct models. Edge TPU is a form of TPU developed by Google intended exclusively to run TinyML models with great performance and minimal power consumption. The Edge TPU is an ASIC (Application-Specific Integrated Circuit) that can conduct real-time inference in conjunction with edge devices such as the Raspberry Pi and Coral Dev Board. The Edge TPU is a low-cost solution for running TinyML models on devices at the network's edge, allowing these devices to do complicated ML tasks with minimal energy consuption, and excellent accuracy that is significantly better than traditional CPUs, according to the [48]. The edge TPU has demonstrated the capacity to run cutting-edge mobile vision models such as mobilenet v2 at around 400 FPS while being power efficient [49].

Table 3 compares various TinyML devices, in terms of processor, CPU clock frequency, flash memory, SRAM size, power consumption or voltage, connectivity, sensors, product developer, and price. The majority of hardware boards use the ARM Cortex processor with CPU clock frequencies ranging from 100 MHz to 480 MHz. Most of the boards have WiFi and Bluetooth connectivity, as well as various on-board sensors such as light sensor, air pressure sensor, microphone, temperature sensor, humidity sensor, gyroscope, gesture sensor, accelerometer, air quality sensor and camera. Some TinyML

TABLE 3. Hardware platforms to support TinyML.

Hardware	Processor	CPU Clock	Flash Memory	SRAM Size	Sensors	Power	Connectivity	Organization	Price ($)
Alif Ensemble E7	Cortex-M55 with Ethos-U55 microNPUs	400MHz	4MB	4MB	Camera and microphone	1.71-3.6V	–	Alif Semiconductor	–
Arduino Nicla Vision	Dual Arm Cortex M7/M4	M7: 480MHz and M4: 240MHz	2MB	1MB	Camera, microphone and IMU	3.7V Li-po battery	WiFi and bluetooth	Arduino	115
Infineon CY8CKIT-062S2 Pioneer Kit	Arm Cortex M4	240MHz	2MB	1MB	Accelerometer, microphone	1.8-3.3 V	WiFi and bluetooth	Infineon	112
Seeed Grove Vision AI Module	Himax HX6537-A	400MHz	2MB	2MB	Camera, microphone and accelerometer	5V	–	Seed Studio	26
SiLabs Thunderboard Sense 2	Cortex-M4F	40MHz	1MB	256KB	Accelerometer and microphone	3.3V	Bluetooth	Silicon Labs	20
SiLabs xG24 Dev Kit	Cortex-M33	78MHz	1.5MB	256KB	Accelerometer and microphone	3.3V	Bluetooth	Silicon Labs	79
ST B-L475E-IOT01A	Arm Cortex M4	240MHz	1MB	128KB	Humidity sensor, temperature sensor, accelerometer and microphone	3-5V	WiFi and bluetooth	STMicroelectronics	51
NUCLEO-H743ZI	ARM Cortex M7	480MHz	2MB	786KB	–	3–5V	Ethernet	STMicroelectronics	26
NUCLEO-U575ZI-Q	Arm Cortex M33	160MHz	2MB	786KB	–	3–5V	Ethernet	STMicroelectronics	23
Portenta H7 Lite	Dual Cortex M7+M4	M7: 480MHz and M4: 240MHz	16MB	8MB	Camera	3.7V Li-Po	WiFi and bluetooth	Arduino	80
NRF5340DK	Arm Cortex-M33	128MHz	1MB	512KB	–	1.7-5.0V	Bluetooth, NFC and Zigbee	Nordic	46
B-U585I-IOT02A	Arm Cortex-M33	160MHz	2MB	786KB	Microphone, temperature sensor, humidity sensor, magnetometer, accelerometer, gyroscope, pressure and gesture detection	–	WiFi and bluetooth	STMicroelectronics	63
CY8CPROTO-062-4343w	Arm Cortex-M4	150MHz	2MB	1MB	Microphone	1.8-3.3	WiFi and bluetooth	Infineon	35
DISCO-F746NG	Arm Cortex-M7	216MHz	1MB	320KB	microphone	1.7-3.6 V	Ethernet	STMicroelectronics	55
NDP9120-EVL	Arm Cortex M0	100MHz	–	48KB	–	–	–	Syntiant	–
Coral Dev Board Micro	Cortex-M7, Cortex-M4 and Coral edge TPU	M7: 480MHz and M4:240MHz	128MB	64MB	Camera and microphone	5V	–	Coral	79

devices also incorporate hardware accelerators, such as the Coral Edge TPU ML.

D. SOFTWARE OPTIMIZATION

TinyML relies mostly on software, which enables the deployment of machine learning models on resource-constrained hardware. Because software allows for the optimization of model size and computational needs, machine learning models may be operated on devices with limited processing power and memory. This enables the incorporation of machine learning capabilities into a wide range of devices, resulting in new applications and better end-user experiences.

Model compression is a strategy for reducing a machine learning model's size and processing needs. This method can result in a 20% to 30% decrease in memory space required

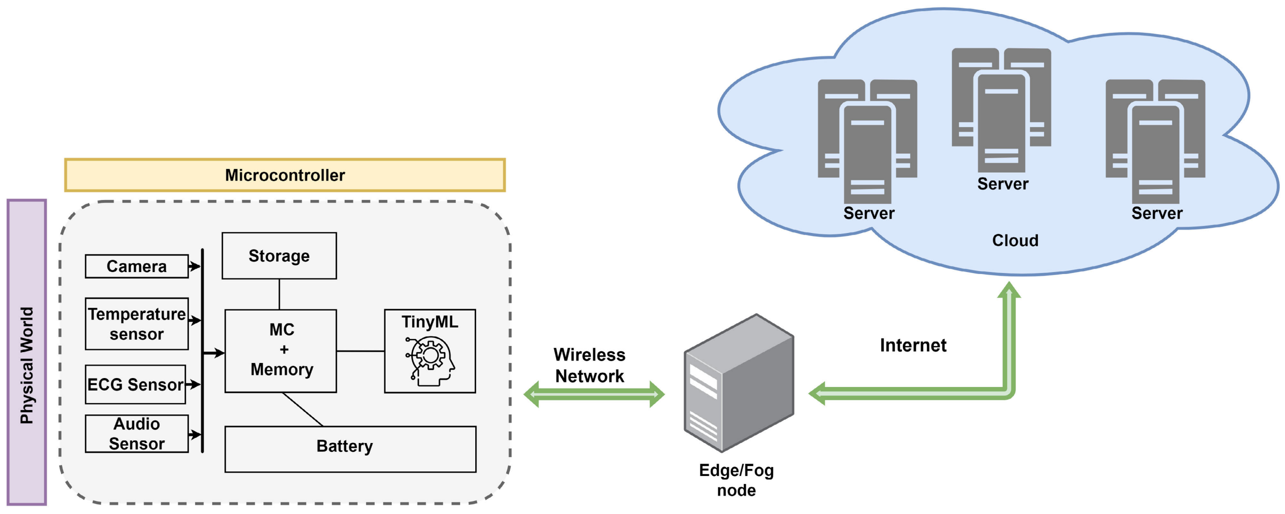


FIGURE 6. TinyML In IoT WorkFlow.

for network parameter storing. Several ways to compress a model exist:

Pruning: The pruning method begins with training the network and then selecting the key links by locating the weights that are greater than a specific threshold. Weights below this level are subsequently eliminated, resulting in a trimmed model. This trimmed model may not provide the same accuracy as the dense network, however retraining the residual weights can restore the accuracy. Pruning also aids in the removal of connections and neurons that have no input or output connections.

Quantization: Quantization is used to reduce the precision of the weights and activation is reduced from 32-bit or 64-bit floating-point numbers to 8-bit or lower fixed-point numbers. As fixed-point arithmetic is quicker and more energy-efficient than floating-point arithmetic, this decreases the model's memory footprint as well as the amount of processing required. Quantization can be done either during or after model training. The objective is to strike a compromise between model accuracy and the precision of the weights and activations, as lowering precision too much might result in severe accuracy loss.

Low-Rank Factorization: Low-rank factorization is a mathematical approach for approximating a high-dimensional matrix with a low-dimensional one while maintaining as much information as possible from the original matrix. The purpose of low-rank factorization in Machine Learning is to decompose a dense weight matrix into the product of two lower-dimensional matrices with lower ranks, hence reducing the matrix's dimensionality while keeping its structure and significant properties. As a result, the model is represented more compactly, with fewer parameters and is more computationally efficient.

Huffman Coding: Huffman coding is a lossless compression method, which means that compressed data can be precisely rebuilt to its original form with no information loss. Huffman coding works by assigning binary codes to each symbol in the data set, with shorter codes for frequently appearing symbols and longer codes for less frequently occurring symbols. The rationale behind this method is that symbols that appear more frequently in the data will take up less space if they are represented by shorter codes.

Knowledge Distillation: In addition to the Model compression techniques, Knowledge distillation is another important technique used in TinyML. Knowledge distillation is a machine learning approach in which a smaller, more compact model (referred to as the student model) is trained to mimic the outputs of a bigger, more accurate model (known as the teacher model). The student model's goal is to approximate the teacher model's predictions as closely as possible, and the idea behind knowledge distillation is that the smaller model can learn useful information about the problem from the teacher model, even if the student model is not as complex or accurate as the teacher model. The knowledge distillation training procedure consists of two steps: (1) training the teacher model on the original training data, and (2) training the student model with the teacher model's predictions as the target. The student model is trained using a loss function that takes into account both the accuracy of its own predictions and the similarity of its predictions to those of the instructor model.

However, the traditional techniques for compressing TinyML models mentioned above can lead to a significant loss of accuracy due to poor matrix characteristics resulting from high compression rates [50]. This prompted the development of Tiny neural networks, which are compact neural network models with a restricted number of parameters designed to function effectively on embedded devices with low processing resources. When compared to bigger, more complicated models, tiny neural networks are often trained

TABLE 4. Tiny neural networks.

Model name	Description	Applications	Number of parameters/Size
Tiny YOLO [51]	A compact version of the YOLO network. It reduces the model size by decomposing the complete convolution operation into depthwise convolution and pointwise convolution	Real-time object detection	60.5MB
SqueezeDet [52]	A fully CNN designed to be fast, energy efficient.It could achieve a state-of-the-art accuracy on the KITTI benchmark and its source code is open-source	Real-time object detection Autonomous driving	7.9 MB
Tiny SSD [53]	A single-shot object detection network that combines the efficiency of the Fire microarchitecture and the object detection performance of SSD. Tiny SSD has a smaller model size compared to Tiny YOLO and requires fewer computations while still achieving better object detection performance.	Real-time object detection	1.13 M parameters 2.3 MB
FastGRNN [54]	FastGRNN extends the residual connection to a gate by reusing the RNN matrices, achieving the accuracy of state-of-the-art gated RNNs (92.10% on Google-12) but with a smaller model.	Detecting utterances Human activity recognition	1-6 KB
Pelee-Text++ [55]	A lightweight neural network architecture for multi-lingual and multi-oriented scene text detection	Scene text detection	27 MB
Tiny-RainNet [56]	A model based on sequential radar echo maps using a combination of (CNNs) and bi-directional long short-term memory (BiLSTM). The model uses 10x10 sequential radar maps as inputs, which takes into account the influence of temporal-spatial meteorological conditions	Rainfall prediction	–
T-RECX [57]	An early-exit structure for tiny-CNNs to address the challenges in adding early-exits to state-of-the-art tiny-CNNs.	image classification keyword spotting visual wake word detection	–
EtinyNet [58]	A parameter-efficient tiny architecture is designed by introducing a dense linear depthwise block. A novel adaptive scale quantization (ASQ) method is proposed to further quantize the tiny models while retaining accuracy.	Object detetction	340KB
T-Net [59]	This network features a dual-stream information flow both inside and outside of the encoder-decoder pair and uses group convolutions to increase the width of the network and learn more low and intermediate-level features. T-Net also employs skip connections to minimize spatial information loss and uses a dice loss for pixel-wise classification.	Retinal vessel segmentation Skin lesion segmentation Digestive tract polyp segmentation	Depends on the filter configuration (23 K to 2.2 M parameters)

on a fraction of a larger dataset and employ a simpler design. This allows them to be trained and deployed more quickly and efficiently while retaining a high degree of precision. The table below 4 illustrates the several Tiny Neural Networks that exist, as well as applications, the number of parameters, and the size if available.

The technique of autonomously searching for the optimum neural network design for a particular job is referred to as NAS. NAS may be used in the context of TinyML to discover compact and efficient neural network architectures that can run on small embedded devices with low computing resources. NAS for TinyML seeks neural network designs that balance accuracy, model size, and computational performance in order to satisfy the restrictions of real-time deployment on small embedded devices. NAS algorithms may find structures utilizing a variety of optimization strategies, including reinforcement learning, evolutionary algorithms, and gradient-based methods. One example of a NAS framework for TinyML is DNAS. It stands for Differentiable Neural Architecture Search, which is a type of neural architecture search method for tiny machine learning. DNAS approaches differ from traditional NAS methods by using gradient-based optimization techniques to search for the best network architecture. The idea is to train a network architecture generator that outputs a candidate architecture, and then use gradient descent to optimize the architecture generator parameters such that the accuracy of the resulting architecture is improved. [60] Another framework is FPGA-aware graph neural architecture search (FGNAS). FGNAS performs a search for the best GNN architecture while considering hardware constraints. The framework is evaluated on benchmark datasets such as Cora, CiteCeer, and PubMed, and the results show that FGNAS has better capability in optimizing the accuracy of GNNs when their hardware implementation is specifically constrained. [61]

The deployment of models to embedded devices cannot currently support model training due to limited resources. Typically, models are trained on the cloud or on a more capable device before being distributed to the embedded device. However, applying machine learning algorithms on embedded devices creates difficulties, such as restricted computing power. There are three methods for deploying models [62]: hand coding, code generation, and ML interpreters. Hand coding provides for low-level optimizations but takes time; code generation creates optimized code but has portability difficulties. An ML interpreter is a tool used to implement machine learning algorithms on embedded devices with limited processing capabilities, including MCUs. It is part of a framework that includes tools and software libraries and calls individual kernels.

Aside from the interpreter, a TinyML framework typically includes TinyML libraries and tools for data processing, as well as a Tiny Inference Engine, which is a low-level software library or hardware accelerator designed to efficiently perform the computation required for machine learning inferences. Table 5 shows some TinyML inference engines along with their supported platform and training library. Overall, the Tiny Inference Engine provides the computing capabilities needed to execute the models, while the TinyML Interpreter handles model execution. The development tools for constructing and testing machine learning models, such as data preparation tools, model training and validation tools, and performance profiling tools, are also included in the frameworks.

A TinyML Framework's goal is to provide a comprehensive solution for constructing and deploying machine learning models on low-power devices, making it easier for developers to create edge computing applications. TinyML Frameworks serve to guarantee that machine learning models operate quickly and successfully on small, low-power devices by providing a comprehensive collection of tools and technologies. Among these frameworks we can cite:

TensorFlow Lite(TFL) [63] is a Google open-source deep learning framework designed for inference on embedded devices. It is made up of two primary parts: the Converter and the Interpreter. The TensorFlow Converter is used to convert TensorFlow code into a compressed flat buffer (.tflite), shrink the size of the model, and optimize the code with minimal accuracy loss. TFL currently supports quantization, pruning and clustering. TensorFlow Lite Micro(TFLM) [64] is an extension of TFL designed to execute machine learning models on 32-bit microcontrollers with limited memory (just a few kB). It's been successfully ported to a variety of processors, including the Arm Cortex-M Series and the ESP32. The core runtime just fits in 16 KB on an Arm Cortex M3 and is capable of executing many basic models. It operates independently of an operating system, eliminating the need for standard C or C++ libraries or dynamic memory allocation [64].

Edge Impulse [65] is a cloud-based solution that facilitates the creation and deployment of machine learning models for TinyML device, from collecting data using IoT devices, to extracting features, training models, and finally deploying and optimizing the models for TinyML devices. The trained models can run on various edge devices, like microcontrollers, single-board computers, and embedded systems. It employs the EON compiler for model deployment and also supports TFLM. Edge Impulse utilises TensorFlow's Model Optimization Toolkit to quantize models, lowering the precision of their weights from float32 to int8 with minimal impact on accuracy [66]. The EON compiler [67] compiles the neural network model directly into C++ source code, reducing the amount of stored ML operators that are not in use. It has been demonstrated that the EON compiler can run the same network with 25% to 55% less SRAM and 35% less flash compared to TFLM [29]. Edge Impulse has also designed a novel machine learning algorithm named FOMO [68]. It is a ML model designed for highly resource-constrained devices. It enables object detection to count objects, determine their location within an image, and track multiple objects in real-time using far less processing power and memory compared to MobileNet SSD or YOLOv5 [68].

Arm NN is a Linux-based, open-source software framework for machine learning inference on embedded devices developed by the company Arm. The core of the framework is the CMSIS NN library, which is a collection of optimized neural network kernels designed for maximum performance and minimum memory footprint on Cortex-M processor cores. Arm NN makes use of fixed-point arithmetic, quantizing model parameters to either 8-bit or 16-bit integers for deployment to microcontrollers for inferencing. The framework leverages the processing architectures of Arm's popular Cortex-M series of microcontrollers to enhance execution time [62].

E. CONNECTED TinyML

TinyML algorithms are intended to be deployed on devices with minimal resources, such as micro-controllers. Achieving the connectivity of the tiny embedded device, while adhering to the given restrictions of low energy consumption, may be possible using Low-power wide-area network (LPWAN). This type of wireless network has various benefits over standard IoT system approaches such as Bluetooth (BLE), WLANs, IEEE 802.15.4-based protocols (Zigbee or 6LoWPAN), and cellular networks. Fig. 7 compares the networking technologies, used by IoT devices, in terms of power consumption, bandwidth, and range.

LPWANs employ low-power radio frequencies and operate at low data speeds. They are based on a cellular-like architecture, in which end-devices communicate directly with a central gateway or a base station allowing to handle

TABLE 5. TinyML inference engine.

Inference engine	Supported platform	Supported training library framework	Free
TensorFlow Lite Micro(Google)	Arm Cortex-M Series, ESP32 and Himax WE-I Plus	TensorFlow	✓
uTensor(ARM)	Arm Cortex-M Serie	TensorFlow	✓
MicroTVM(uTVM)	Arm Cortex-M Serie	TensorFlow, Pytroch and Keras	✓
STM32Cube.AI (STMicroelectronics)	STM32 ARM Cortex series (32bits)	TensorFlow, Scikit-Learn, Matlab, and PyTorch	✓
NanoEdge AI Stu-dio(STMicroelectronics)	STM32 ARM Cortex series (32bits)	–	✗
EloquentTinyML	Arduino boards	TensorFlow	✓
emlearn	ARM Cortex M (STM32), ESP32 and AVR Atmega	Keras and Scikit-Learn	✓
EON compiler(Edge impulse)	ARM Cortex M series, Arm Cortex A series and ESP32	TensorFlow and Scikit-Learn	✓

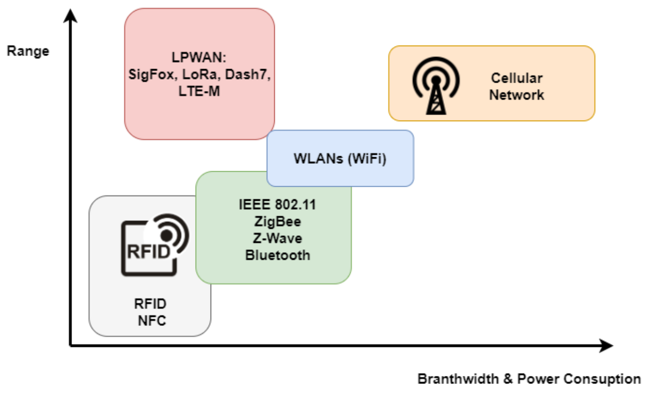

Range
LPWAN: SigFox, LoRa, Dash7, LTE-M
Cellular Network
WLANs (WiFi)
IEEE 802.11 ZigBee Z-Wave Bluetooth
RFID
RFID NFC
Branthwidth & Power Consuption

FIGURE 7. Different wireless networks used in embedded devices.

a large number of devices while consuming little power. Thus, LPWANs are well-suited for IoT applications in which devices must run for lengthy periods of time on batteries or other restricted power sources, and may be scattered across broad regions that sometimes can exceed dozens of kilometers [69]. LPWANs support applications with low-rate requirements where related devices can remain operational in the field for years without battery charging or replacement. They are generally considered as a type of Wireless Sensor and Actuator Networks (WSANs) [70]. LPWANs also offer great scalability and node density, making them suitable for ultra-dense scenarios like highly populated cities, and their network architecture does not require cooperation among end-devices or the use of routing protocols or node coordination. The research [24] proposed an example of employing LPWAN with TinyML in a federated learning (FL) scenario, where a variety of edge devices collaborate to develop a global model using only local copies of the data. This enables the model to be developed in a decentralized and distributed manner, eliminating the requirement for parties to share data.

There are other LPWAN-based wireless communication technologies available today, but LoraWAN is the most often used in TinyML projects (Table 1).

LoRaWAN: LoRaWAN is a media access control (MAC) protocol developed on top of LoRa, a communication technology that uses a modulation method called Chirp Spread Spectrum (CSS) to send data over vast distances while consuming minimal power. It is intended for wireless battery-powered Things in a regional, national, or worldwide network [71]. The utilization of license-free sub-gigahertz radio frequency channels is a significant characteristic of LoRaWAN. This allows the deployment of a large number of gateways, enabling extensive coverage and low-power communication with IoT devices. Sanchez-Iborra [72] evaluated the connectivity of a wearable device within a test region using a pre-existing LoRaWAN deployment. The results show that the device achieved good connectivity both indoor and outdoor locations, with a coverage shadow found in an area with dense vegetation and big buildings. The packet delivery ratio (PDR) obtained in the trials was over 92% in all cases, with better performance in the uplink direction due to the better noise factor of the gateway radio module as compared with that of the end-device. Overall, the results demonstrate that the TinyML device was able to achieve long-range connectivity with an efficient transmission technology.

Sigfox: It deploys its own base stations in multiple nations using its patented ultra-narrowband (UNB) technology in the unlicensed sub-GHz ISM frequencies (e.g. 915MHz in North America and 868MHz in Europe). End-devices communicate with these base stations through BPSK modulation in a 100 Hz ultra-narrow band with a maximum data rate of 100 bits/s [73]. Sigfox effectively utilises the frequency band and has very low noise levels by employing ultra-narrowband in the sub-GHz spectrum, resulting in low power consumption, great receiver sensitivity, and low-cost antenna design [74].

NB-IoT: To empower the Internet of Things, 3GPP developed Narrowband Internet of Things (NB-IoT) technology that may be used in both licensed and unauthorized bands. It is an air interface that is part of the LTE standard but has been reduced including handover, channel quality measures, carrier aggregation, and dual connection. [75] It is meant to promote long battery life and low-cost devices by utilizing one resource block of LTE networks, equivalent to 180 kHz in the frequency range. [76]

III. TINYML AND ENVIRONMENT

TinyML has the potential to play an important role in environmental problem solving. TinyML enables the deployment of intelligent devices capable of monitoring and collecting data on numerous environmental aspects such as air quality, water quality, and meteorological conditions. Indeed, these devices are designed to be resistant to noise and changes in data distribution. They may also be required to work in tough environments, such as extreme weather temperatures or places with limited or no access to power. TinyML devices may also be installed on a wide scale and offer real-time data on the state of the environment, allowing for a quick identification and resolution of problems.

A. ATMOSPHERE-RELATED APPLICATIONS

1) VEHICULAR EMISSION

The greenhouse effect, a key contributor to environmental concerns, is mostly generated by automobile emissions. These emissions contain significant volumes of carbon dioxide. To solve this issue, the authors in [77] employed TinyML and the on-board diagnostics system, available in most automobiles, to monitor the amount of CO2 emitted per liter and send the data to Typicality and Eccentricity Data Analytic(TEDA), an unsupervised anomaly detection algorithm. This method measures the quantity of CO2 created in grams per second; it saves the data on a microSD card for cloud processing by external systems through Bluetooth or 4G. The algorithm's recursive structure makes it time-efficient, with low computation and memory needs. The collected data can be used to guide the execution of public actions to mitigate the greenhouse impact.

2) TEMPERATURE PREDICTION

TinyML's temperature forecasting is a critical application in the realm of environmental monitoring. Accurate temperature forecasts may help in making decisions such as a) when to utilize energy in buildings; b) when to plant crops and c) what type of crops to plant. The authors in [78] proposed an approch to predict temperature values; it uses an edge computing technique and a TinyML-based device, such as the Arduino Nano 33 BLE Sense. They processed time-series data and made temperature forecasts using a multilayer perceptron, a sort of artificial neural network, with three distinct cost functions (Root Mean Square Error(RMSE), Mean Absolute Error(MAE), and R-squared).

3) PRESSURE PREDICTION

The authors in [79] employed Tiny Neural Networks and edge computing methodologies to obtain numerical weather forecasts, especially pressure forecasts. They employed a Tiny Deep Neural Network, a hybrid of Recurrent Neural Network and Convolutional Neural Network. They evaluated four distinct network versions and two different cost functions. The model was trained and validated using data from a recognized weather station; the findings revealed that the forecasts were highly accurate in matching the ground truth data. The model was then implemented on an STM32 microcontroller using a C-library that was tailored for simplicity and power economy. The output data was moved to a visual platform for analysis. Using edge computing for this task can drastically cut costs and compute complexity; it while also allowing for the production of online simulations based on atmospheric models.

B. HYDROSPHERE-RELATED APPLICATIONS

1) WATER QUALITY MONITORING

The authors in [80] proposed combining machine learning and TinyML-based devices(Raspberry Pi) to build a method to monitor and evaluate the water quality. Sensors capture data on numerous water quality variables(e.g., temperature, pH, and chemical material concentration) and communicate the data to a Raspberry Pi linked to a data center. The data is then analyzed by a machine learning model and transferred to the cloud for analysis. The authors emphasize the advantages of this technique, such as water saving and the capacity to monitor many elements that impact water quality.

The authors in [81] created prediction models for water quality utilizing multiple characteristics employing a graphical user interface (GUI) and a deep learning algorithm based on Artificial Neural Networks (ANNs). Their software properly predicted the quantity of toxic compounds in a Malaysian lake.

2) WATER MISUSE DETECTION

The authors in [82] developed a system for maintaining water reservoirs using a mix of Raspberry Pi, Arduino IDE, and ANN model. Sensors embedded in water faucets collect and send data using Raspberry Pi, which analyses it with a deep learning application. The application may then determine regular water usage as well as any leaks or waste. This has the potential to increase water conservation and decrease water waste.

3) WATER DISEASE DETECTION AND PREDICTION

Authors in [83] employed Edge AI to safeguard water supplies from dangerous contaminants such as cholera. They criticized the present Alkaline Peptone Wate (APW) technique and advocated employing edge computing to construct an experimental setup to monitor the physicochemical parameters of water in order to avoid Cholera outbreaks. Because of the absence of wireless connectivity in remote regions, they proposed to use an offline model that achieves 96% accuracy while conserving battery power. The device was eventually connected with on-tap inference and could monitor water safety metrics. The authors intend to expand and generalize the model in order to handle other illnesses that may contaminate water.

4) UNDERWATER IMAGING

Traditional underwater imaging systems require active power sources, which are not accessible in most underwater environments. These systems can be used to research marine species, climate change, marine geology, aquaculture farms, particulate organic carbon transport, and maritime archeology. Recent research has revealed that completely submerged, battery-free cameras and acoustic backscatter can be used for on-site wireless underwater imaging. However, the narrow bandwidth of the underwater acoustic channel makes picture acquisition and communication energy inefficient. To solve this issue, the authors in [84] created proposed a fish visual wake word (fishVWW) model based on wake word models. A wake word network is a sort of machine learning approach that is used to detect a certain word or phrase (the ''wake word'') in a stream of data. Wake word networks are often used in voice assistants such as Amazon's Alexa or Apple's Siri to allow users to activate the assistant by uttering the ''wake word'': ''Alexa'' or ''Hey Siri.'' The goal of a wake word network is to allow devices to operate on minimal power most of the time. This is accomplished by activating and processing audio data, only when the wake word is detected. Similarly, the fishVWW model was created for battery-free underwater cameras that can only take, compress, and send images when they detect a fish. The fishVWW model was tested on the STM32L476RG, a very low-power microcontroller (with 1 Mb Flash and 128 kbs SRAM). In order to decrease the amount of needless data transmitted, the authors [84] did investigate eliminating underwater artifacts from the image.

C. LITHOSPHERE-RELATED APPLICATIONS

1) HYPERSPECTRAL IMAGING (HSI)

TinyML has the ability to give useful insights in geology by analyzing sensor data in real time and making it more accessible, usable, and interpretable. One possible TinyML application in geology is the analysis and classification of various types of rocks and minerals using hyperspectral imaging (HSI). HSI collects detailed information on light reluctance at various wavelengths, which may be utilized to assess the composition and attributes of the materials being photographed. This data may be utilized to pinpoint specific rock formations, mineral deposits, and geologic features. HSI pictures, on the other hand, are often high-dimensional and need a significant amount of power and storage. To address these issues, the authors of the research [85] offer an algorithm-hardware co-design solution for 2D TinyML workloads. They propose inserting sophisticated calculations like convolutions and non-linear activation functions inside and the periphery pixel array. This decreases the need for data transfer between the image sensor and the accelerator for Convolutional Neural Networks (CNN) processing, resulting in lower energy and bandwidth requirements. The authors in [85] also offer two CNN models that are implemented using PIP (pixel-in-pixel), which yields considerable compression while reducing data rates and power consumption.

2) EARTHQUAKE DETECTION

Researchers in [86] focuse on earthquake detection as a vital initial step in Earthquake Early Warning (EEW) systems. The authors point out that in order to give real-time alerts, robust EEW systems must have high detection accuracy, low detection latency, and a high sensor density. They note that classic EEW systems rely on fixed sensor networks or, more recently, networks of mobile phones equipped with micro-electromechanical systems (MEMS) accelerometers. The authors suggest a new technique for global-scale earthquake detection and warning based on IoT edge devices with TinyML capabilities and always-on, always-connected stationary MEMS accelerometers. They explored and assessed deep learning ML algorithms for earthquake detection using a limited-resource Arduino Cortex M4 microcontroller (256 kB of RAM).

D. BIOSPHERE-RELATED APPLICATIONS

1) WILDLIFE CONSERVATION

Sensor technology is important in wildlife conservation efforts, particularly when it comes to protecting endangered species. Authors in [87] discuss the use of ML techniques, including on-animal sensors for tracking movement and bioacoustic sensors, to gather data about wildlife and the environment. These techniques can be used to improve our understanding of biodiversity and the environment. However, the paper also notes that there are limitations and challenges to be addressed, including issues with low latency and low capacity due to the large amount of data and the vastness of habitats.

Another application in the same scenario was presented in [88]. TinyML was used to small payload satellites weighing less than 180 kg, often known as SmallSats, to meet a specific conservation task. The study focuses primarily on the protection of sea turtles, who are under threat from factors such as uncontrolled fishing and sea pollution. The TinyML framework was used to assist conservation efforts by implementing state-of-the-art real-time vision-based TinyML support. The study's goal was to utilize this technology to track sea turtles in real time, giving real-time data on their position, activity, and population to assist enhance the efficiency and efficacy of conservation operations. This may be utilized not only for sea turtles, but also for other conservation duties such as unlawful hunting, logging, and animal welfare.

Hackster.io and Smart Parks organized an event named ElephantEdge [89] to encourage members of Hackster.io to

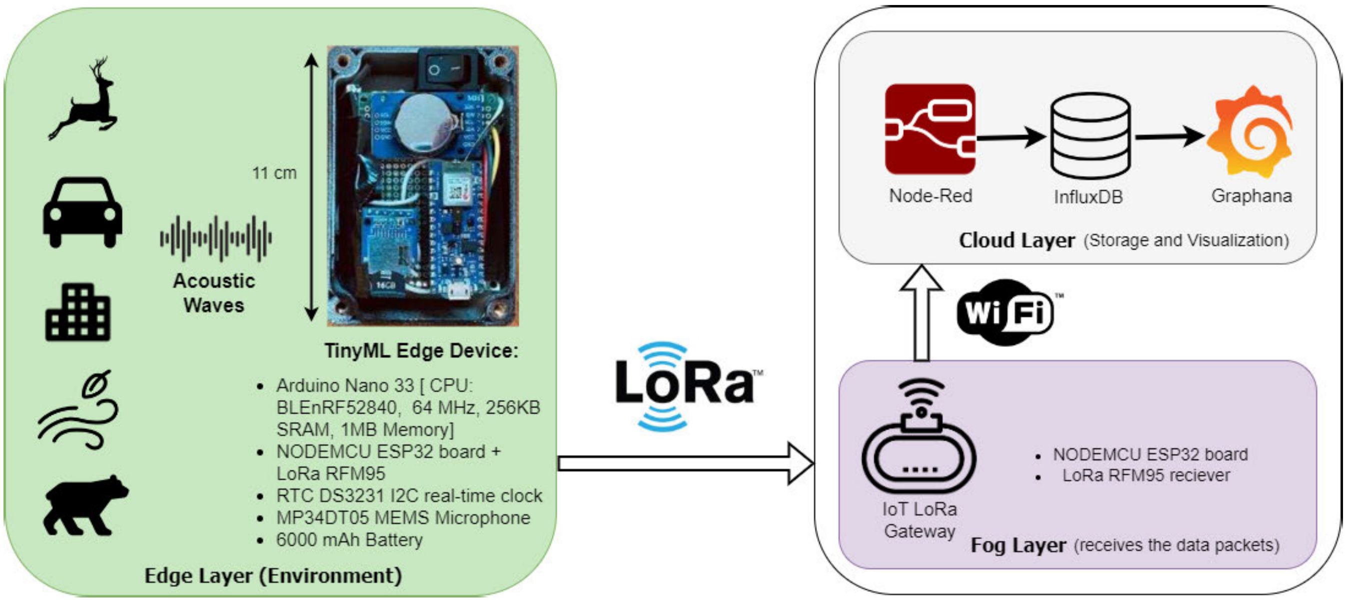

11 cm
Acoustic Waves
TinyML Edge Device:
• Arduino Nano 33 [CPU: BLEnRF52840, 64 MHz, 256KB SRAM, 1MB Memory]
• NODEMCU ESP32 board + LoRa RFM95
• RTC DS3231 I2C real-time clock
• MP34DT05 MEMS Microphone
• 6000 mAh Battery
Edge Layer (Environment)
LoRa
Node-Red
InfluxDB
Graphana
Cloud Layer (Storage and Visualization)
WiFi
IoT LoRa Gateway
• NODEMCU ESP32 board
• LoRa RFM95 reciever
Fog Layer (receives the data packets)

FIGURE 8. WASN-based soundscape monitoring system architecture.

develop TinyML models for tracking collars, with the goal of reducing the loss of elephants due to illegal ivory poaching, trophy hunting, human conflict, and environmental degradation. The authors in [90] proposed two TinyML models based on two different input such as, audio and accelerometer data(x,y,z). The first model detects poacher attacks based on the sound of kalashnikovs and bush arrow. The second model detects poacher attacks based on the elephant's physical activity. Both models were created using Edge Impulse Studio.

2) BIO-ACTIVITY MONITORING

Using TinyML can also help monitor vital biological activities of different species, providing valuable information about their behavior, movement, and ecosystem. Ultrasonic sensors can be used to track bat activity, as described in [91]. The sensors capture sound waves and generate a spectrogram, which is processed by Machine-Learning-based hardware to classify the signal and provide information about the bat's location, genus, and time of recording. This approach can give insights into biodiversity loss and other important ecological information.

3) SOUNDS CLASSIFICATION

TinyML might be useful in the subject of biophony, which is the study of the sounds generated by living creatures in their natural surroundings. It is feasible to analyze and interpret the complex audio patterns created by different kinds sources in a studied environment using TinyML models on small, low-power devices, and potentially utilise this information for conservation and environment management. TinyML was utilized in the article [92] to construct an in-situ soundscape monitoring system based on a wireless acoustic sensor network (WASN). The system was created to statistically anticipate soundscapes using an RF52840 32-bits-microcontroller on a low-cost edge device. The system's machine learning algorithms were able to identify noises into four categories: anthrophony (human sounds), traffic, biophony (animal sounds), and geophony (environmental sounds). The final model utilized in the system was a CNN based on mel-frequency cepstral coefficients (MFCCs), with an accuracy of 81.6%. The paper proposes a system architecture using three layers: the edge layer, which consists of sensors and a machine learning model running on the Arduino Nano 33 BLE Sense development board; the fog layer, which consists of an ESP32 and a LoRa transceiver acting as a gateway; and the cloud layer, which handles global storage as well as web-based application for visualization and analysis using Grafana and Node-Red. The development board is equipped with environmental sensors, a MEMS microphone, and the nRF52840 processor, and is capable of running TinyML and TensorFlow Lite. The sensor module also includes a real-time clock and a small battery for retaining date and time. Fig. 8. depicts the architecture of the soundscape monitoring system.

IV. TinyML AND HEALTHCARE

The development of TinyML has inevitably touched the sector of healthcare, which is substantial in every society worldwide. And since the human body can be considered as a source of signals emitted by a variety of organs, sensors can be deployed to collect these data and use it to mitigate dilemmas related to healthcare. The deployment of TinyML would fit to solve these issues as illustrated in table 6. Previous surveys [20], [23], [93] indicate that the integration of these edge devices would revolutionize the healthcare sector, and enhance the well-being of people. The latter can

TABLE 6. Summary of the applications of TinyML in healthcare.

Work	Purpose of Wearable Device	Type of Signal	Target Device	ML/DL Algorithm	Accuracy	Power Consumption	Latency	Memory Consumption
[99]	Detection and Recovery from FoGnin Parkinson's Disease	Accelerometer signals	ATMega2560 microcontroller	K-nearest neighbors, Decision Tree, Random Forest, Ada Boost, Support Vector Machine	93.58 - 97.65%	22.25 mW	44.5 ms	1.4 KB
[101]	Treatment of Epilepsy	Brain activity via Electroencephalography	STM32L476 ARM cortex-M4	Random Forest	93%	205 mW	27.9 ms	128 KB
[102]	Diagnosis of Focal Liver Lesion	Images obtained via Ultrasound Imaging	CAD transferred to a small memory footprint	Deep Neural Network	88%	100 - 200 mW	75 ms	14000 KB
[104]	Emotion Detection	Physiological signals	Shimmer GSR+	Random Forest, SVM, Logistic Regression	69 - 73.08%	100 - 300 mW	10 - 200 ms	50 - 200 KB
[100]	EDetection of Cardiac Arrhythmia	Singlelead Electrocardiogram	Chip nRF52 with an ARM's CortexM4 processing core	Convolutional Neural Network	78.4%	55.73 mW	0.34 ms	195.6 KB
[97]	Efficient Neural Speech Enhancement for Hearing Aids	Audio Data	STM32F746VE MCU	Recurrent Neural Network	61%	308 mW	2.39 ms	400 - 500 KB
[105]	Emotion Detection	Bioelectrical signal	MUSE and Shimmer GSR+	K-nearest neighbours	53.6 - 69.9%	100 - 500 mW	10 - 500 ms	50 - 200 KB
[103]	Epileptic Seizure Detection	physiological signals	ARM Cortex-M4	Support Vector Machine (SVM), Random Forest (RF), Extra Trees (ET), and AdaBoost (AB) classifiers	99.1%	21.89 - 25.41 mW	0.052 - 0.20 ms	50 - 100 KB

provide quality and reliable healthcare monitoring; in addition to the improvement of many health products. Moreover, the technology has the potential to enable new forms of medical monitoring, diagnostics, and therapy, improve the quality of care, and ultimately improve patient outcomes. Especially with the ability of TinyML to process data in real-time and in low profile devices. This opens new possibilities for healthcare professionals to monitor and treat patients in a more effective and efficient way [94]. The analysis of previous works allows us to amalgamate the general process adopted to apply TinyML in healthcare as illustrated in Fig. 9. The human body is the provider of biological signals that are detected by specialized sensors and later transmitted to the heart of the embedded system that consists of ML or DL algorithms interacting with a cloud environment. This combination allows the device to reveal the type of deficiency, as well as to monitor, predict or assist the patient. In this section, we explore promising TinyML-based solutions in the healthcare field, as summarized in table. 6., by shedding light on some of its trending applications.

A. BLOOD-PRESSURE MONITORING

One way to see how TinyML can find its place in healthcare is by investigating [95], where researchers apply the edge device to monitor high level blood pressure, called "TinyCare" [95]. The authors developed a cloud independent TinyML solution that merely relies on data obtained from patients. The authors in [95] adopt a systematic procedure to tackle the problem, starting by preprocessing the data based on physiological signals and then extracting the features. Their model used a variety of ML algorithms deployed on three Edge Devices: Arduino uno, ESP32 Wrover Board, and AdaFruit PyBadge. [95]'s methodology enabled to test a variety of models, not only with respect to accuracy, but also latency and complexity.

B. NEURAL SPEECH ENHANCEMENT FOR HEARING AIDS

Noise suppression and Hearing Aids are medical solutions used by people with damaged ears, they help in the decrease of listening difficulties, especially in noisy environments. Many models were developed through usage of RNN [96], the authors in [97] built on these models, but using TinyML.

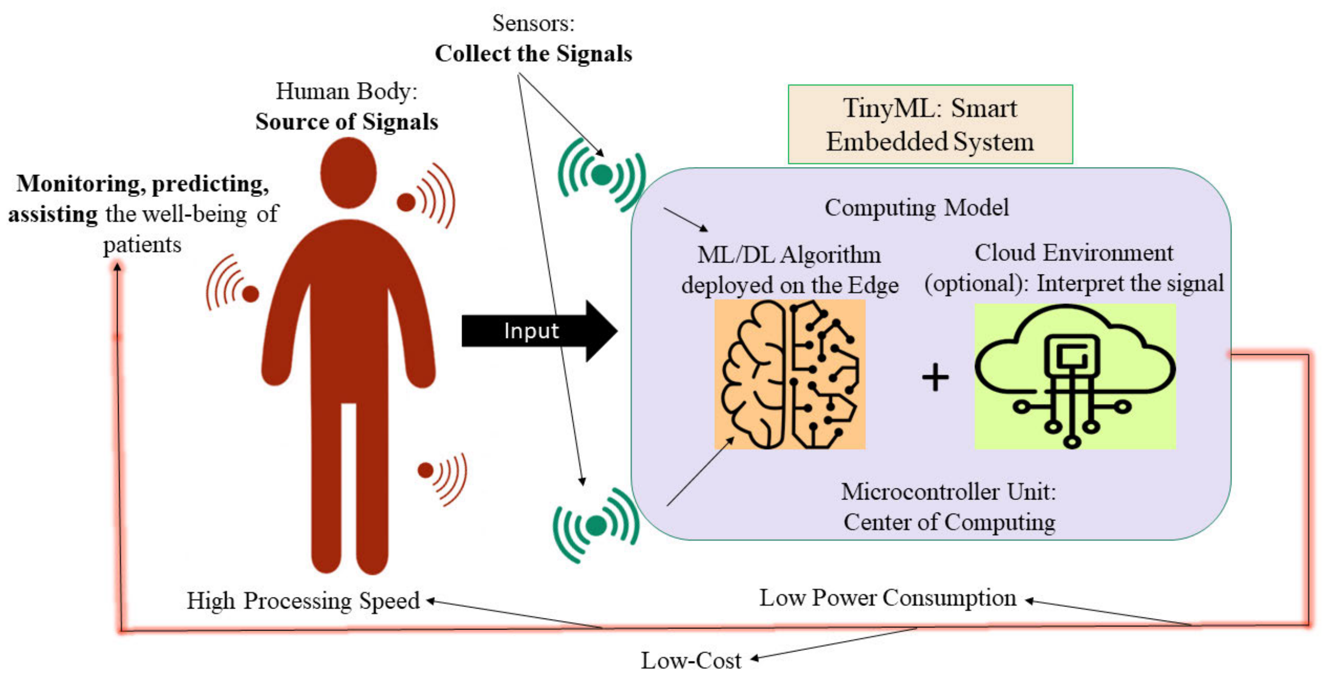

Sensors:
Collect the Signals
Human Body:
Source of Signals
Monitoring, predicting, assisting the well-being of patients
TinyML: Smart Embedded System
Computing Model
ML/DL Algorithm deployed on the Edge
Cloud Environment (optional): Interpret the signal
Input
+
Microcontroller Unit: Center of Computing
High Processing Speed
Low Power Consumption
Low-Cost

FIGURE 9. Demonstration of the general process adopted to apply TinyML technology in the healthcare sector.

The base of the hardware was chosen to be the Hearing Aid, and the authors used RNNs and pruning techniques to create the model that would enhance the hearing of speech. Authors in [97] acknowledged the constraints imposed by the edge device, since they required to train large neural networks with large data-sets due to the limited storage feature; nonetheless, the authors were able to achieve moderate satisfactory rating for their model.

C. IMPROVING WEARABLE AND AMBULATORY SYSTEMS

According to [98], TinyML can generate many embedded solutions and optimization techniques in the domain of wearable devices in Healthcare. Such contributions can be recognized in the detection and recovery from FoGnin Parkinson's patients, achieving high accuracy of 93.58% by a ML model deployed on ATMega2560 microcontroller that is able to balance between power consumption and processing speed [99]. The applications of TinyML in the healthcare domain can be further extended to the detection of Cardiac Arrhythmia,an irregular heartbeat caused by abnormal electrical activity in the heart. The detection process was boosted by the usage of low-powered microcontrollers as indicated by [100]. The authors in [100] relied on the CMSIS-NN library as a software tool to deploy their convolutional neural network. The authors reported a power efficiency of 1.64 GOps/s/W achieving a reasonable accuracy of 78.4% (including the implementation of CMSIS-NN). The treatment of epilepsy is a typical paradigm for the application of TinyML. The authors in [101] deploy an ultra low powered wearable device in treatment of this neurological deficiency through the detection of epileptic seizure. The experimental setup included an ML model (Random Forest) as a classification algorithm, that classifies signals after being deployed on STM32L476 ARM cortex-M4 microcontroller. The entire setup accomplish up o 40.87 hours of monitoring using a single battery charge (approximately a whole day of work); in addition to reduced amount of false-alarms that [101] worked on to obtain. are all domains to apply this technology. The authors in [102] and [103] introduce TinyML as a medical embedded system for the diagnosis of of focal liver lesion (FLL) using a DNN model trained on liver images that were priory obtained using ultrasound imaging. The entire process was implemented using computer aided diagnosis, which was later transfered in a small memory footprint that runs in an edge platform [102].

D. EMOTION DETECTION

Another substantial application of TinyML comes with emotion detection. The authors in [104] and [105] aimed to construct a smart wearable device to achieve the recognition of emotions through physiological signals emitted by the human body. For instance, in [104] from the respiratory belt, photoplethysmography (a measuring tool of variation of blood in a certain organ), and fingertip temperature, meanwhile, The authors in [105] used bioelectrical measurement technologies to measure parameters such as skin conductance, electroencephalographic, and heart rate signals. After deploying the data collect by sensors on trained ML models, [105] obtained an accuracy that ranges from 53.6% to 69.9%, in contrast, [104] achieved an accuracy of 69% to

73.08% which proves that these devices can be further harvested and polished in the identification of the emotional state of a person, which may contribute in the improvement of ergonomic conditions of people around the world.

E. IDENTIFICATION OF DEADLY MOSQUITOES

Applications in healthcare also involve the detection of harmful species that provoke disease and infectious illnesses. Works already exist before adopting TinyML in the process, such as [106], where they use CNN to classify audios captured through smartphones. However, the authors in [107] introduced TinyML technology by establishing an approach to detect hazardous mosquitoes using a model sensor that compiles wing beats to audios, which will be classified by a Tiny Embedded System that employs both Machine Learning and Edge Impulse Platform. The authors in [107] rely on Arduino Nano BLE 33 Sense-based prototype that is equipped by a trained ML model capable of classifying collected audio data of wing beats so as to identify the type of species in which the mosquito belongs to. The model of [107] achieves an accuracy of 88.3%, which is a good result especially with respect to the low-cost and low-energy consumption.

F. TINY RESERVOIR NETWORK FOR THE DETECTION OF PATHOLOGICAL CONDITIONS

One of the areas where tinyML can be applied is in the processing of electrocardiograms (ECGs). ECGs are signals used for measuring the electrical activity of the heart, which provides substantial information about the cardiovascular state of the human body. The authors in [108] aimed to reduce the complexity of this process by introducing a tinyML based ECGs that supports reservoir computing (e.g. a machine learning algorithm that uses a recurrent neural network (RNN) with a fixed architecture). In their model, they deployed their model low-power microcontroller unit able to process biological signals coming from the ECG for the recognition of a variety of pathological conditions. Their setup consumed less power, and acquired an accuracy of 95.4% with variance over the processed data of 0.001 [108]. The adoption of tinyML-based ECG analysis can unlock many new use cases, such as continuous monitoring and real-time feedback for medical doctors or patients.

V. TinyML AND SMART FARMING

According to the United Nations(UN) [109], the world population is expected to reach 9.8 billions by 2050. As consequence, the requirements of agricultural products are continually increasing to serve the future population. However, rapidly growing population, climate change, soil degradation and depleted of natural resources are all factors affect the food production. The need for an evolutionary agricultural to keep up with growing demand in food production is necessary.

Farmers, scientists, and agricultural industries turn to new technologies and solutions such as IoT, drones, ML, big data, cloud, fog and edge computing to transform traditional agriculture into sustainable, smart, efficient, and eco-friendly agriculture named Smart Agriculture (SA) or Smart Farming (SF) [110].

IoT plays a vital role in this transformation. Thanks to IoT solutions, farms can monitor the health of the crop and soil, detect any diseases can affect the plants, check the growth of plantation with drones and more. The electronics industry has seen significant advancements, leading to the availability of high-quality and cost-effective components such as MCUs, single-board computers, sensors, and radio transceivers. The newer generation of MCU are not only capable of performing standard sensing and control tasks, but also support complex operations such as running ML model. Moreover, contemporary radio technology has progressed to the point where long-range transmissions can be achieved with lower energy consumption.

The SF IoT network employs IoT devices to gather data on soil, crops, greenhouses, irrigation, and weather via cameras or sensors [110]. This data is transmitted to the cloud through Wireless Sensor Networks (WSN) and can be utilized by farmers to monitor crop health and identify diseases in the plants [110]. By analyzing and interpreting the data, farmers can comprehend the relationship between different agricultural factors, such as soil characteristics and climate variables, which aids in informed decisions and effective planning [111]. In this context, ML plays a crucial role in modeling the complex patterns present in the data, forming the backbone of decision support systems.

There are various architectures used in SF, depending on the specific application. Some common architectures include those with two layers(physical-edge), three layers (physical-edge-cloud) and four layers (physical-edge-fog-cloud). Many SF applications use three-layer architecture [112](see Fig. 10). The physical layer consists of sensors, actuators, and drones that collect data from the soil, animals, greenhouse, and weather. Data collected are transmitted to edge nodes using WSNs. The edge layer consists of computing devices that interpret and analyze data received from physical layer. The edge nodes have low or medium computing resources. The cloud layer represents the brain and data storage in SF architecture; it has a high storage capacity to save the data generated by various sensors. It also aggregates and draws insights to provide ML models for decision-making.

Complex ML algorithms require significant computing resources for effective execution, leading to the adoption of cloud computing; it has the capacity to handle large ML models with millions of parameters, as well as the high-speed processors and gigabytes of memory needed to run these models efficiently. However, this can be challenging in certain locations [113], such as in Africa [18], where internet bandwidth may not be sufficient to support the quick transmission to the cloud. Poor internet connection causes some issues such as huge latency, data loss and the reliability issues [112]. Moreover, frequent access to the cloud increase cyber-attack threats and decrease data protection [114].

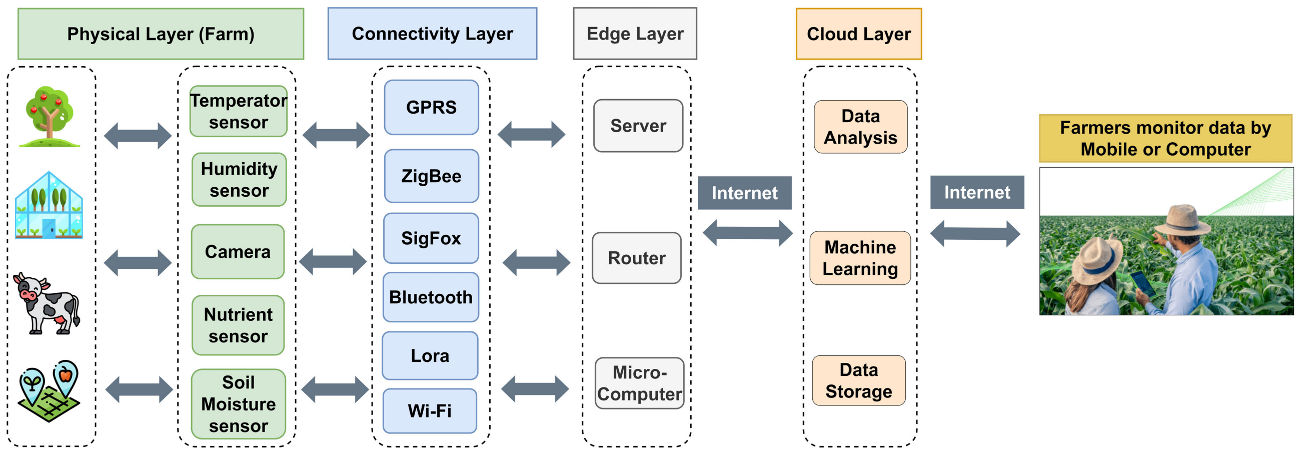


FIGURE 10. Smart farming architecture.

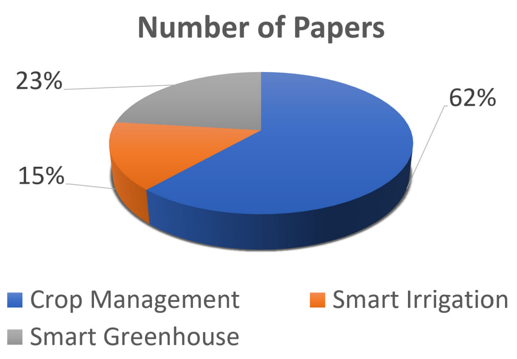


FIGURE 11. Number of articles on TinyML in smart farming.

To overcome the challenges posed by the cloud, some specialized techniques have been developed that bring the computation process closer to end devices, such as fog and edge computing. In recent years, the integration of TinyML in SF has also gained attention. TinyML allows sensor devices to perform ML tasks, such as monitoring crop health, detecting plant diseases, and predicting the best crops, without the need for the cloud. TinyML offers several advantages over cloud, fog, and edge computing in terms of privacy, security, latency, and energy consumption [112].

TinyML holds enormous potential in SF, especially in Africa where embedded systems and AI are currently underutilized [18]. PlantVillage [128] is one of these chances. It is an open-source project managed by Penn State University. The team in this project [128] has created the Nuru app to aid farmers in identifying and combating plant diseases. By using ML through TensorFlow Lite on mobile phones, the app provides real-time solutions without internet access, which is crucial for farmers in remote areas. The future development of the system will utilize TinyML and TensorFlow lite micro to install sensors across distant farms, resulting in improved tracking and analysis [24].

In this section, we explore promising TinyML-based solutions in SF field. As shown in Fig. 11, the most deployed case is crop management with 8 papers, smart greenhouse with 3 papers and smart irrigation with 2 papers. The table 7 summaries the application of TinyML works in SF in terms of area, dataset, classification algorithm, target device, framework, power consumption, memory consumption, latency and accuracy.

A. CROP MANAGEMENT

Crop management refers to the various techniques used to grow and maintain crops in an efficient and sustainable manner. This can include things like proper irrigation and fertilization, pest and disease management, and optimizing crop yields. TinyML could be used to gather data on factors such as soil moisture and temperature, as well as analyze this data to make predictions and provide insights that can improve crop yields and efficiency. The authors in [115] designed an embedded ML pipeline that helps farmers and scientists to monitor the health of the crop and its growth. This pipeline allows users to create an embedded ML that can be used for different plants in labs, greenhouses, farms or gardens. The first step of the pipeline is data collection, where the authors proposed best practices to collect data plants. The next step is training a Convolution Neural Network (CNN) for two cases: a) estimation of Leaf Area Index (LAI) and b) prediction of the plant growth stage. After training phase, the ML model is compressed and converted to TensorFlow Lite(TFLite) format to be deployed on MCU device. For testing, the authors have chosen Sony Spresense setup as target device.

The authors in [33] proposed a TinyML solution to detect drought stress in soybeans. The system is composed of a Raspberry Pi zero W and Sony IMX219 camera module. The Raspberry device runs CNN model on the captured image to detect crop drought stress and then sends prediction to a web platform. The CNN model was converted to TFLite format in order to be deployed on limited-resources device.

TABLE 7. Summary of the applications of TinyML in smart farming. Annotation: "–" is used to signify that authors did not provide information.

Reference	Application	Dataset	ML Algorithm	Target Device	Framework	Accuracy(%)	Power Consumption	Latency	Memory consumption
[115]	Monitor the health of the crop	Collecting data using Sony camera 5.11 MP	CNN	Sony spresense	TensorFlow	96.2	–	1544 ms	362.5 KB
[33]	Detection of drought stress in soybeans	Collecting data using Pi camera	CNN	Raspberry Pi zero W	TensorFlow	74	60-70 mW	172 ms	–
[116]	Detection grape leaf esca diseases	ESCA dataset [119]	CNN with DP decomposition	OpenMV Cam STM32H7	TensorFlow	98	–	100 ms	–
[118]	Detection coffee plant diseases	Coffee dataset [121]	CNN quantized with Q-format	STM32 F746G DISCOVERY kit	TensorFlow and X-Cube-AI	95	120-140 mW	299.60 ms	32.35 KB
[120]	Classification fruits and vegetables	fruit-360 dataset [123]	Mobile Net v1	ESP 32 Cam	TensorFlow	17 classes: 77 7 classes: 98	10-20 mW	51 ms	66.1 KB
[31]	Prediction of the best crop to grow	Crop-Recommender dataset [124]	Neural Network	STM32-F401CC	Edge impulse	92.2	10-20 mW	1 ms	1.7 KB
[94]	Detection and counting strawberries	Collecting data by sensors	FOMO	Arduino Portenta H7	Edge impulse	90	120-130 mW	148 ms	243,9 kB
[123]	Recognition type of gaz	Dataset generated from sensors	Multilayer Preceptron	NUCLEO-L476RG	X-CUBE-AI	70	20-33 mW	8.60 ms	42 kB
[124]	Micro-climate control of a strawberry agricultural greenhouse	Collecting data by sensors	Multilayer Preceptron	–	TensorFlow	96	–	–	–
[125]	Monitor the status of the plants for irrigation	Collecting data by sensors	Descion tree	Arduino UNO	emlearn4 and MicroMLGen5	99	10-20 mW	14.5 ms	80 KB
[32]	Analysis of soil photographs for irrigation	Kaggle soil dataset	VGG-19	–	TensorFlow	85	–	–	–
[126]	Read numerical water meters	Water meters dataset	Neural Network	ESP32-CAM	AutoKeras framework and TensorFlow for MCU	88	10-20 mW	20 ms	100 KB
[127]	Crop diseases detection	Tomato and potato dataset	CNN	Sony-CXD5602	–	99	2.63 mW	140 ms	370 KB

The authors in [116] proposed a low-power and real-time image detector for grape leaf esca diseases based on a compressed CNN model. Many compressing techniques such as, CP decomposition, tucker decomposition and tSVD are analyzed to chose the method with the best compression factor and accuracy. CP decomposition are chosen and it applied on CNN model. After training and validation of the model; it compressed with post-training quantization using TFLite to generate a model with 8bits. The compressed model is deployed on OpenMV Cam STM32H7. The device is mounted on an agricultural vehicle moving with a constant speed through the cultivation field.

The authors in [118] developed a TinyML CNN model aims to classify between infected and healthy coffee plant. This model aims to monitor the health of the coffee plants and prevents the propagation of the epidemic to the others. The authors compressed a model with Qm,n format using X-CUBE-AI tool. The TinyML model is deployed on STMicroelectronic "STM32F746G-disco" board connected to an STM32F4DISCAM module; both embedded in a box equipped with LEDs which are used to light coffee leaf while taking picture(see Fig. 12).

The authors in [120] built a prototype to identify and classify fruits and vegetables from images deployed on MCU

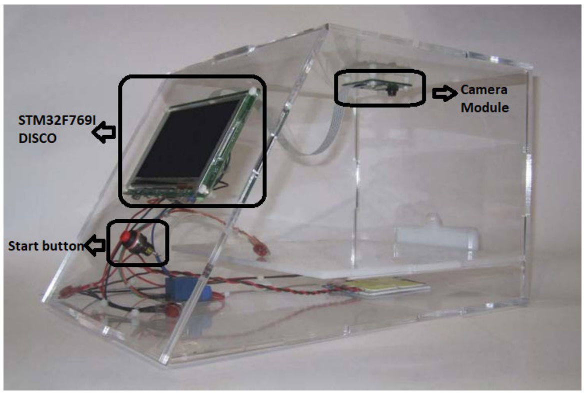


FIGURE 12. Coffee plant disease prototype [118].

ESP-32. This prototype helps farmers to identify fruits and vegetables in real-time. They performed experimentation on three models such as CNN, pre-trained MobileNet model v1 and v2 in order to choose a better model in terms of accuracy, inference time, RAM and flash occupation. These models were converted to TFLite format. Finally, the Mobile Net model v1 was chosen to be deployed on ESP-32.

The authors in [31] proposed a device(STM32F401CC) with a lightweight neural network model to predict the best crop to grow based on the observed soil parameters, such as nitrogen, phosphorus, potassium, soil PH, temperature, and humidity of the soil. The embedded ML model predicts five crops: beans, maize, lentil, peas, and watermelon. The device sends the prediction results to the cloud via GSM to notify farmers. The model is created and optimized via Edge Impulse platform.

The authors in [94] built a prototype to detect and count the number of strawberries in the image. This prototype aims to help farmers to know where and when strawberries need attention to harvest them or to apply fertilizer. The authors used the new ML model for constrained devices developed by Edge Impulse named FOMO (Faster Objects, More Objects). They used Edge Impulse cloud platform to design, train, validate, optimize and deploy ML model on Arduino Portenta H7.

The authors in [127] created a TinyML image classification deep neural network model deployed on smart camera to detect crop diseases. The study applying depthwise separable convolution module instead of standard convolution and squezze to reduce model parameter and computational complexity. The Sony-CXD5602 MCU device has been used as a target device. The model achieved a good performance in terms of time inference and accuracy on two publicly agricultural datasets such as, potato and tomato datasets.

B. SMART IRRIGATION

The authors in [32] proposed a system for agro-environmental management employing moisture sensors and real-time video analysis of soil photographs. The VGG-19 model is used to distinguish picture types and calculate the quantity of water required for irrigation based on the kind of crop planted. The model is trained on a Kaggle soil structure dataset and evaluated on a proprietary dataset.

The authors in [126] proposed a low-cost device based on the MCU ESP32-CAM that uses a camera to gather data from numerical water meters to monitor central pivot irrigation systems. The device runs a TinyML model to process the images in order to read the water meter; it then transmitted to a server using LoRaWAN. The TinyML model scored an accuracy of 88%.

C. SMART GREENHOUSE

A greenhouse is a structure designed to provide controlled environment for plants to grow. This controlled environment can include factors such as temperature, humidity, light, and nutrient levels, which can be adjusted to optimize crop growth. Greenhouses can be used to grow a wide variety of crops, including fruits, vegetables, and flowers, year-round, regardless of the outdoor weather conditions. TinyML can be used in greenhouse to optimize crop growth by using ML on low-power devices to gather data and make predictions. The authors in [123] proposed a low+cost device that aims to recognize different types of gas (NH3, CH4, N2O). This device helps farmers to monitor greenhouse gases coming from the soil. They chose an ANN algorithm to classify three types of gas. It was compressed and deployed on NUCLEO-L476RG device using X-CUBE-AI toolchain.

The authors in [124] developed a multi-label TinyML model based on Multi-Layer Perceptron(MLP) architecture for micro-climate control of a strawberry agricultural greenhouse. The model takes five parameters as input such as, temperature, humidity, soil humidity, solar illuminance and the air CO2 concentration. And then generates as output a five-dimensional vector where each binary numerical value is associated with one control action to be performed automatically within the greenhouse. The model scored an accuracy of 97% on the validation sets, and 96% on test set, with a number of 151 trainable parameters.

The authors in [125] proposed a system where IoT devices and edge nodes work together to monitor the status of the plants in a greenhouse to make decisions about the operation of sprinklers. Each IoT device uses its embedded TinyML model based on Decision Tree(DT) to determine the needs of the plants, and then sends this information to the edge node. The edge node collects decisions made by all the sensors for a specific sprinkler and uses an ML model to decide the final action (''no action'', ''irrigation'', or ''fertigation''). The study uses Arduino Uno board as a target device for end device and edge node. The TinyML model scored an accuracy of 99%.

VI. TinyML AND ANOMALY DETECTION

Anomaly detection(AD) or outlier detection refers to the techniques for identifying patterns or observations in data that do not conform to an expected behavior [129]. This techniques are used in variety of IoT smart applications [129] such as, smart city, smart monitoring and smart power management. Mostly of this applications used sensors as an input device. These devices generated a huge volume of data that are transmitted to the cloud servers for analysis, decision making and storage [129].

Many detection methods have been introduced in the literature to detect anomalies on data such as, geometrical, statistical, and ML [130]. Data-driven approach using ML algorithms and deep learning are becoming more and more popular as a way to identify abnormal situations [131]. It used to build models from large data generated by sensors to distinguish between ordinary and abnormal classes.

Cloud-based architectures have been used for many anomaly detection application [129]. An example of cloud based anomaly detection is condition monitoring [132]. The study used cloud computing to monitor the health condition of machines using vibration signals. The edge devices collect vibration data and transfer them to the cloud for storing. Cloud provides an ML model to detect machine's abnormal behavior. Analyzing and processing data in the cloud server poses some limitations such as, latency and security issues.

TinyML has the potential to play a big role in anomaly detection by using ML at the edge, allowing for real-time monitoring and identification of unusual patterns or behaviors without a connection to a more powerful computing device. In this section, we explore exciting work that integrate TinyML in AD as summarized in table.8 in terms of application, dataset, ML algorithm, target device, framework, accuracy, latency, power and memory consumption.

A. CONDITION MONITORING

The authors in [133] developed TinyML model deployed in STMicroelectronic STM32H743Z12 to detect anomalies in rotating machinery. The MCU device acquires vibration signals through an accelerometer, process it to extract features and store data on MCU to train locally an Auto-encoder ML model. The Auto-encoder model is trained sequentially by providing the training data in small batches at a time (4 batches of 100 samples). After training step, the model is stored in flash memory of MCU to be used during the inference. To determine whether an anomaly is present or not, unseen data is input to the model and the MCU calculates the MSE between the sample and its reconstruction. If the MSE is larger than the anomaly threshold, it is determined that an anomaly.

The authors in [134] describe a system for detecting anomalies in submersible pumps at wastewater management plants using a retrofitting kit with MCU. The kit includes temperature and vibration sensors, an ESP32DEVKIT MCU, and power line communication equipment installed in the terminal chamber of the pump. The MCU collects data from the sensors, processes it to extract features, and uses this data to train locally a Isolation Forest model to identify anomalies in the data stream. After training step, the MCU switches to inference model. Even during this mode, the MCU continues to process and learn to update the model. The model is developed using python 3 and libraries NumPy and SciPy. The authors opted for the MicroPython firmware as base firmware for the MCU.

The authors in [135] compared two types of TinyML models, an autoencoder and a variational autoencoder for mechanical anomaly detection in washing machines using an MCU device (Arduino Nano 33 BLE Sense). The MCU device collects accelerometer data (x, y, z) and runs the ML model to detect an imbalanced laundry load. To determine which ML model should be deployed on the MCU device, the authors evaluate models using accuracy, precision, and recall. The autoencoder model was chosen due to its performance in terms of accuracy. Both models were written in Python using TensorFlow library and then converted to TFLite format to be deployed on Arduino device. The results show around 92% accuracy and 90% precision.

B. PREDICTIVE MAINTENANCE

The authors in [136] created a TinyML model deployed on ESP-WROOM-32 MCU device to detect anomalies in thermal images; it sends data only when it detects an anomaly on a machine to the server using Message Queuing Telemetry Transport(MQTT) protocol. The MCU device runs a CNN model created by Keras and converted using tinymlgen library from EloquentArduino. The study was tested on hydraulic refrigeration system. The results show around 94% accuracy and 92% f1-score.

The authors in [137] proposed a online learning anomaly detection model named Deep Echo State Network(DeepESN) to monitor the reliability of water distribution systems; it adapts itself to the environmental changes and it can be deployed on a MCU. DeepESN is based on Recurrent Neural Network(RNN). To optimize the complexity of the models, the online learning can be made in two different ways: single iteration and batch decomposition. The model was tested on STM32H743ZI Nucleo Arm.

The authors in [138] proposed a Block based Binary Shallow Echo State Network (BBS-ESN) model which is a deeply quantized anomaly detector of oil leaks that happen in the wind turbines. This TinyML model uses DeepESN, binarized images and one-bit quantization of weights and activations function. The model was tested on STMicroelectronics NUCLEO-H743ZI device.

C. INTERNET OF INTELLIGENT VEHICLES

Authors in [139] proposed a MCU device attached to a vehicle to detect road anomalies such as potholes, bumps and obstacles. The study used Arduino Nano 33 IoT that collects accelerometer data and runs an unsupervised TinyML algorithm named TEDA. The results on real experiments (Asphalt pavement) show an accuracy of 99% and an f1-score of 0.76.

VII. NAVIGATING THE CHALLENGES: AN OVERVIEW OF THE CHALLENGES FACING TINYML

Investigating the challenges of TinyML technology is a crucial part in the current survey as they define the state of the

TABLE 8. Summary of the applications of TinyML in anomaly detection. Annotation: "–" is used to signify that authors do not provide information.

Reference	Application	Dataset	ML Algorithm	Target Device	Framework	Accuracy(%)	Power Consumption	Latency	Memory consumption
[133]	Detecting anomalies in rotating machinery	Trained on healthy machine data (unlabeled data)	Auto-encoder	STM32-H743Z12	X-CUBE-AI	99	51 mW	8.6 ms	50 KB
[134]	Detecting anomalies in submersible pumps at wastewater management plants	Dataset generated	Isolation Forest	ESP32-DEVKIT	MicroPython	88	20 mW	14.5 ms	80 KB
[135]	Detecting mechanical anomalies in washing machines	Trained on normal healthy washing machine data (unlabeled data)	Auto-encoder	Arduino Nano 33 BLE Sense	Keras and TensorFlow Lite	92	54.1 mW	11.7 ms	–
[136]	Detecting anomalies on industrial machines	Thermal Image dataset	CNN	ESP-WROOM-32	Eloquent-Arduino	94	26 mW	140 ms	100 KB
[137]	Monitoring the reliability of water distribution system	Skoltech Anomaly Benchmark(SKA B) dataset	Deep Echo State Network	STM32-H743ZI Nucleo Arm	X-CUBE-AI	90	51 mW	12.06 ms	42.29 KB
[138]	Detecting anomaly of oil leaks in the wind turbines	Oil leak dataset	Block based Binary Shallow Echo State Network	NUCLEO-H743ZI STMicro-electronics	X-CUBE-AI	81.3	791 mW	12.06 ms	129.2 KB
[139]	Detecting road anomalies	–	TEDA	Arduino Nano 33 BLE Sense	TensorFlow	98	54 mW	11 ms	–

art, open research questions, and possible directions. In spite of the potential benefits and advantages that TinyML offers, as ditailed in the aforementioned sections, it is important to understand and address the obstacles and challenges that may impede its progress. From technical limitations to societal concerns, these challenges must be carefully considered in order to fully realize the potential of this technology, and reinforce its capacities to fulfill further tasks in human society. In this section, we explore the various challenges that are currently facing the implementation and adoption of TinyML. We delve into the specific issues in more detail in an attempt to provide insight on possible solutions.

- *Evolution of the Environment:* The current TinyML solutions are based on offline learning. ML model is first trained on powerful machine or cloud and then deployed on tiny edge device. Edge devices cannot adapt themselves to the evolution of the environment because they cannot learn from the data. The ML model performance will consequently be dropped. This problem is known as concept drift [140]. On-device learning facilitates the transition from offline ML model training to updating it automatically using real-time data. Attempts already addressed this issue such as, the authors in [140] proposed a novel solution called TinyOL (TinyML with Online-Learning) allowing MCUs to learn on streaming data. This study is based on the concept of online learning. Online Learning follows a process of updating a model's parameters in real-time using new data as it becomes available, rather than using a fixed dataset. This allows the model to adapt to changes in the underlying data distribution and improve its performance over time. Other implementations of On-Device learning are presented on this paper [10].
- *Limited Memory:* The memory challenge in TinyML refers to the difficulties that arise from the limited amount of memory that is available on these devices. This challenge remains a major challenge and a trade-off between the model's performance and memory usage. These devices, such as sensors, wearables, and IoT devices, have limited computational resources and memory, which makes it difficult to run complex machine learning models. Larger models tend to be more accurate, but they also require more memory. This can lead to trade-offs between model accuracy and memory usage, which can result in models that are less accurate than desired. In addition, the increase of memory, which is

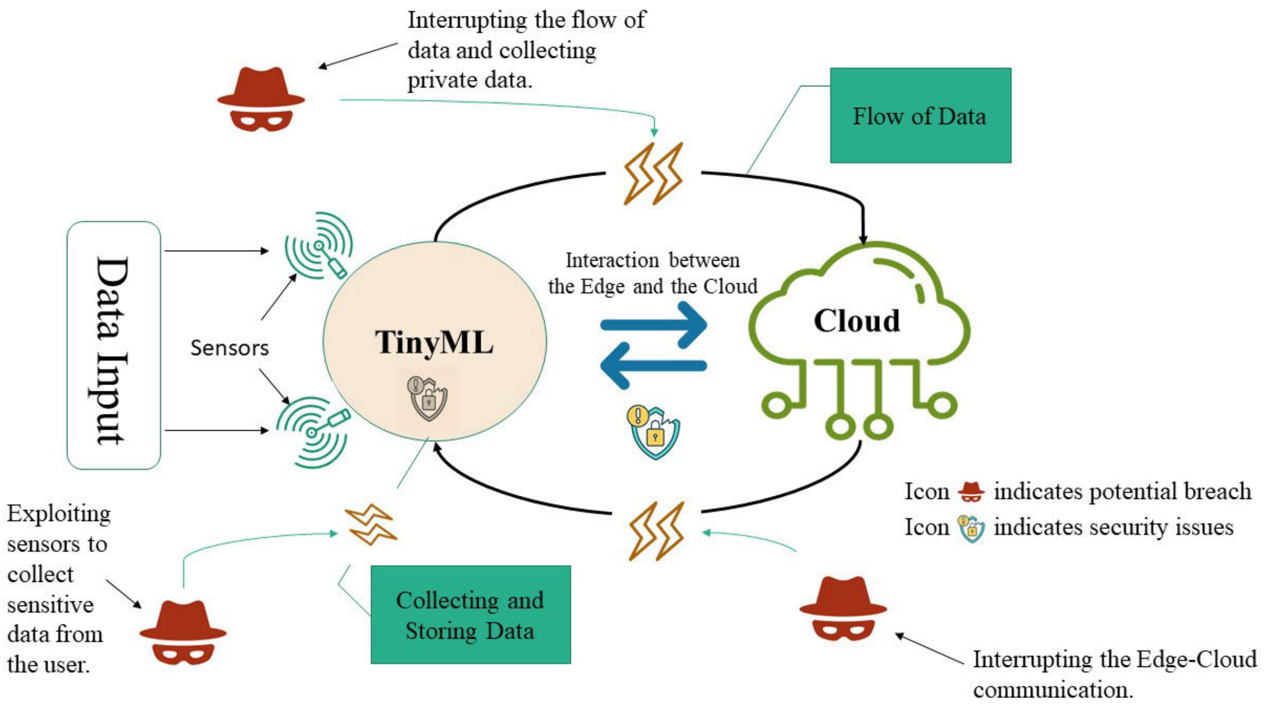

Interrupting the flow of data and collecting private data.
Flow of Data
Data Input
Sensors
TinyML
Interaction between the Edge and the Cloud
Cloud
Exploiting sensors to collect sensitive data from the user.
Collecting and Storing Data
Icon indicates potential breach
Icon indicates security issues
Interrupting the Edge-Cloud communication.

FIGURE 13. **Illustration of the security challenges facing TinyML, indicating potential security breaches within the process.**

a power-hungry component, yield to heavy power consumption which can affect the battery life of the edge devices. To avoid the memory constraint, data might be stored in a remote server and accessed on-demand, this can increase the latency and communication overhead. Some possible solutions exist via AI Model Efficiency Toolkit (AIMET) [141], which blends the AI trained algorithm with compression and quantization which leads to the optimization of the ML or DL model while maintaining the same accuracy.

- *Heterogeneity of Hardwares and Softwares:* which mainly includes the diversity in the devices and systems being used, each with their own unique software and features [23]. Which creates a challenge when it comes to managing and coordinating the various systems, as different software may not be compatible or may require different levels of resources in terms of the operating system and programming language; in addition to the varieties of levels of memory, processing power and storage. The heterogeneity of data is also worth mentioning. It mainly refers to the diversity and complexity of data that is captured and processed by small, low-power devices sensors. This can be a result of the inevitable noise that interferes with real data. Format and resolution also takes part in this heterogeneity as they hinder the ML algorithm to generalize the model. In addition to that, Tiny-ML algorithms requires an assemblage of systems, devices and software, including microcontrollers, sensors and the cloud environment, which may obstruct researchers wishing to exploit this technology by generating experimental complexity, which may also lead to discrepancies between frameworks [18].
- *Accuracy Drop:* As many researchers report a decrease in the accuracy of the ML model once imported on the edge. Especially that TinyML systems are typically designed to work low-power devices (e.g. wearables, edge devices, IoT devices) the limited available resources of these devices can make it challenging and difficult to achieve the same level of accuracy as larger systems. The limited amount of data that can be stored and processed on these devices can also lead to lower accuracy. Additionally, the compression of the algorithm may also yield to the reduction of the accuracy, further reducing accuracy. In [100] for instance, the team recorded an accuracy drop from 80% to 78.4% as soon as they deployed it in the edge. In [99], the accuracy drop is estimated to be 1.3%. These percentages are substantial considering the sensitivity of the healthcare domain.
- *The Privacy Problem:* some of the most common sensors used in TinyML are cameras and microphones that may possibly collect sensitive or private information about the user without their knowledge which obviously implies many privacy concerns. The limited resources available in this technology obstruct the possibility to secure and protect the collected data and its storage, which makes it vulnerable to many breaches such as unauthorized manipulation or access by impostors. As indicated by figure 13, the whole edge and computing

TABLE 9. Table of acronyms.

Acronym	Explanation
ML	Machine Learning
AI	Artificial Intelligence
TinyML	Tiny Machine Learning
DL	Deep Learning
WASN	Wireless Acoustic Sensor Network
WSAN	Wireless Sensor and Actuator Network
NLP	Natural Language Processing
GPU	Graphics Processing Unit
CPU	Central Processing Unit
RL	Reinforcement Learning
AV	Autonomous Vehicle
IDE	Integrated Development Environment
ANN	Artificial Neural Network
APW	Alkaline Peptone Wate
HSI	Hyperspectral imaging
CNN	Convolutional Neural Networks
LPWAN	Low-Power Wide-Area Network
IoT	Internet of Things
FLL	Focal Liver Lesion
SA	Smart Agriculture
SF	Smart Farming
FOMO	Faster Objects More Objects
MSE	Mean Squared Error
MCU	Microcontroller Unit
TinyOL	Tiny Online Learning
TEDA	Typicality and Eccentricity Data Analytics
GPS	Global Positioning System
GOps/s/W	Giga operations per second per watt
IoT	Internet of Things
SLR	Systematic Literature Review
ms	Millisecond

process is susceptible to unethical security breaches, which implies unsafety in terms of data collection and data flow. The wide deployment of TinyML may yield to the accumulation of personal data of users with respect to the lack of regulation, standardization and specifications of this technology, and thus exacerbating the privacy issue. Attempts already addressed this issue such as the TinyMLaaS [23], where they build an embedded architecture capable of confining business sensitive data within the IoT device boundaries. However, to fulfill this shield of protection, the device should be constraint by a narrowband connectivity (NB-IoT) meaning that the device should have a very limited possession of data transmissions. Another promising model that considered this important concern is [142], where they established a TinyML model whose data is processed within the device, which [142] called it privacy-centric on-device transfer learning, without any interaction or sharing with the cloud and external servers, Hence solidifying data privacy and its security.

- *Product Trustworthiness and Reliability:* TinyML is a relatively new field that deals with the application of machine learning algorithms on small, low-power devices such as microcontrollers and sensors. The device will definitely gain many use cases in a variety of sectors and domains; nonetheless, the trustworthiness and reliability of such device remains questionable, especially in the healthcare domain where the life and health of patients is at stake, and the robustness of the device becomes a necessity. Additionally, researchers of [143] corroborate the reliability challenge of TinyML, which consists of the variations in precision of these devices, since they vulnerable to variety of extensive factors; in addition to the errors that may occur during the construction of circuits and assemblage of wafers. The hardware layer is also exposed to high energy particles that modifies the output of the algorithm. Finally, aging of the device and the degradation of its components may provoke other variations in battery-life, accuracy, and even data collection by sensors.

VIII. CONCLUSION

Over the last couple of years, the topic of TinyML has gained a tremendous attention from industry and academia, with the drive of unlocking various new possibilities for sustainable development technologies. Artificial intelligence has become nowadays ubiquitous and has demonstrated its capability to bring new approaches and solutions to various research problems. However, training AI algorithms needs extensive computation as well as specialized (costly) hardware, which leads to a higher energy consumption and a significant carbon footprint. To overcome such AI issues, TinyML is the suitable candidate technology, which will make the future of AI tiny and bright. In this paper, we presented the results of a comprehensive literature survey of all TinyML applications and related research efforts. Special emphasis has been placed on building a taxonomy of TinyML techniques that have been used so far to bring new solutions to various domains, such as healthcare, smart farming, environment, and anomaly detection. Our survey has shed light on the new insights of TinyML and how it unlocks new possibilities for sustainable development. For this we discussed how TinyML reduces latency and enables real time applications to be deployed in the source of data. We also brought to the attention of AI researchers how TinyML models can run even when there is no internet connection, and how the processed data do not leave the device, which significantly improves user privacy and thus complies with data protection regulations. Finally, this survey highlighted the remaining challenges and discussed future research directions. We believe that our survey will serve as a guideline for the future research initiatives in TinyML and will motivate further discussions in this promising field.

REFERENCES

[1] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, "General video game AI: A multitrack framework for evaluating agents, games, and content generation algorithms," *IEEE Trans. Games*, vol. 11, no. 3, pp. 195–214, Sep. 2019.

[2] N. L. Bragazzi, H. Dai, G. Damiani, M. Behzadifar, M. Martini, and J. Wu, "How big data and artificial intelligence can help better manage the COVID-19 pandemic," *Int. J. Environ. Res. Public Health*, vol. 17, no. 9, p. 3176, May 2020.

[3] F. E. El Aidos, M. Kassab, N. Benamar, and B. Falah, "A comprehensive survey on blockchain-based solutions to combat COVID-19 pandemic," *Int. J. Comput. Digit. Syst.*, vol. 11, no. 1, pp. 873–892, Feb. 2022.

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

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

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

[7] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, "A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7366–7390, Oct. 2022.

[8] S. El Hamdani and N. Benamar, "A comprehensive study of intelligent transportation system architectures for road congestion avoidance," in *Ubiquitous Networking*. Cham, Switzerland: Springer, 2017, pp. 95–106.

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

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

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

[12] Y. Huang, B. Guo, and Y. Shen, "GPU energy consumption optimization with a global-based neural network method," *IEEE Access*, vol. 7, pp. 64303–64314, 2019.

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

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

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

[16] A. Curry, "The internet of animals," *Nature*, vol. 562, no. 7727, pp. 322–326, 2018.

[17] T. A. Wild, M. Wikelski, S. Tyndel, G. Alarcón-Nieto, B. C. Klump, L. M. Aplin, M. Meboldt, and H. J. Williams, "Internet on animals: Wi-Fi-enabled devices provide a solution for big data transmission in biologging," *Methods Ecology Evol.*, vol. 14, no. 1, pp. 87–102, Jan. 2023.

[18] S. O. Ooko, M. M. Ogore, J. Nsenga, and M. Zennaro, "TinyML in Africa: Opportunities and challenges," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2021, pp. 1–6.

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

[20] V. Tsoukas, E. Boumpa, G. Giannakas, and A. Kakarountas, "A review of machine learning and TinyML in healthcare," in *Proc. 25th Pan-Hellenic Conf. Informat.*, Nov. 2021, pp. 1–7.

[21] S. Bangalore Lakshman and N. U. Eisty, "Software engineering approaches for TinyML based IoT embedded vision: A systematic literature review," 2022, *arXiv:2204.08702*.

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

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

[24] N. Schizas, A. Karras, C. Karras, and S. Sioutas, "TinyML for ultra-low power AI and large scale IoT deployments: A systematic review," *Future Internet*, vol. 14, no. 12, p. 363, Dec. 2022.

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

[26] M. Shafique, T. Theocharides, V. J. Reddy, and B. Murmann, "TinyML: Current progress, research challenges, and future roadmap," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 1303–1306.

[27] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, D. Patterson, D. Pau, J.-S. Seo, J. Sieracki, U. Thakker, M. Verhelst, and P. Yadav, "Benchmarking TinyML systems: Challenges and direction," 2020, *arXiv:2003.04821*.

[28] V. Rajapakse, I. Karunanayake, and N. Ahmed, "Intelligence at the extreme edge: A survey on reformable TinyML," 2022, *arXiv:2204.00827*.

[29] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware: A review," *IEEE Sensors J.*, vol. 22, no. 22, pp. 21362–21390, Nov. 2022.

[30] S. Keele, "Guidelines for performing systematic literature reviews in software engineering," Dept. Comput. Sci., Univ. Durham, Durham, U.K., Tech. Rep. EBSE-2007-01, Ver. 2.3, 2007.

[31] R. Nalwanga, J. Nsenga, G. Rushingabigwi, and I. Gatare, "Design of an embedded machine learning based system for an environmental-friendly crop prediction using a sustainable soil fertility management," in *Proc. IEEE 19th Conf. Res. Develop. (SCOReD)*, Nov. 2021, pp. 251–256.

[32] S. W. Mohammed, N. R. Soora, N. Polala, and S. Saman, "Smart water resource management by analyzing the soil structure and moisture using deep learning," in *IoT With Smart Systems*, J. Choudrie, P. Mahalle, T. Perumal, and A. Joshi, Eds. Singapore: Springer Nature, 2023, pp. 709–719.

[33] P. Ramos-Giraldo, S. C. Reberg-Horton, S. Mirsky, E. Lobaton, A. M. Locke, E. Henriquez, A. Zuniga, and A. Minin, "Low-cost smart camera system for water stress detection in crops," in *Proc. IEEE SENSORS*, Dec. 2020, pp. 1–4.

[34] D. Evans, "The Internet of Things. How the next evolution of the internet is changing everything, whitepaper," Cisco Internet Bus. Solutions Group (IBSG), San Jose, CA, USA, Tech. Rep., 2011.

[35] S. G. H. Soumyalatha, "Study of IoT: Understanding IoT architecture, applications, issues and challenges," in *Proc. 1st Int. Conf. Innov. Comput. NetWork. (ICICN), CSE, RRCE. Int. J. Adv. Netw. Appl.*, vol. 478, 2016, pp. 1–5.

[36] R. T. Tiburski, C. R. Moratelli, S. F. Johann, M. V. Neves, E. D. Matos, L. A. Amaral, and F. Hessel, "Lightweight security architecture based on embedded virtualization and trust mechanisms for IoT edge devices," *IEEE Commun. Mag.*, vol. 57, no. 2, pp. 67–73, Feb. 2019.

[37] A. S. Syed, D. Sierra-Sosa, A. Kumar, and A. Elmaghraby, "IoT in smart cities: A survey of technologies, practices and challenges," *Smart Cities*, vol. 4, no. 2, pp. 429–475, Mar. 2021.

[38] M. Ahmed, R. Mumtaz, S. M. H. Zaidi, M. Hafeez, S. A. R. Zaidi, and M. Ahmad, "Distributed fog computing for Internet of Things (IoT) based ambient data processing and analysis," *Electronics*, vol. 9, no. 11, p. 1756, Oct. 2020.

[39] E. Ahmed, A. Ahmed, I. Yaqoob, J. Shuja, A. Gani, M. Imran, and M. Shoaib, "Bringing computation closer toward the user network: Is edge computing the solution?" *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 138–144, Nov. 2017.

[40] S. Duan, D. Wang, J. Ren, F. Lyu, Y. Zhang, H. Wu, and X. Shen, "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 591–624, 1st Quart., 2023.

[41] L. Greco, G. Percannella, P. Ritrovato, F. Tortorella, and M. Vento, "Trends in IoT based solutions for health care: Moving AI to the edge," *Pattern Recognit. Lett.*, vol. 135, pp. 346–353, Jul. 2020.

[42] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "AI benchmark: Running deep neural networks on Android smartphones," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, Sep. 2018, pp. 288–314.

[43] V. J. Reddi, B. Plancher, S. Kennedy, L. Moroney, P. Warden, L. Suzuki, A. Agarwal, C. Banbury, M. Banzi, and M. Bennett, "Widening access to applied machine learning with TinyML," *Harvard Data Sci. Rev.*, 2022.

[44] *V1.0 Results*. [Online]. Available: https://mlcommons.org/en/inference-tiny-10/

[45] *Dawnbench*. [Online]. Available: https://dawn.cs.stanford.edu/benchmark/CIFAR10/inference.html

[46] *Microcontroller Market Size and Forecast*. Accessed: Feb. 5, 2023. [Online]. Available: https://www.verifiedmarketresearch.com/product/microcontroller-market/

[47] Z. Li, J. Gao, and J. Lai, "HBDCA: A toolchain for high-accuracy BRAM-defined CNN accelerator on FPGA with flexible structure," *IEICE Trans. Inf. Syst.*, vol. 104, no. 10, pp. 1724–1733, 2021.

[48] S. Hosseininoorbin, S. Layeghy, M. Sarhan, R. Jurdak, and M. Portmann, "Exploring edge TPU for network intrusion detection in IoT," 2021, *arXiv:2103.16295*.

[49] *Edge TPU—Run Inference at the Edge &NBSP;|&NBSP; Google Cloud*. [Online]. Available: https://cloud.google.com/edge-tpu/

[50] U. Thakker, J. Beu, D. Gope, C. Zhou, I. Fedorov, G. Dasika, and M. Mattina, "Compressing RNNs for IoT devices by 15–38x using Kronecker products," 2019, *arXiv:1906.02876*.

[51] S. Zhang, Y. Wu, C. Men, and X. Li, "Tiny YOLO optimization oriented bus passenger object detection," *Chin. J. Electron.*, vol. 29, no. 1, pp. 132–138, Jan. 2020.

[52] B. Wu, A. Wan, F. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 446–454.

[53] A. Womg, M. J. Shafiee, F. Li, and B. Chwyl, "Tiny SSD: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection," in *Proc. 15th Conf. Comput. Robot Vis. (CRV)*, May 2018, pp. 95–101.

[54] A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma, "FastGRNN: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31. Red Hook, NY, USA: Curran Associates, 2018, pp. 1–23. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/ab013ca67cf2d50796b0c11d1b8bc95d-Paper.pdf

[55] M. córdova, A. Pinto, H. Pedrini, and R. d. S. Torres, "Pelee-Text++: A tiny neural network for scene text detection," *IEEE Access*, vol. 8, pp. 223172–223188, 2020.

[56] C. Zhang, H. Wang, J. Zeng, L. Ma, and L. Guan, "Tiny-RainNet : A deep convolutional neural network with bi-directional long short-term memory model for short-term rainfall prediction," *Meteorological Appl.*, vol. 27, no. 5, p. e1956, Sep. 2020.

[57] N. P. Ghanathe and S. Wilton, "T-RECX: Tiny-resource efficient convolutional neural networks with early-eXit," 2022, *arXiv:2207.06613*.

[58] K. Xu, Y. Li, H. Zhang, R. Lai, and L. Gu, "EtinyNet: Extremely tiny network for TinyML," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 4, 2022, pp. 4628–4636.

[59] T. M. Khan, A. Robles-Kelly, and S. S. Naqvi, "T-Net: A resource-constrained tiny convolutional neural network for medical image segmentation," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 1799–1808.

[60] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.

[61] Q. Lu, W. Jiang, M. Jiang, J. Hu, S. Dasgupta, and Y. Shi, "FGNAS: FPGA-aware graph neural architecture search," Tech. Rep., 2020.

[62] S. Soro, "TinyML for ubiquitous edge AI," 2021, *arXiv:2102.01255*.

[63] TensorFlow. (2022). *Tensorflow Lite Guide*. [Online]. Available: https://www.tensorflow.org/lite/guide

[64] (2022). *Tensorflow Lite Micro*. [Online]. Available: https://www.tensorflow.org/lite/microcontrollers

[65] Edge Impulse. (2022). *Getting Started*. [Online]. Available: https://docs.edgeimpulse.com/docs/

[66] D. Situnayake. (2021). *How Tensorflow Helps Edge Impulse Make ML Accessible to Embedded Engineers*. [Online]. Available: https://blog.tensorflow.org/2021/06/how-tensorflow-helps-edge-impulse-make-ml-accessible.html

[67] J. Jongboom. (2020). *Introducing EON: Neural Networks in up to 55% Less Ram and 35% Less Rom*. https://www.edgeimpulse.com/blog/introducing-eon

[68] M. K. L. Moreau. (2020). *Announcing Fomo (Faster Objects, More Objects)*. [Online]. Available: https://www.edgeimpulse.com/blog/announcing-fomo-faste

[69] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology," in *Proc. 14th Int. Conf. Telecommun. (ITST)*, Dec. 2015, pp. 55–59.

[70] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial Internet of Things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78238–78259, 2018.

[71] *What is LoRawan*®. [Online]. Available: https://resources.lora-alliance.org/document/what-is-lorawan

[72] R. Sanchez-Iborra, "LPWAN and embedded machine learning as enablers for the next generation of wearable devices," *Sensors*, vol. 21, no. 15, p. 5218, 2021.

[73] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment," *ICT Exp.*, vol. 5, no. 1, pp. 1–7, Mar. 2019.

[74] M. I. Hossain and J. I. Markendahl, "Comparison of LPWAN technologies: Cost structure and scalability," *Wireless Pers. Commun.*, vol. 121, no. 1, pp. 887–903, Nov. 2021.

[75] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT," *ICT Exp.*, vol. 3, no. 1, pp. 14–21, Mar. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959517300061

[76] S. Pizzi, C. Suraci, A. Iera, A. Molinaro, and G. Araniti, "A sidelink-aided approach for secure multicast service delivery: From human-oriented multimedia traffic to machine type communications," *IEEE Trans. Broadcast.*, vol. 67, no. 1, pp. 313–323, Mar. 2021.

[77] P. Andrade, I. Silva, M. Silva, T. Flores, J. Cassiano, and D. G. Costa, "A TinyML soft-sensor approach for low-cost detection and monitoring of vehicular emissions," *Sensors*, vol. 22, no. 10, p. 3838, May 2022.

[78] M. F. Alati, G. Fortino, J. Morales, J. M. Cecilia, and P. Manzoni, "Time series analysis for temperature forecasting using TinyML," in *Proc. IEEE 19th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2022, pp. 691–694.

[79] F. Alongi, N. Ghielmetti, D. Pau, F. Terraneo, and W. Fornaciari, "Tiny neural networks for environmental predictions: An integrated approach with miosix," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Sep. 2020, pp. 350–355.

[80] A. Omambia, B. Maake, and A. Wambua, "Water quality monitoring using IoT & machine learning," in *Proc. IST-Africa Conf. (IST-Africa)*, May 2022, pp. 1–8.

[81] N. N. M. Rizal, G. Hayder, and K. A. Yusof, "Water quality predictive analytics using an artificial neural network with a graphical user interface," *Water*, vol. 14, no. 8, p. 1221, Apr. 2022.

[82] D. Loukatos, K.-A. Lygkoura, C. Maraveas, and K. G. Arvanitis, "Enriching IoT modules with edge AI functionality to detect water misuse events in a decentralized manner," *Sensors*, vol. 22, no. 13, p. 4874, Jun. 2022.

[83] M. M. Ogore, K. Nkurikiyeyezu, and J. Nsenga, "Offline prediction of cholera in rural communal tap waters using edge AI inference," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2021, pp. 1–6.

[84] N. Naeem, T. Boroushaki, and W. Chen, *Efficient Ultra Low Power Underwater Imaging*.

[85] G. Datta, Z. Yin, A. Jacob, A. R. Jaiswal, and P. A. Beerel, "Toward efficient hyperspectral image processing inside camera pixels," Tech. Rep., 2022.

[86] T. Clements, "Earthquake detection with TinyML," in *Proc. AGU Fall Meeting Abstr.*, Dec. 2021, pp. S15A–0229.

[87] D. Tuia, B. Kellenberger, S. Beery, B. R. Costelloe, S. Zuffi, B. Risse, A. Mathis, M. W. Mathis, F. van Langevelde, and T. Burghardt, "Perspectives in machine learning for wildlife conservation," *Nature Commun.*, vol. 13, no. 1, p. 792, 2022.

[88] D. J. Curnick, A. J. Davies, C. Duncan, R. Freeman, D. M. P. Jacoby, H. T. E. Shelley, C. Rossi, O. R. Wearn, M. J. Williamson, and N. Pettorelli, "SmallSats: A new technological frontier in ecology and conservation?" *Remote Sens. Ecology Conservation*, vol. 8, no. 2, pp. 139–150, Apr. 2022.

[89] Hackster IO. (2020). *ElephantEdge—Hackster.Io*. Accessed: Jan. 13, 2023. [Online]. Available: https://www.hackster.io/contests/ElephantEdge

[90] Manivannan. (2020). *Elephant Guardian—TinyMLs to Save the Giant Wild Animal*. Accessed: Jan. 13, 2023. [Online]. Available: https://www.hackster.io/manivannan/elephant-guardian-tinymls-to-save-the-giant-wild-animal-372306

[91] S. Gallacher, D. Wilson, A. Fairbrass, D. Turmukhambetov, M. Firman, S. Kreitmayer, O. M. Aodha, G. Brostow, and K. Jones, "Shazam for bats: Internet of Things for continuous real-time biodiversity monitoring," *IET Smart Cities*, vol. 3, no. 3, pp. 171–183, Sep. 2021.

[92] N. Karges, J. Staab, J. Rauh, M. Wegmann, and H. Taubenböck, "Soundscapes on edge-the real-time machine learning approach for measuring soundscapes on resource-constrained devices," in *Proc. 24th Int. Congr. Acoust.*, 2022, pp. 128–139.

[93] M. S. Diab and E. Rodriguez-Villegas, "Embedded machine learning using microcontrollers in wearable and ambulatory systems for health and care applications: A review," *IEEE Access*, vol. 10, pp. 98450–98474, 2022.

[94] C. Nicolas, B. Naila, and R.-C. Amar, "TinyML smart sensor for energy saving in Internet of Things precision agriculture platform," in *Proc. 13th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2022, pp. 256–259.

[95] K. Ahmed and M. Hassan, "TinyCare: A TinyML-based low-cost continuous blood pressure estimation on the extreme edge," in *Proc. IEEE 10th Int. Conf. Healthcare Informat. (ICHI)*, Jun. 2022, pp. 264–275.

[96] D. Takeuchi, K. Yatabe, Y. Koizumi, Y. Oikawa, and N. Harada, "Real-time speech enhancement using equilibriated RNN," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 851–855.

[97] I. Fedorov, M. Stamenovic, C. Jensen, L.-C. Yang, A. Mandell, Y. Gan, M. Mattina, and P. N. Whatmough, "TinyLSTMS: Efficient neural speech enhancement for hearing aids," Cornell Univ., Ithaca, NY, USA, Tech. Rep., 2020.

[98] F. Sabry, T. Eltaras, W. Labda, K. Alzoubi, and Q. Malluhi, "Machine learning for healthcare wearable devices: The big picture," *J. Healthcare Eng.*, vol. 2022, pp. 1–25, Apr. 2022.

[99] H. Gokul, P. Suresh, B. H. Vignesh, R. P. Kumaar, and V. Vijayaraghavan, "Gait recovery system for Parkinson's disease using machine learning on embedded platforms," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, Aug. 2020, pp. 1–8.

[100] A. Faraone and R. Delgado-Gonzalo, "Convolutional-recurrent neural networks on low-power wearable platforms for cardiac arrhythmia detection," in *Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Aug. 2020, pp. 153–157.

[101] R. Zanetti, A. Aminifar, and D. Atienza, "Robust epileptic seizure detection on wearable systems with reduced false-alarm rate," in *Proc. 42nd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2020, pp. 4248–4251.

[102] C. D. Caleanu, C. L. Sîrbu, and G. Simion, "Deep neural architectures for contrast enhanced ultrasound (CEUS) focal liver lesions automated diagnosis," in *Proc. Int. Symp. Electron. Telecommun. (ISETC)*, 2021, pp. 1–9.

[103] T. M. Ingolfsson, A. Cossettini, X. Wang, E. Tabanelli, G. Tagliavini, P. Ryvlin, L. Benini, and S. Benatti, "Towards long-term non-invasive monitoring for epilepsy via wearable EEG devices," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2021, pp. 1–4.

[104] D. Ayata, Y. Yaslan, and M. E. Kamasak, "Emotion recognition from multimodal physiological signals for emotion aware healthcare systems," *J. Med. Biol. Eng.*, vol. 40, pp. 149–157, Jan. 2020.

[105] R. Laureanti, M. Bilucaglia, M. Zito, R. Circi, A. Fici, F. Rivetti, R. Valesi, C. Oldrini, L. T. Mainardi, and V. Russo, "Emotion assessment using machine learning and low-cost wearable devices," in *Proc. 42nd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2020, pp. 576–579.

[106] M. S. Fernandes, W. Cordeiro, and M. Recamonde-Mendoza, "Detecting aedes aegypti mosquitoes through audio classification with convolutional neural networks," *Comput. Biol. Med.*, vol. 129, Feb. 2021, Art. no. 104152.

[107] K. Trivedi and H. Shroff, "Identification of deadliest mosquitoes using wing beats sound classification on tiny embedded system using machine learning and edge impulse platform," in *Proc. ITU Kaleidoscope, Connecting Phys. Virtual Worlds (ITUK)*, 2021, pp. 1–6.

[108] N. Abdennadher, D. Pau, and A. Bruna, "Fixed complexity tiny reservoir heterogeneous network for on-line ECG learning of anomalies," in *Proc. IEEE 10th Global Conf. Consum. Electron. (GCCE)*, Oct. 2021, pp. 233–237.

[109] United Nation. (2017). *World Population Projected to Reach 9.8 Billion in 2050, and 11.2 Billion in 2100*. Accessed: Dec. 20, 2022. [Online]. Available: https://www.un.org/en/desa/world-population-projected-reach-98-billion-2050-and-112-billion-2100

[110] A. Mitra, S. L. T. Vangipuram, A. K. Bapatla, V. K. V. V. Bathalapalli, S. P. Mohanty, E. Kougianos, and C. Ray, "Everything you wanted to know about smart agriculture," 2022, *arXiv:2201.04754*.

[111] S. Condran, M. Bewong, M. Z. Islam, L. Maphosa, and L. Zheng, "Machine learning in precision agriculture: A survey on trends, applications and evaluations over two decades," *IEEE Access*, vol. 10, pp. 73786–73803, 2022.

[112] Y. Kalyani and R. Collier, "A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture," *Sensors*, vol. 21, no. 17, p. 5922, Sep. 2021.

[113] G. Singh, A. Singh, and G. Kaur, "Role of artificial intelligence and the Internet of Things in agriculture," in *Artificial Intelligence to Solve Pervasive Internet of Things Issues*. Amsterdam, The Netherlands: Elsevier, 2021, pp. 317–330.

[114] V. K. Quy, N. V. Hau, D. V. Anh, N. M. Quy, N. T. Ban, S. Lanza, G. Randazzo, and A. Muzirafuti, "IoT-enabled smart agriculture: Architecture, applications, and challenges," *Appl. Sci.*, vol. 12, no. 7, p. 3396, Mar. 2022.

[115] D. Sheth, B. Sudharsan, J. G. Breslin, and M. I. Ali, "Embedded ML pipeline for precision agriculture," in *Proc. 21st ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, May 2022, pp. 527–528.

[116] L. Falaschetti, L. Manoni, R. C. F. Rivera, D. Pau, G. Romanazzi, O. Silvestroni, V. Tomaselli, and C. Turchetti, "A low-cost, low-power and real-time image detector for grape leaf esca disease based on a compressed CNN," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 3, pp. 468–481, Sep. 2021.

[117] M. Alessandrini, R. Calero Fuentes Rivera, L. Falaschetti, D. Pau, V. Tomaselli, and C. Turchetti, "A grapevine leaves dataset for early detection and classification of esca disease in vineyards through machine learning," *Data Brief*, vol. 35, Apr. 2021, Art. no. 106809.

[118] F. de Vita, G. Nocera, D. Bruneo, V. Tomaselli, D. Giacalone, and S. K. Das, "Quantitative analysis of deep leaf: A plant disease detector on the smart edge," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Sep. 2020, pp. 49–56.

[119] G. Esgario. (2019). *Coffee Dataset*. Accessed: Dec. 20, 2022. [Online]. Available: https://drive.google.com/file/d/15YHebAGrx1Vhv8-naave-R5o3Uo70jsm

[120] V. Gutti and R. Karthi, "Real time classification of fruits and vegetables deployed on low power embedded devices using tiny ML," in *Proc. Int. Conf. Image Process. Capsule Netw.* Cham, Switzerland: Springer, 2022, pp. 347–359.

[121] M. Oltean. (2017). *Fruits 360 Dataset: A Dataset of Images Containing Fruits and Vegetables*. Accessed: Dec. 20, 2022. [Online]. Available: https://www.kaggle.com/datasets/moltean/fruits

[122] G. J. Miguel. (2021). *Crop Recommendation Using Machine Learning*. Accessed: Dec. 20, 2022. [Online]. Available: https://github.com/gabbygab1233/Crop-Recommender

[123] C. Bruno, A. Licciardello, G. A. M. Nastasi, F. Passaniti, C. Brigante, F. Sudano, A. Faulisi, and E. Alessi, "Embedded artificial intelligence approach for gas recognition in smart agriculture applications using low cost MOX gas sensors," in *Proc. Smart Syst. Integr. (SSI)*, Apr. 2021, pp. 1–5.

[124] I. Ihoume, R. Tadili, N. Arbaoui, M. Benchrifa, A. Idrissi, and M. Daoudi, "Developing a multi-label TinyML machine learning model for an active and optimized greenhouse microclimate control from multivariate sensed data," *Artif. Intell. Agricult.*, vol. 6, pp. 129–137, Jan. 2022.

[125] R. Sanchez-Iborra, A. Zoubir, A. Hamdouchi, A. Idri, and A. Skarmeta, "Intelligent and efficient IoT through the cooperation of TinyML and edge computing," *Informatica*, vol. 34, no. 1, pp. 147–168, 2023.

[126] D. M. Matilla, A. L. Murciego, D. M. Jiménez-Bravo, A. S. Mendes, and V. R. Leithardt, "Low-cost edge computing devices and novel user interfaces for monitoring pivot irrigation systems based on Internet of Things and LoRaWAN technologies," *Biosystems Eng.*, vol. 223, pp. 14–29, Nov. 2022.

[127] P. Du, T. Polonelli, M. Magno, and Z. Cheng, "Towards lightweight deep neural network for smart agriculture on embedded systems," in *Proc. IEEE Sensors Appl. Symp. (SAS)*, Aug. 2022, pp. 1–6.

[128] *Plant Village*. Accessed: Feb. 5, 2023. [Online]. Available: https://plantvillage.psu.edu/

[129] J. Manokaran and G. Vairavel, "Smart anomaly detection using data-driven techniques in IoT edge: A survey," in *Proc. 3rd Int. Conf. Commun., Comput. Electron. Syst.* Cham, Switzerland: Springer, 2022, pp. 685–702.

[130] A. Chatterjee and B. S. Ahmed, "IoT anomaly detection methods and applications: A survey," *Internet Things*, vol. 19, Aug. 2022, Art. no. 100568.

[131] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine learning for anomaly detection: A systematic review," *IEEE Access*, vol. 9, pp. 78658–78700, 2021.

[132] A. Xenakis, A. Karageorgos, E. Lallas, A. E. Chis, and H. González-Vélez, "Towards distributed IoT/Cloud based fault detection and maintenance in industrial automation," *Proc. Comput. Sci.*, vol. 151, pp. 683–690, Jan. 2019.

[133] A. Mostafavi and A. Sadighi, "A novel online machine learning approach for real-time condition monitoring of rotating machines," in *Proc. 9th RSI Int. Conf. Robot. Mechatronics (ICRoM)*, Nov. 2021, pp. 267–273.

[134] M. Antonini, M. Pincheira, M. Vecchio, and F. Antonelli, "A TinyML approach to non-repudiable anomaly detection in extreme industrial environments," in *Proc. IEEE Int. Workshop Metrology Ind. 4.0 IoT (MetroInd4.0IoT)*, Jun. 2022, pp. 397–402.

[135] M. Lord and A. Kaplan, "Mechanical anomaly detection on an embedded microcontroller," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2021, pp. 562–568.

[136] V. M. Oliveira and A. H. Moreira, "Edge AI system using a thermal camera for industrial anomaly detection," in *International Summit Smart City 360°*. Cham, Switzerland: Springer, 2022, pp. 172–187.

[137] D. Pau, A. Khiari, and D. Denaro, "Online learning on tiny microcontrollers for anomaly detection in water distribution systems," in *Proc. IEEE 11th Int. Conf. Consum. Electron. (ICCE-Berlin)*, Nov. 2021, pp. 1–6.

[138] M. Cardoni, D. P. Pau, L. Falaschetti, C. Turchetti, and M. Lattuada, "Online learning of oil leak anomalies in wind turbines with block-based binary reservoir," *Electronics*, vol. 10, no. 22, p. 2836, Nov. 2021.

[139] P. Andrade, I. Silva, G. Signoretti, M. Silva, J. Dias, L. Marques, and D. G. Costa, "An unsupervised TinyML approach applied for pavement anomalies detection under the Internet of Intelligent vehicles," in *Proc. IEEE Int. Workshop Metrology for Ind. 4.0 IoT (MetroInd4.0IoT)*, Jun. 2021, pp. 642–647.

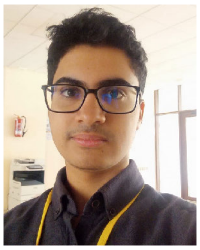
[140] H. Ren, D. Anicic, and T. A. Runkler, "TinyOL: TinyML with online-learning on microcontrollers," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.

[141] *State of the Tinyautoml Market 2022*, TinyML Found., Altos, CA, USA, 2022.

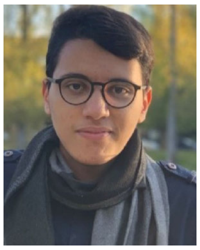
[142] K. Kopparapu, E. Lin, J. G. Breslin, and B. Sudharsan, "TinyFedTL: Federated transfer learning on ubiquitous tiny IoT devices," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Affiliated Events (PerCom Workshops)*, Mar. 2022, pp. 79–81.

[143] M. Shafique, A. Marchisio, R. V. W. Putra, and M. A. Hanif, "Towards energy-efficient and secure edge AI: A cross-layer framework," in *Proc. 40th IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*. Ithaca, NY, USA: Cornell University, 2021, pp. 1–9.

YOUSSEF ABADADE received the M.Sc. degree in information systems engineering from the Faculty of Sciences Semlalia, Cadi Ayyad University, Marrakesh, Morocco, in 2015. He is currently pursuing the Ph.D. degree in TinyML with the System Engineering Laboratory, National School of Applied Sciences, Ibn Tofail University, Kenitra, Morocco. He is also the Technical Lead of Java with KARAVEL and the Leader of Tourism Market in Paris, France.



ANAS TEMOUDEN is currently pursuing the degree in artificial intelligence and robotization with Al Akhawayn University, Ifrane, Morocco. He has been interested in research since his first year of university and he has published his first paper at the age of 18. His research interests include artificial intelligence, embedded systems in general, and tiny machine learning in particular.



HATIM BAMOUMEN (Student Member, IEEE) is currently a highly motivated student with Al Akhawayn University, Ifrane. He has a passion for cutting-edge technologies and has demonstrated a strong commitment to research and innovation. As a member of several research projects, he has contributed to the advancement of the field of TinyML and its applications in various sectors, typically in the environment. As he elucidated the advantages of applying TinyML for sustainable environmental development. He has also served as a member for the IEEE Student Branch and has earned accredited certifications as a peer tutor in several STEM subjects, including applied mechanics and electrical engineering. His research interests include beyond engineering and into the realm of economics, where he has conducted numerous research analysis projects. One notable project involved the application of machine learning models to predict African economies, highlighting his multidisciplinary approach to problem-solving. With his diverse skill set and passion for innovation, he is poised to make a significant impact in the field of engineering and beyond.

NABIL BENAMAR received the master's and Ph.D. degrees from Moulay Ismail University, Meknes, Morocco, in 2001 and 2004, respectively. He is currently a Professor in computer science with the School of Technology, Moulay Ismail University, and an Adjunct Faculty Member of Computer Science with Al Akhawayn University, Ifrane, Morocco. He is the author of several journal articles and IETF standard documents. His research interests include future generation networks, autonomous driving, the IoT, and TinyML. He is a member of the Tiny Machine Learning Open Education Initiative (TinyMLedu). He is also serving as an Associate Editor for the IEEE Access journal and the *Journal of King Saud University—Computer and Information Sciences* (IF 8.8). He is a TPC Member of highly ranked IEEE Flagship Conferences (GLOBECOM, ICC, PIMRC, and WCNC). He served as the Chair for IEEE MenaComm'20 Conference and a member of the Organizing Committee for IEEE WCNC'2019 and IWCMC'23. He is an expert in internet governance and he was an ISOC Ambassador to IGF (2012 and 2013), Google panelist in the first Arab-IGF, an ISOC Fellow to IETF'89&92&95&99&103, and an ICANN'50&54 Fellow. Among his international commitments, he is currently serving as the Chair for the Task Force for Arabic Script IDNs, a team of people working on the implementation of the Arabic script in the DNS root zone. He is also chairing the UASG measurement WG promoting the universal acceptance of all valid domain names and e-mail addresses.

YOUSRA CHTOUKI received the bachelor's degree in computer science from UCO, USA, in 2001, the master's degree in database management from OCU, USA, in 2005, and the Ph.D. degree (Hons.) in web services applications in e-Learning from Ecole Mohammadia, Rabat, in 2017. She has been a full-time Professor and a Researcher with Al Akhawayn University (AUI), since 2008. She has coauthored two books and contributed to one book on technology in education related topics. She has published multiple journals and conference papers in well ranked journals. She is currently the Founder of the EFP Educational Farm Project with AUI, where she is also an Advisor of the first sustainability club during which time the club earned the PRME winner 2022 Award for promoting sustainability in education. She created multiple courses with the Computer Science Department implementing different pedagogies related to computer science education. She is also a Main Contributor to the launch of the undergraduate research program with AUI through which undergraduate students published papers specific to data analysis, machine learning, and TinyML. She has been an active member of the EduSummIT UNESCO sponsored research group, since 2013; a member of the AI Committee with AUI; and a member of TinyML Foundation. She is also working on the implementation of the first living laboratory with AUI, a multidisciplinary research space to learn and innovate mainly aimed to make use of technologies, such as TinyML to solve environment and agriculture related topics. She has developed multiple theories related to teaching computer science and programming courses such as ''Content: Less is More'' ''Pace Management in Introductory Programming.''

ABDELHAKIM SENHAJI HAFID received the M.S. and Ph.D. degrees in computer science. He spent several years as a Senior Research Scientist with Bell Communications Research (Bellcore), Piscataway, NJ, USA, working in the context of major research projects on the management of next-generation networks. He was also an Assistant Professor with Western University (WU), London, ON, Canada, the Research Director of the Advance Communication Engineering Center (venture established by WU, Bell Canada, and Bay Networks), London, a Researcher with CRIM, Montreal, QC, Canada, a Visiting Scientist with GMD-Fokus, Berlin, Germany, and a Visiting Professor with the University of Évry, Évry-Courcouronnes, France. He is currently a Full Professor with the University of Montreal, Montreal. He is also the Founding Director of the Network Research Laboratory, Montreal, QC, Canada, and the Montreal Blockchain Laboratory, Montreal. He is also a Research Fellow with CIRRELT, Montreal. He co-founded Tipot Technologies Inc., Ottawa, ON, Canada (research and development platform for IoT). He consulted for a number of telecommunication companies and startups in North America. He has extensive academic and industrial research experience in the area of the management and design of next-generation networks. He has supervised to graduation over 50 graduate and postgraduate students. He has authored or coauthored over 250 journals and conference papers. He also holds three U.S. patents. His current research interests include the IoT, fog/edge computing, blockchain, and intelligent transport systems. He also gave talks/keynotes at a number of international conferences.