electronics

Review

Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices

Abdussalam Elhanashi [1,*], Pierpaolo Dini [1], Sergio Saponara [1] and Qinghe Zheng [2]

[1] Department of Information Engineering, University of Pisa, 56126 Pisa, Italy; pierpaolo.dini@unipi.it (P.D.); sergio.saponara@unipi.it (S.S.)
[2] School of Intelligent Engineering, Shandong Management University, Jinan 250357, China; zqh@sdmu.edu.cn
[*] Correspondence: abdussalam.elhanashi@ing.unipi.it

Abstract: Artificial Intelligence (AI) and Machine Learning (ML) have experienced rapid growth in both industry and academia. However, the current ML and AI models demand significant computing and processing power to achieve desired accuracy and results, often restricting their use to high-capability devices. With advancements in embedded system technology and the substantial development in the Internet of Things (IoT) industry, there is a growing desire to integrate ML techniques into resource-constrained embedded systems for ubiquitous intelligence. This aspiration has led to the emergence of TinyML, a specialized approach that enables the deployment of ML models on resource-constrained, power-efficient, and low-cost devices. Despite its potential, the implementation of ML on such devices presents challenges, including optimization, processing capacity, reliability, and maintenance. This article delves into the TinyML model, exploring its background, the tools that support it, and its applications in advanced technologies. By understanding these aspects, we can better appreciate how TinyML is transforming the landscape of AI and ML in embedded and IoT systems.

Keywords: TinyML; embedded systems; internet of things (IoT); Machine Learning optimization; resource-constrained devices

check for updates

Citation: Elhanashi, A.; Dini, P.; Saponara, S.; Zheng, Q. Advancements in TinyML: Applications, Limitations, and Impact on IoT Devices. *Electronics* **2024**, *13*, 3562. https://doi.org/10.3390/electronics13173562

Academic Editors: Muhammad Shahid Anwar, Maria Torres Vega and Muhammad Salman Pathan

Received: 13 August 2024
Revised: 31 August 2024
Accepted: 4 September 2024
Published: 8 September 2024

1. Introduction

The Internet of Things (IoT) aims to leverage edge computing, a paradigm that enables seamless and real-time processing of data from millions of interconnected sensors and devices. Edge computing refers to a range of devices and networks near the user, facilitating local data processing to enhance efficiency and reduce latency [1]. One significant advantage of IoT devices is their requirement for low computing and processing power, as they are deployed at the network edge, leading to a low memory footprint [2]. IoT devices collect sensory data and transmit them either to a nearby location or cloud platforms for further processing. Edge computing stores and processes these data, providing the necessary infrastructure to support distributed computing [3].

The implementation of edge computing in IoT devices offers several benefits: enhanced security, privacy, and reliability for end-users, reduced latency, and increased availability and throughput response for applications and services. By processing data closer to the source, edge computing mitigates the risks associated with transmitting sensitive information across long distances, thereby reducing potential vulnerabilities. Additionally, it supports real-time analytics, enabling faster decision making and improving the overall responsiveness of IoT systems. This is particularly crucial for time-sensitive applications, such as autonomous vehicles, healthcare monitoring, and industrial automation, where delays in data processing could have significant consequences. Furthermore, edge computing alleviates the burden on centralized cloud systems, optimizing bandwidth usage and ensuring that critical functions remain operational even in the face of network disruptions. Additionally, edge devices can collaborate with sensors and cloud platforms to process data at the network edge rather than solely relying on cloud processing. This collaboration results in effective data management, persistence, delivery, and content caching. Edge computing significantly improves network services, particularly in applications involving human-to-machine interaction and modern healthcare [4,5].

Recent research has highlighted the potential of implementing Machine Learning (ML) techniques in various IoT use cases. However, traditional ML models often demand significant computing power, processing capabilities, and high memory capacity, limiting their implementation in IoT devices and applications [6–8]. Current edge computing technology also faces challenges such as limited transmission capacity and power efficiency, leading to heterogeneous systems. This necessitates a harmonious and holistic infrastructure for updating, training, and deploying ML models [9]. Figure 1 presents a detailed taxonomy of the main applications of TinyML, showcasing the diverse areas where TinyML is making substantial impacts. This includes healthcare, environmental monitoring, industrial automation, and consumer electronics. The taxonomy emphasizes specific use cases and their benefits, offering a clear perspective on how TinyML is transforming various industries through its capability to execute complex Machine Learning tasks on ultra-low-power devices.

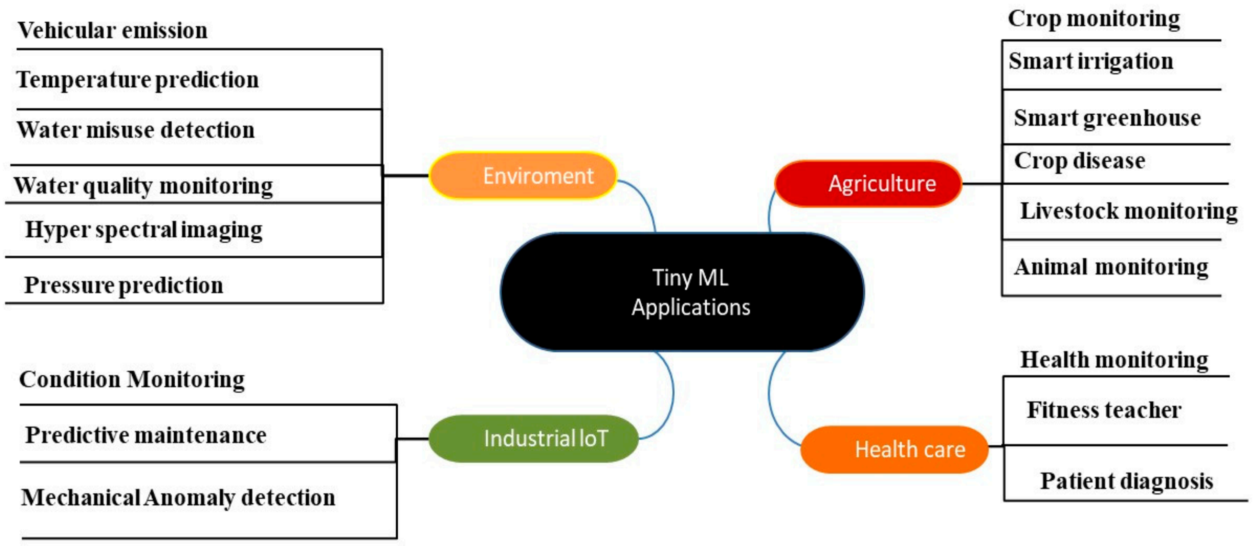

Figure 1. Comprehensive taxonomy categorizing the primary applications of TinyML. It encompasses various domains where TinyML is making significant impacts, highlighting specific use cases and their respective advantages. From healthcare and environmental monitoring to industrial automation and consumer electronics, this taxonomy provides a clear overview of how TinyML is revolutionizing different sectors with its ability to perform complex Machine Learning tasks on ultra-low-power devices.

Furthermore, the architecture designed for embedded devices poses another challenge due to varying hardware and software requirements, making it difficult to build a standard ML architecture for IoT networks [10]. Currently, data generated by different devices are often sent to cloud platforms for processing due to the computationally intensive nature of network implementations. ML models typically rely on deep learning application-specific integrated circuits (ASICs) and graphic processing units (GPUs) for data processing, which require substantial power and memory. Therefore, deploying full-fledged ML models on IoT devices is not feasible due to their limited computing power and storage capacity [11].

The demand for miniaturizing low-power embedded devices and optimizing ML models for better power and memory efficiency has paved the way for TinyML. TinyML aims to implement ML models and practices on edge IoT devices and frameworks. It enables signal processing on IoT devices and provides embedded intelligence, eliminating the need to transfer data to cloud platforms for processing. The successful implementation of TinyML on IoT devices can enhance privacy and efficiency while reducing operating costs. Moreover, TinyML is particularly appealing because it can provide on-premise analytics even in cases of inadequate connectivity [12].

Contributions of This Work

This research review on "Advancements in TinyML" provides a comprehensive overview of the current state and future potential of TinyML. It makes several significant contributions to the field, which are as follows:

- This review highlights an intuitive understanding of TinyML, providing detailed insights into its fundamental principles and applications, making it a valuable resource for readers seeking to grasp both the theoretical and practical aspects of this emerging technology.
- It explores the wide range of applications of TinyML across different fields, including healthcare, agriculture, industrial automation, and environmental monitoring, demonstrating the transformative potential of TinyML in these areas.
- This review provides an in-depth exploration of the key enablers driving the advancement of TinyML technology, while also showcasing a diverse range of use cases and applications across various industries, highlighting its transformative impact on resource-constrained environments.
- This paper explores the current and future challenges in TinyML research, offers practical solutions, and provides a forward-looking perspective on advancing research in this rapidly evolving field, emphasizing the importance of innovation and collaboration to overcome these hurdles.

The rest of this paper is organized as follows. Section 2 presents an overview of TinyML Section 3 explores Methodological Insights and Analysis. Conclusions and future directions are drawn in Section 4.

2. Overview of TinyML

Tiny Machine Learning represents a groundbreaking approach to embedding Machine Learning algorithms into resource-constrained devices, enabling intelligent decision making at the edge of the network. This paradigm shift leverages advancements in hardware, software, and algorithms to process data locally on microcontrollers [13], drastically reducing latency, power consumption [14], and dependency on cloud computing. The advent of TinyML has opened up a plethora of applications across various domains, such as healthcare, agriculture, manufacturing, and environmental monitoring, by bringing AI capabilities to the smallest of devices. One of the primary drivers of TinyML is the continuous improvement in microcontroller units (MCUs), which are becoming increasingly powerful yet energy efficient. These MCUs, often equipped with ARM Cortex-M processors, are now capable of running complex neural network models within a constrained power budget. This makes it feasible to deploy Machine Learning models in scenarios where energy efficiency is paramount, such as battery-operated sensors in remote locations or wearable devices. The software landscape for TinyML is equally vital, with frameworks like TensorFlow Lite [15] for Microcontrollers, uTensor, and Edge Impulse providing the tools necessary to optimize and deploy models on these low-power devices. TensorFlow Lite for Microcontrollers, for instance, enables the conversion of trained models into a format that can run on MCUs with as little as 16 KB of RAM, demonstrating the remarkable strides made in reducing the computational footprint of Machine Learning algorithms. The optimization process often involves techniques such as model quantization, which reduces the precision of the model's weights and activations from a 32-bit floating point to 8-bit integers without significant loss in accuracy. This not only decreases the model size but also enhances the inference speed and reduces power consumption. Pruning, another technique, removes redundant parameters from the model, further shrinking its size and computational requirements. These optimizations are crucial for enabling real-time inference on devices with limited processing power and memory. The applications of TinyML are diverse and transformative. In healthcare, for example, wearable devices equipped with TinyML models can continuously monitor vital signs and detect anomalies, providing early warnings for conditions like arrhythmias or sleep apnea. In agriculture, TinyML-powered sensors can monitor soil moisture, temperature, and crop health, enabling precision farming practices that enhance yield and resource efficiency. In industrial settings, predictive maintenance systems equipped with TinyML can monitor machinery for signs of wear and tear, preventing costly breakdowns and downtime. Environmental monitoring is another area where TinyML is making a significant impact. TinyML sensors can be deployed in large numbers across forests, oceans, and urban areas to monitor air quality, detect forest fires, or track wildlife movements. These sensors operate autonomously for extended periods, thanks to their low power consumption, and provide valuable data for environmental conservation and management. The ability to process data at the edge also enhances privacy and security, as sensitive information does not need to be transmitted to the cloud. This is particularly important in applications like smart homes and healthcare, where data privacy is a major concern. By processing data locally, TinyML reduces the risk of data breaches and ensures that personal information remains secure. The development of TinyML is supported by a vibrant ecosystem of hardware manufacturers, software developers, and research institutions. Companies like ARM, NVIDIA, and Qualcomm are continuously pushing the boundaries of what is possible with edge AI, developing new hardware and software solutions that enhance the capabilities of TinyML. Academic institutions and research labs are also contributing to the field by developing novel algorithms and optimization techniques that further improve the efficiency and performance of TinyML models. Despite the significant progress made, TinyML faces several challenges that need to be addressed to fully realize its potential. One of the main challenges is the limited computational and memory resources available on microcontrollers, which constrain the complexity and accuracy of the models that can be deployed. While techniques like quantization and pruning help mitigate these limitations, there is an ongoing need for more efficient algorithms and model architectures that can deliver high performance within these constraints. Another challenge is the lack of standardized tools and frameworks for developing and deploying TinyML models. While frameworks like TensorFlow Lite for Microcontrollers provide a good starting point, there is a need for more comprehensive tools that can streamline the entire development process, from model training and optimization to deployment and monitoring. Additionally, the integration of TinyML into existing systems and workflows requires significant expertise in both Machine Learning and embedded systems, highlighting the need for better educational resources and training programs. The future of TinyML holds exciting possibilities, with the potential to revolutionize industries and improve quality of life in numerous ways. As hardware continues to advance and new algorithms and optimization techniques are developed, the capabilities of TinyML will continue to expand, enabling even more sophisticated applications. The convergence of TinyML with other emerging technologies, such as the Internet of Things (IoT) and 5G, will further enhance its impact, creating a connected world where intelligent decision making is embedded in every device Figure 2 presents Tiny Machine Learning (TinyML) is focused on revolutionizing application development by embedding Machine Learning models directly into small, resource-constrained devices. This integration allows for advanced data processing and decision-making capabilities on devices with limited computational power and storage, enabling a range of new, innovative applications across various industries. By leveraging TinyML, developers can create more intelligent, responsive, and autonomous systems, enhancing functionalities and user experiences in everyday technology.

TinyML represents a significant leap forward in the field of Machine Learning, enabling intelligent processing at the edge of the network on resource-constrained devices. Through advancements in hardware, software, and algorithms, TinyML is unlocking new applications across various domains, from healthcare and agriculture to industrial automation and environmental monitoring. Despite the challenges, the continued development and adoption of TinyML hold the promise of a smarter, more efficient, and more connected world.

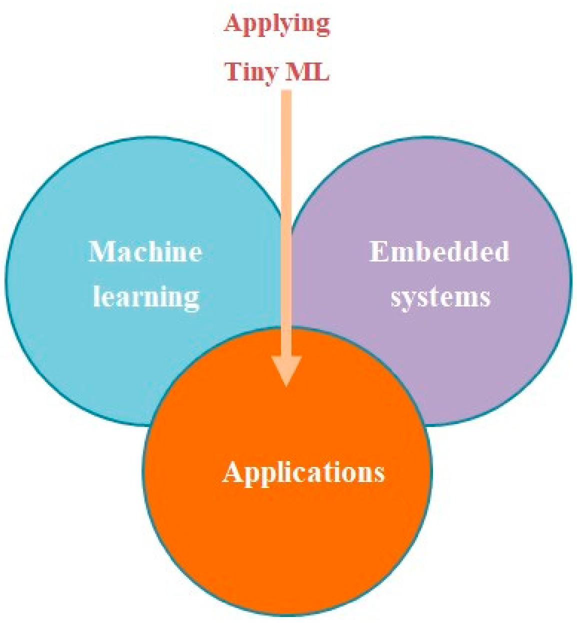

Figure 2. Tiny Machine Learning (TinyML) aims to create new applications by integrating Machine Learning models into embedded systems.

Tiny Machine Learning (TinyML) represents a significant advancement in the integration of Machine Learning with Internet of Things (IoT) devices. This technology enables the deployment of Machine Learning models on ultra-low-power devices, allowing for real-time data processing and decision making without reliance on cloud computing. The following sections provide an overview of the state-of-the-art in TinyML, its applications, challenges, and future directions. TinyML, which involves deploying Machine Learning algorithms on resource-constrained devices, has become a transformative technology in the Internet of Things (IoT) domain. This overview highlights recent advancements and current trends in TinyML for IoT applications. Modern microcontrollers, such as those from the ARM Cortex-M series, and specialized processors like Google's Edge TPU and Intel's Movidius Myriad X, are increasingly used to run TinyML models efficiently, handling computational needs while consuming minimal power [16,17]. Innovations in hardware accelerators, including FPGAs and custom ASICs, enhance the performance and efficiency of ML operations on edge devices by optimizing tasks like matrix multiplications and convolution operations [18,19]. Techniques for model compression, such as quantization, pruning, and knowledge distillation, are crucial for reducing the size of ML models without significantly compromising performance, with quantization converting model weights to lower-bit representations to reduce memory usage [11,20]. Efficient algorithms designed for resource-constrained environments include models like MobileNet and EfficientNet, which minimize computational and memory requirements [21,22]. Software frameworks and tools like TensorFlow Lite for Microcontrollers (TFLite Micro) which has been made by Google that enables deployment on embedded devices with limited resources, providing a lightweight runtime and tools for model conversion and optimization [23], while platforms like Edge Impulse simplify the development and deployment of TinyML models on edge devices [24]. In application areas, TinyML facilitates real-time data processing on smart sensors and actuators for tasks such as anomaly detection, predictive maintenance, and environmental monitoring [10,25], as well as continuous health monitoring and activity recognition in wearables, improving privacy by reducing the need for constant cloud communication [26,27]. Challenges and future directions include optimizing power consumption to extend battery life in IoT devices [28], addressing security and privacy concerns through techniques like local data encryption and secure model updates [29], and exploring solutions for managing and updating large numbers of devices as IoT deployments scale [30]. Figure 3 presents a detailed framework that depicts the cohesive integration of Internet of Things (IoT) applications with cloud computing, edge computing, and TinyML technologies. It highlights the synergistic interplay among these components, illustrating how data are processed and managed across various layers, from edge devices to centralized cloud infrastructures. The framework underscores TinyML's critical role in delivering on-device intelligence and facilitating real-time analytics, which significantly boosts the efficiency and responsiveness of IoT systems. By demonstrating the interplay between local processing at the edge and broader data management in the cloud, this framework showcases how these technologies collectively enhance the performance and capabilities of IoT solutions.

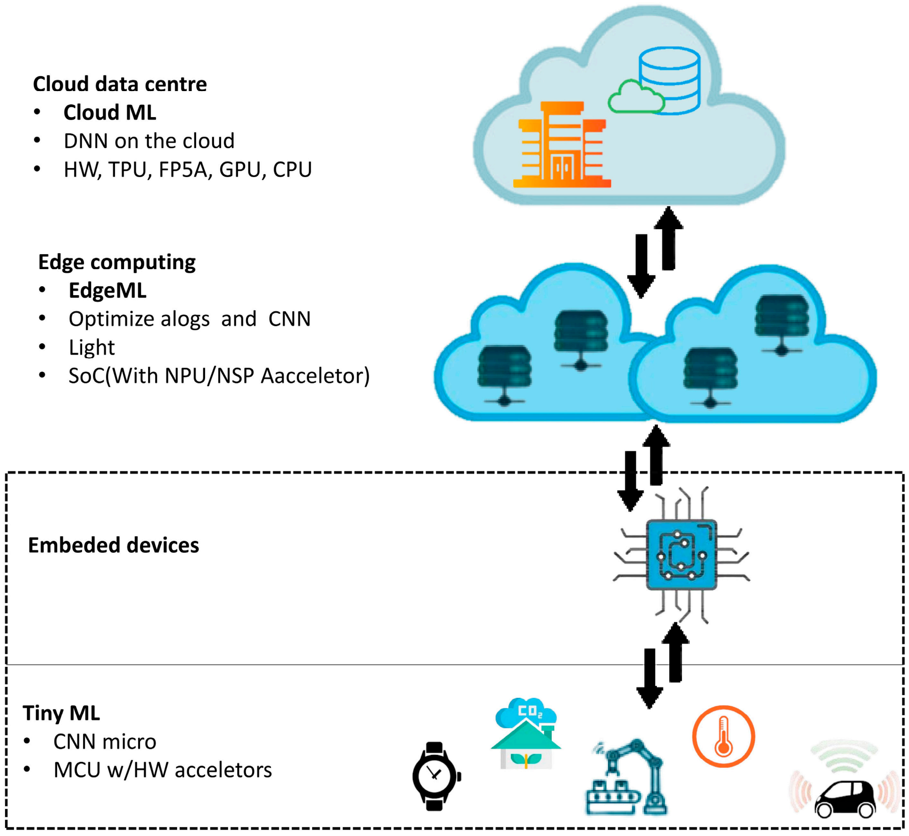

Figure 3. A comprehensive framework illustrating the integration of IoT applications with cloud computing, edge computing, and TinyML. This framework highlights the synergistic interaction between these technologies, showcasing how data are processed and managed at different levels from edge devices to centralized cloud systems. It emphasizes the role of TinyML in enabling on-device intelligence and real-time analytics, enhancing the overall efficiency and responsiveness of IoT solutions.

Deep Learning is a core component of TinyML, enabling complex tasks like image recognition and natural language processing on resource-constrained devices. However, their computational demands require specialized hardware (HW) to run efficiently on these small platforms. Tensor Processing Units (TPUs) are one such specialized hardware designed to accelerate DNN operations, particularly for inferencing tasks, by optimizing for the low power and latency requirements of TinyML. FP5A, a flexible processing unit, is another example that supports various bit-widths to enhance efficiency in running DNNs on embedded systems. Convolutional Neural Networks (CNNs), a type of DNN particularly effective for image processing tasks, have been adapted into microarchitectures like CNN Micro, which is optimized to fit within the tight power and memory constraints typical of TinyML applications. These components work in tandem to ensure that sophisticated Machine Learning models can be deployed effectively in embedded environments, driving innovation in edge AI.

The integration of cloud computing, edge computing, and TinyML forms a powerful framework for IoT applications. cloud computing provides scalable storage and processing power, enabling complex analytics and data management. Edge computing brings processing closer to the data source, reducing latency and enhancing real-time decision making. TinyML, with its efficient Machine Learning models, facilitates low-power, on-device analytics, making IoT devices smarter and more responsive. This synergistic framework enhances data processing efficiency, reduces bandwidth requirements, and supports advanced applications like predictive maintenance, smart homes, and healthcare monitoring, creating a robust and intelligent IoT ecosystem.

3. Methodological Insights and Analysis

We conducted a thorough search of online databases, including Google Scholar, IEEE Xplore, and PubMed, using specific keywords related to TinyML and its applications in IoT. This initial search aimed to gather a broad range of relevant studies and reports, ensuring a comprehensive understanding of the current landscape. Our search strategy was meticulously designed to capture the most pertinent literature, encompassing both foundational research and cutting-edge advancements in the field.

To ensure the quality and relevance of the selected papers, we established stringent criteria for selection. These criteria included the publication date, with a focus on the most recent studies from 2020 to 2023, to capture the latest developments and trends. We also considered the relevance of TinyML and IoT, ensuring that the papers directly addressed the intersection of these technologies. Additionally, we prioritized the availability of full-text papers in English to facilitate thorough analysis and accessibility for a broader audience. From the selected papers, we extracted key information related to optimization techniques, applications, challenges, and future directions of TinyML. This extraction process was systematic and detailed, allowing us to organize the information comprehensively. We categorized the data to facilitate a structured analysis, ensuring that each aspect of TinyML was thoroughly examined. Our analysis involved identifying common themes, significant advancements, and gaps in the current research. We synthesized the findings into coherent sections, highlighting the main contributions of TinyML to various domains. This synthesis provided a clear and detailed overview of the state of the art in TinyML, emphasizing its potential and identifying areas for future research. The synthesized information was reviewed and validated by multiple authors to ensure accuracy and comprehensiveness. This collaborative review process involved cross-checking the data and resolving any discrepancies through discussion and consensus. By leveraging the expertise of multiple authors, we ensured that the final document was both accurate and comprehensive, providing valuable insights into the advancements in TinyML.

3.1. Tools and Framework for TinyML

The survey highlights the diversity and innovation within the TinyML landscape, focusing on tools and frameworks pivotal for IoT and edge devices. Table 1 presents a detailed examination of several tools and frameworks commonly utilized in TinyML applications. It includes information on the developers of each tool, the platforms they support, the typical applications they are used for, and key references associated with them. The table features well-known tools such as TensorFlow Lite, uTensor, Edge Impulse, NanoEdge AI Studio, and PyTorch Mobile. Each tool's unique features, diverse functionalities, and impact on the advancement of TinyML technology are highlighted, providing insight into their roles and contributions within the field. TensorFlow Lite [12]: Developed by the Google Brain Team, TensorFlow Lite supports a range of platforms including Android, iOS, embedded Linux, and microcontrollers. It is widely utilized for applications such as image/audio classification and object detection. The framework's versatility and robust model support make it ideal for real-time processing in smart cameras, voice-activated assistants, and various IoT devices. TensorFlow Lite's comprehensive cross-platform support enables seamless deployment of Machine Learning models on diverse hardware, from smartphones to embedded systems, enhancing its utility in creating intelligent applications across multiple environments. uTensor [31]: A product of ARM, uTensor is optimized for ARM's microcontroller ecosystem and supports platforms like Mbed and ST K64 ARM boards. It excels in image classification and gesture recognition, making it crucial for smart devices and wearables. uTensor's design emphasizes low latency and power efficiency, essential for battery-powered devices. This focus on optimization for ARM hardware ensures reliable and continuous operation in portable and wearable technology, positioning uTensor as a key player in enhancing the functionality of small, resource-constrained devices. PyTorch Mobile [32]: Originating from Meta AI, PyTorch Mobile extends the capabilities of the popular PyTorch framework to mobile and embedded devices, supporting Android, iOS, and Linux CPU platforms. It is tailored for applications in computer vision and natural language processing (NLP), including image recognition, object detection, and speech recognition. PyTorch Mobile's strength lies in its ability to efficiently run complex models on edge devices, enabling advanced AI capabilities in mobile and embedded systems. This flexibility and computational power make it a top choice for developers aiming to deploy sophisticated Machine Learning models across various platforms, thereby enhancing mobile and embedded applications with cutting-edge AI features. Edge Impulse [33]: This tool is particularly adept at edge-based asset tracking and predictive maintenance. Edge Impulse supports a variety of hardware platforms and is designed for rapid prototyping and deployment of TinyML models in edge devices. Its emphasis on ease of use and integration into existing workflows makes it suitable for applications in industrial and commercial environments where real-time data processing and analysis are critical. The platform's focus on practical deployment and actionable insights helps organizations leverage edge-based analytics for improved operational efficiency and predictive capabilities. NanoEdge AI Studio [34]: Focused on industrial applications, NanoEdge AI Studio specializes in anomaly detection and predictive maintenance. It provides tools for building and deploying Machine Learning models that can detect anomalies in operational data, contributing to increased efficiency and reduced downtime in industrial settings. Its emphasis on industrial-grade solutions highlights the importance of robust and reliable AI tools for critical operational environments, where precision and reliability are paramount.

MediaPipe [35], another tool developed by Google, provides a cross-platform framework for building real-time perception pipelines. It supports Android, iOS, Linux, and web platforms, and is widely used in applications such as hand and face tracking and pose estimation. MediaPipe's ability to handle complex computer vision tasks efficiently on edge devices makes it a crucial tool in the TinyML ecosystem.

Web of Science AI [36], developed by Clarivate, is a powerful research analytics tool designed to assist researchers in curating, analysing, and interpreting vast amounts of scientific literature. It leverages Artificial Intelligence to streamline the process of discovering trends, identifying key research areas, and understanding the impact of publications across various disciplines. Web of Science AI's integration with platforms like Windows, macOS, and Web allows researchers to access comprehensive datasets and utilize advanced metrics for citation analysis, collaboration networks, and research performance benchmarking. This tool is instrumental in providing insights that drive research strategies, funding opportunities, and policy decisions, making it a cornerstone for academic and industrial research environments.

On the other hand, OpenVINO [37], developed by Intel, is an open-source toolkit designed for optimizing and deploying deep learning models across a wide range of hardware platforms, including CPUs, GPUs, and specialized accelerators. It is particularly well-suited for computer vision applications, such as object detection, image segmentation, and facial recognition, due to its robust support for convolutional neural networks (CNNs) and other deep learning architectures. OpenVINO's versatility extends to its compatibility with Linux, Windows, and macOS, providing developers with tools for model optimization, inference acceleration, and seamless deployment across edge and cloud environments. Its ability to convert and optimize models from popular frameworks like TensorFlow and PyTorch further enhances its utility in developing high-performance AI applications. Together, Web of Science AI and OpenVINO represent cutting-edge tools in their respective domains, driving innovation in research analytics and deep learning inference. Each tool has unique strengths tailored to specific platforms and applications. TensorFlow Lite and PyTorch Mobile offer broad support and versatility, making them suitable for a wide range of use cases across different industries. uTensor and Edge Impulse provide specialized solutions for particular hardware and application needs, with uTensor focusing on ARM-based microcontrollers and Edge Impulse excelling in edge-based asset tracking and predictive maintenance. NanoEdge AI Studio caters to industrial applications with its focus on anomaly detection and predictive maintenance, offering critical insights and operational efficiencies in industrial settings.

Table 1. Overview of various tools and frameworks used in TinyML applications, detailing their developers, supported platforms, typical applications, and key references. This includes popular tools such as TensorFlow Lite, uTensor, Edge Impulse, NanoEdge AI Studio, PyTorch Mobile, and MediaPipe, highlighting their diverse capabilities and contributions to the field.

Tool/Framework	Developer	Supporting Platforms	Applications	References
TensorFlow Lite	Google Brain Team	Embedded Linux, Android	Image and Audio recognition, Object Tracking	[12]
uTensor	ARM	Mbed, ST K64 ARM Boards	Audio Classification, Gesture Recognition	[31]
Edge Impulse	Zach Shelby, Jan Jongboom	Android, iOS, Embedded Linux, Microcontrollers	Asset Tracking, Predictive Maintenance	[33]
NanoEdge AI Studio	Cartesiam	Android, Linux	Predictive Maintenance	[34]
PyTorch Mobile	Meta AI	Android, iOS, Linux CPU	Computer Vision	[32]
MediaPipe	Google	Android, iOS, Linux, Web	Hand/Face Tracking, Pose Estimation	[35]
Web of Science	Clarivate	Windows, macOS, Web	Research Analytics, Data Curation	[36]
OpenVINO	Intel	Linux, Windows, macOS	Deep Learning, Inference	[37]

The continuous development and refinement of these tools, driven by both major tech companies and innovative startups, are essential for advancing the field of TinyML. This diverse ecosystem enables smarter, more efficient IoT and edge devices by leveraging the latest advancements in hardware, software, and Machine Learning algorithms. As TinyML capabilities continue to expand, these tools and frameworks will play a crucial role in bringing intelligent decision making to the edge, transforming industries, and improving quality of life through enhanced technology solutions.

Programming Languages in TinyML Development

The development of TinyML applications involves a range of programming languages, each tailored to meet the unique demands of building Machine Learning models that can operate on resource-constrained devices like microcontrollers and edge systems. These languages are integral to different stages of TinyML development, from model training and optimization to deployment and execution on tiny hardware. Below is an analysis of the primary programming languages used in the TinyML ecosystem, highlighting their specific roles, strengths, and limitations.

- Python

Python is widely used for the initial development and training of Machine Learning models. It is the predominant language for frameworks like TensorFlow and PyTorch. Ease of Use: Python's simple syntax and extensive libraries make it accessible for both beginners and experts. Python has a vast array of libraries and frameworks (e.g., NumPy, TensorFlow, PyTorch) that streamline the development of Machine Learning models. A large and active community provides extensive resources, tutorials, and support. Python is an interpreted language and is slower than compiled languages like C++. This can be a limitation in environments where performance is critical. Python's runtime environment consumes significant memory and processing power, which is problematic for resource-constrained edge devices.

- C/C++

C and C++ are commonly used for the deployment and optimisation of TinyML models on microcontrollers. Libraries like TensorFlow Lite Micro and uTensor are written in C++ to ensure efficiency.: C/C++ provides high performance and low-level control over hardware, making it ideal for optimizing code to run on constrained devices. C/C++ allows precise management of memory usage, which is critical for devices with limited resources. These languages are supported by virtually all microcontrollers and embedded systems. C/C++ has a steeper learning curve compared to Python, requiring more effort to write and maintain code. Unlike Python, C/C++ lacks high-level libraries for Machine Learning, making the development process more time-consuming.

- JavaScript

JavaScript is used for developing web-based TinyML applications. Frameworks like TensorFlow.js allow models to be deployed directly in the browser or on Node.js servers JavaScript's primary advantage is its seamless integration with web technologies, making it possible to run ML models in the browser. JavaScript is platform-independent, running on any device with a web browser. The JavaScript ecosystem is large, with a wealth of libraries and tools available for web-based ML. Like Python, JavaScript is an interpreted language, leading to performance limitations in resource-constrained environments. JavaScript lacks the extensive Machine Learning libraries and frameworks found in Python, making it less suitable for complex ML tasks.

3.2. *Applications of TinyML*

Table 2 provides a comprehensive survey of state-of-the-art TinyML applications, illustrating its transformative impact across multiple domains by embedding Machine Learning capabilities into resource-constrained devices. In the realm of speech-based applications, TinyML facilitates real-time speech detection, recognition, and enhancement, as explored by Fedorov et al. [38] and Kwon and Park [39]. These advancements enable devices to understand and process spoken language locally, improving user interaction and maintaining privacy by minimizing cloud dependency. Vision-based applications, covered by Paul et al. [40] and Mohan et al. [41], leverage TinyML for tasks such as image classification, object detection, and gesture recognition.

These capabilities allow devices to interpret visual data directly at the edge, enhancing performance in areas like surveillance, security, and interactive systems, where immediate visual analysis is crucial. Data pattern classification, as demonstrated by Ren et al. [13], focuses on classifying and compressing data patterns on edge devices, optimizing data management and reducing bandwidth usage. This is essential for IoT devices that collect and analyse data continuously, enabling more efficient operations and conserving resources. In the healthcare domain, Dutta and Bharali [42] highlight how TinyML supports diagnostic applications such as respiratory monitoring and pose estimation. These applications enable continuous, real-time health monitoring and early detection of medical conditions, which is particularly valuable in remote or underserved areas with limited access to healthcare professionals. Edge computing benefits from TinyML, as discussed by Guleria et al. [43], by enhancing the performance and efficiency of edge devices through local data processing. This reduces the reliance on cloud infrastructure, speeds up decision making, and improves system scalability and reliability. Prado et al. [44] illustrate the application of TinyML in autonomous vehicles, where it is used for navigation and object detection, crucial for the safe and efficient operation of autonomous systems. By enabling real-time sensory data processing, TinyML helps autonomous vehicles detect obstacles and make decisions without constant cloud communication, thus improving safety and functionality. Overall, the table underscores TinyML's broad and transformative potential across diverse applications, driving advancements in real-time data processing, efficiency, and intelligent decision making at the edge of the network. The table presents an overview of the various domains in which TinyML technology is applied. It encompasses a range of areas such as speech recognition, computer vision, pattern classification, healthcare diagnostics, edge computing, and autonomous vehicles. For each category, the table includes a concise description of the application and provides references for additional reading. This comprehensive summary aims to highlight the versatility and impact of TinyML in different sectors. Environmental monitoring is another application area where TinyML is used to monitor conditions such as air quality, temperature, and humidity [45]. This application is crucial for maintaining environmental health and safety and providing timely data for decision making. In wearable technology, TinyML is integrated into devices for activity tracking, health monitoring, and fitness applications [46]. This integration allows for more personalized and accurate monitoring of users' health and fitness metrics. Smart agriculture leverages TinyML for precision farming, crop monitoring, and livestock management [47]. This application helps in optimizing agricultural practices, leading to better yield and resource management. Security and surveillance systems utilize TinyML for real-time threat detection and monitoring capabilities [48]. This application enhances the effectiveness of surveillance systems, providing timely alerts and improving security. TinyML applications, while transformative, face significant cybersecurity challenges due to their resource constraints, which limit the implementation of robust security measures. The small footprint of these devices increases their vulnerability to physical tampering and cyberattacks such as model extraction, adversarial attacks, and data poisoning. To address these issues, lightweight cryptography can secure data transmission and storage without overburdening device resources, while secure boot processes and firmware updates prevent unauthorized code execution. Real-time anomaly detection can help identify and mitigate threats proactively. Additionally, protecting models through obfuscation and secure enclaves safeguards intellectual property. Current trends enhancing TinyML cybersecurity include federated learning, which maintains data privacy by keeping data local while enabling collaborative model training, adversarial training to fortify models against manipulated inputs, blockchain integration for decentralized, tamper-resistant data exchange, and edge-to-cloud security frameworks for comprehensive protection. As TinyML expands across domains such as healthcare, surveillance, and autonomous vehicles, these advanced cybersecurity strategies are essential for protecting sensitive data, ensuring system integrity, and maintaining user trust.

TinyML utilizes lightweight algorithms optimized for resource-constrained edge devices. The most common algorithms include variants of neural networks such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and Decision Trees. These algorithms are tailored to run efficiently on microcontrollers with limited memory, processing power, and energy.

Key features of these algorithms include model compression, quantization, and pruning. Model compression reduces the model size without significantly sacrificing accuracy. Quantization converts model parameters to lower precision, such as from 32-bit floating point to 8-bit integers, reducing memory usage and computational demand. Pruning eliminates less critical parts of the network, streamlining the model further. TinyML architectures often follow a modular design, with layers dedicated to input preprocessing, feature extraction, and decision making. For instance, in a CNN, the convolutional layers handle feature extraction, while fully connected layers make decisions based on the extracted features. Practical examples include voice recognition systems like Google's Keyword Spotting, image classification for real-time object detection, and anomaly detection in IoT devices. Future improvements in TinyML will focus on developing more efficient algorithms, enhancing model interpretability, and improving adaptability to various hardware platforms, enabling broader adoption in fields like healthcare, agriculture, and smart cities.

Table 2. TinyML applications across various domains. This table summarizes the different application areas where TinyML is utilized, including speech, vision, data pattern classification, health diagnosis, edge computing, and autonomous vehicles. Each entry provides a brief description and references for further reading.

Application Area	Description	References
Speech-Based Applications	Includes speech detection, recognition, and enhancement for devices.	[38,39]
Vision-Based Applications	Encompasses image classification, object detection, and gesture recognition.	[40,41]
Data Pattern Classification	Focuses on classifying and compressing data patterns on edge devices.	[13]
Health Diagnosis	Involves diagnostic applications like respiratory monitoring and pose estimation.	[42]
Edge Computing	Utilizes TinyML for enhancing edge device performance and efficiency.	[43]
Autonomous Vehicles	Applies TinyML for navigation and object detection in autonomous systems.	[44]
Environmental Monitoring	Used for monitoring environmental conditions such as air quality, temperature, humidity.	[45]
Wearable Technology	Integrates TinyML in wearable devices for activity tracking, health monitoring, and fitness applications	[46]
Smart Agriculture	Implements TinyML for precision farming, crop monitoring, and livestock management	[47]
Security and Surveillance	Enables intelligent surveillance systems with real-time threat detection and monitoring capabilities.	[48]

Emulators for TinyML Development

Emulators play a crucial role in the development and testing of TinyML applications, allowing developers to simulate the behaviour of microcontrollers and other edge devices in a controlled environment before deployment on actual hardware. These tools are essential for debugging and ensuring the reliability of code, particularly in resource-constrained settings. Open-source emulators like QEMU and Renode are popular choices for their cost-effectiveness and flexibility. QEMU supports a wide range of architectures, including ARM, making it a versatile option for simulating microcontroller environments, though it requires manual setup and configuration. Renode, specifically designed for IoT and edge devices, offers advanced debugging capabilities and can simulate entire networks of devices, making it particularly useful for complex TinyML projects. However, open-source emulators often come with the challenges of complex setup and potentially slower performance, which can affect the accuracy of performance testing. On the other hand, commercial emulators such as Keil MDK and Proteus provide user-friendly interfaces and professional support, making them easier to use and integrate into development workflows. Keil MDK is particularly well suited for ARM-based microcontrollers, offering advanced features and seamless integration with hardware debugging tools. Proteus supports a wide range of microcontroller platforms and provides extensive simulation features. Despite their advantages, commercial emulators can be expensive, limiting their accessibility to smaller development teams or hobbyists, and may also lead to vendor lock-in, restricting flexibility in hardware choices. Table 3 presents a comprehensive comparison of TinyML-compatible hardware, focusing on their processors, memory, computational power, power consumption, and typical use cases. The STWIN platform [49] features an

ARM Cortex-M4 processor with 640 KB of SRAM and 2 MB of Flash memory, operating at 80 MHz. Its low power consumption makes it well suited for industrial IoT applications, where reliability and efficient sensor integration are critical. This hardware is designed for applications such as condition monitoring and vibration analysis, leveraging its industrial-grade sensors and BLE connectivity to provide robust data transmission and extensive interfacing capabilities. The Arduino Nano 33 BLE [50], also equipped with an ARM Cortex-M4 processor, offers 256 KB of SRAM and 1 MB of Flash memory, running at 64 MHz. It is known for its ultra-low power consumption, making it ideal for battery-operated devices. Its integrated Bluetooth 5.0 and 9-axis IMU facilitate advanced applications like wearable health monitoring and motion tracking, where its compact design and precise sensing are key advantages. The Edge TPU [51], a custom ASIC, excels in hardware acceleration for Machine Learning tasks, delivering 4 TOPS (Tera Operations Per Second) with very low power consumption (approximately 2 W). This specialized hardware is optimized for inferencing tasks such as image classification and object detection, providing high computational efficiency and minimal latency, which are essential for real-time AI applications in edge computing environments. The Raspberry Pi Pico [52] features an ARM Cortex-M0+ dual-core processor, with 264 KB of SRAM and 2 MB of Flash memory, operating at 133 MHz. It balances low power consumption with adequate performance for basic Machine Learning tasks and educational projects. Its cost-effectiveness and simplicity make it a popular choice for prototyping and learning applications, where straightforward functionality and affordability are prioritized. This analysis underscores how each hardware option caters to different TinyML requirements, offering a range of memory capacities, power efficiencies, and computational capabilities to address various application needs effectively.

Table 3. Detailed specifications and capabilities of TinyML hardware options.

Hardware	Processor/MCU	Special Features	Typical Use Cases	References
STWIN	ARM Cortex-M4	Industrial-grade, sensors, BLE connectivity	Industrial IoTs	[49]
Arduino Nano 33 BLE	ARM Cortex-M4	Bluetooth 5.0, 9-axis IMU	Wearable, remote sensing	[50]
Edge TPU	Custom ASIC	Hardware acceleration For ML	Image classification, and object detection	[51]
Raspberry Pi	ARM Cortex-M0+	Dual-core processors	Simple Machine Learning Tasks	[52]

3.3. Limitations of TinyML

The deployment of TinyML on IoT and edge devices presents several intricate challenges that affect the performance and feasibility of Machine Learning models in resource-constrained environments. One of the foremost concerns is power consumption. Edge devices are typically limited in their energy resources, making it challenging to run complex ML models efficiently. To mitigate this, techniques such as model quantization and pruning are vital. Quantization reduces the precision of the model's weights and activations, while pruning eliminates less significant components, both contributing to reduced energy consumption without substantial losses in functionality [36]. For example, Google's Edge TPU employs quantization to deliver high performance with minimal power usage, illustrating how these techniques can effectively balance power and performance.

Memory constraints further complicate these issues, as the limited memory on edge devices restricts the size and complexity of deployable models. This often necessitates the use of compact neural network architectures and model compression strategies. Research into lightweight models, such as MobileNet and EfficientNet, highlights how these architectures are specifically designed to fit within memory limitations while maintaining competitive performance [53]. For instance, MobileNet's use of depthwise separable convolutions significantly reduces the model size and computational load, making it suitable for edge devices with limited memory. Additionally, the processing capabilities of edge devices are generally lower than those of traditional servers, leading to slower inference times and reduced responsiveness, particularly in real-time applications. To address this, hardware accelerators like specialized processors and FPGAs are employed. These accelerators are designed to handle specific computations more efficiently, thus enhancing performance without imposing excessive computational demands [54]. An example is Intel's Movidius Myriad X, which is optimized for neural network inference and provides substantial performance improvements over conventional processors. Another significant issue is the lack of standardization in TinyML frameworks and benchmarks, which results in inconsistencies and compatibility issues across different devices and applications. Standardizing tools, protocols, and evaluation metrics is crucial for facilitating deployment and ensuring interoperability [55]. For instance, the lack of uniform benchmarks makes it challenging to compare the performance of TinyML models across different platforms, hindering the development of universally applicable solutions. Latency is another critical challenge, particularly for real-time applications that require extremely low latency. Achieving such performance with TinyML on edge devices is difficult due to their limited processing power and memory. Techniques to address latency include optimizing algorithms and leveraging edge–cloud computing architectures, where edge devices handle simple tasks and offload complex computations to cloud servers [56]. This hybrid approach can help mitigate the processing constraints of edge devices. Data privacy is also a major concern when handling sensitive information locally on edge devices. Ensuring robust privacy-preserving techniques is essential to safeguard user data from unauthorized access and breaches [57]. For example, techniques like federated learning enable models to be trained across multiple devices without transferring raw data, thus enhancing privacy. Deploying models on millions of distributed edge devices presents logistical challenges in updating and maintaining these models. Streamlining this process requires effective strategies to manage and distribute updates across a vast network [58]. Solutions such as over-the-air updates and version control mechanisms are critical for ensuring that all devices remain current and functional. Addressing these challenges through innovative optimization techniques, efficient hardware design, and the development of universal standards is essential for advancing TinyML's integration into diverse IoT and edge computing scenarios. Table 4 details the challenges and considerations for deploying TinyML on edge devices

Table 4. Challenges and considerations in deploying TinyML on edge devices. This table outlines key obstacles such as power consumption, memory constraints, processing capacity limitations, and the lack of standardization, along with relevant references for further reading.

Challenge	Description	References
Power Consumption	Edge devices have limited power, making efficient ML model execution challenging.	[36]
Memory Constraints	TinyML models need to operate within limited memory available on edge devices.	[53]
Processing Capacity	Edge devices have lower processing capabilities compared to traditional servers.	[54]
Standardization	Lack of standardized frameworks and benchmarks for TinyML deployment.	[55]
Latency Requirements	Real-time applications require low latency, which can be difficult to achieve with TinyML on edge devices.	[56]
Data Privacy	Handling sensitive data locally on edge devices necessitates strong privacy-preserving techniques.	[57]
Model Deployment	Deploying and updating models on millions of distributed edge devices can be logistically challenging.	[58]

4. Conclusions and Future Directions

4.1. Future Directions

Future research directions for advancing TinyML in the context of IoT and edge devices highlight several critical areas of focus. One significant area is the development of energy-efficient TinyML models, which aims to optimize Machine Learning algorithms for low-power devices without compromising accuracy (Immonen and Hämäläinen, 2022) [59]. This involves implementing model optimization techniques such as quantization, pruning, and hardware integration to enhance power efficiency while maintaining model performance. Another crucial direction is enhanced memory management, which seeks to improve techniques for managing memory to accommodate more complex ML models on resource-constrained devices (Han and Siebert, 2022) [60]. Table 5 provides an overview of significant research areas in TinyML. It includes initiatives aimed at creating energy-efficient models, improving memory management, optimizing the integration of edge and cloud computing, and advancing standardization efforts. Each area is detailed with pertinent references to recent research. This includes employing memory optimization techniques, efficient data handling strategies, and advanced compression algorithms to manage and reduce the memory footprint of models. Collaborative edge–cloud models are also a focal point, exploring how edge devices and cloud servers can work together to optimize computing resources and performance Bao et al. [61]. This approach leverages hybrid processing, effective data aggregation, and load balancing to combine the strengths of both edge and cloud computing, reducing latency and bandwidth usage while enhancing overall system efficiency. Additionally, standardization efforts are essential for developing consistent frameworks and benchmarks for TinyML Shafique et al. [55]. Establishing standardized tools, benchmarks, and metrics will facilitate the creation, deployment, and evaluation of TinyML models across diverse platforms, ensuring consistency, interoperability, and improved integration. Research in this area, as discussed in [47], aims to make AI more environmentally friendly by reducing energy consumption and resource usage. TinyML for real-time applications is a critical focus, particularly for mission-critical tasks such as autonomous vehicles. Ensuring that TinyML models can operate effectively in real-time scenarios is essential for applications that require immediate decision making and response. The challenges and solutions in this area are explored in [62]. Hardware-software co-design represents an integrated approach where both hardware and software are designed together to optimize performance in TinyML applications. This co-design strategy, discussed by [39], aims to create more efficient and powerful systems by aligning the capabilities of hardware with the requirements of software.

Subsequent investigations in TinyML should prioritise the following domains in order to tackle the recognised obstacles and propel the discipline forward:

- Compact and power-saving TinyML models: It is essential to develop Machine Learning models that are optimised for energy conservation while maintaining high accuracy in order to ensure the sustainable implementation of TinyML on-edge devices. Research should investigate innovative methods for model quantisation pruning and compression. Enhanced Memory Management: It is crucial to enhance memory management strategies in order to enable the deployment of more intricate Machine Learning models on edge devices with limited memory. This involves creating algorithms that adaptively optimise memory utilisation according to the available resources. Collaborative edge–cloud models involve the exploration and optimisation of computing models that leverage the combined capabilities of edge devices and cloud servers to efficiently handle Machine Learning workloads. This approach is particularly useful in overcoming the processing capacity constraints of edge devices. Efforts to establish a set of standardised practices or guidelines: Facilitating the development of standardised frameworks and benchmarks for TinyML would promote its wider adoption and compatibility across many platforms and applications.

Table 5. Table summarizing key research directions in TinyML, including efforts to develop energy-efficient models, enhance memory management, optimize collaborative edge–cloud computing, and advance standardization. Each direction is described with relevant references to recent studies.

Research Direction	Description	References
Energy-Efficient TinyML Models	Developing ML models that are optimized for energy efficiency without sacrificing accuracy.	[59]
Enhanced Memory Management	Improving memory management techniques to allow the deployment of more complex ML models.	[60]
Collaborative Edge–Cloud Models	Exploring and optimizing collaborative computing models where edge devices and cloud servers work together.	[61]
Standardization Efforts	Contributing to the development of standardized frameworks and benchmarks for TinyML.	[55]
Sustainable AI with TinyML	Exploring the use of TinyML in promoting sustainable AI practices, minimizing carbon footprints of deployments	[47]
TinyML for Real-Time Applications	Focusing on the deployment of TinyML models in real-time, mission-critical applications like autonomous vehicles.	[62]
Hardware-Software Co-Design	Creating integrated approaches where HW and SW are co-designed to optimize performance in TinyML.	[39]

4.2. Conclusions

The advancements in TinyML represent a significant leap forward in the integration of Machine Learning with resource-constrained devices, enabling intelligent decision making at the edge of the network. This review has highlighted the transformative potential of TinyML across various domains, including healthcare, agriculture, industrial automation, and environmental monitoring. By leveraging optimization techniques such as model quantization and pruning, TinyML models can operate efficiently on low-power devices, making them suitable for a wide range of applications. Despite the promising developments, several challenges remain, including power consumption, memory constraints, and the need for standardized frameworks. Addressing these challenges requires ongoing research and innovation in both hardware and software. The integration of TinyML with IoT devices offers significant benefits, such as enhanced real-time data processing and reduced latency, which are crucial for the effective deployment of intelligent systems. Future research directions should focus on developing energy-efficient models, improving memory management techniques, and exploring collaborative edge–cloud computing models. Additionally, efforts to standardize tools and frameworks will be essential for the widespread adoption of TinyML. As the field continues to evolve, the potential for TinyML to revolutionize industries and improve quality of life becomes increasingly apparent. TinyML stands at the forefront of a new era in Machine Learning, bringing powerful AI capabilities to the smallest of devices. The continued advancement and adoption of TinyML hold the promise of a smarter, more efficient, and more connected world.

Author Contributions: Conceptualization, A.E.; methodology, A.E. and P.D.; software, A.E. and Q.Z.; validation, A.E.; formal analysis, A.E. and S.S.; investigation, A.E. and Q.Z.; resources, A.E., P.D., S.S. and Q.Z.; data curation, A.E.; writing—original draft preparation, A.E. and P.D.; writing—review and editing, A.E. and Q.Z.; visualization, A.E. and Q.Z.; supervision, A.E. and S.S.; project administration, A.E., P.D., S.S. and Q.Z.; funding acquisition, A.E. and S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the "Horizon Europe program under grant agreement 101092850 (AERO project)"; by the "European High-Performance Computing Joint Undertaking (JU) program under grant agreement 101033975 (EUPEX)"; and by "PNRR project CN1 Big Data, HPC and Quantum Computing in Spoke 6 multiscale modelling and engineering applications".

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shi, W.; Dustdar, S. The Promise of Edge Computing. *Computer* **2016**, *49*, 78–81. [CrossRef]
2. Satyanarayanan, M. The Emergence of Edge Computing. *Computer* **2017**, *50*, 30–39. [CrossRef]
3. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
4. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
5. Elhanashi, A.; Saponara, S.; Zheng, Q. Classification and Localization of Multi-Type Abnormalities on Chest X-rays Images. *IEEE Access* **2023**, *11*, 83264–83277. [CrossRef]
6. Zheng, Q.; Wang, R.; Tian, X.; Yu, Z.; Wang, H.; Elhanashi, A.; Saponara, S. A real-time transformer discharge pattern recognition method based on CNN-LSTM driven by few-shot learning. *Electr. Power Syst. Res.* **2023**, *219*, 109241. [CrossRef]
7. Elhanashi, A.; Saponara, S.; Dini, P.; Zheng, Q.; Morita, D.; Raytchev, B. An integrated and real-time social distancing, mask detection, and facial temperature video measurement system for pandemic monitoring. *J. Real-Time Image Process.* **2023**, *20*, 95. [CrossRef]
8. Sze, V.; Chen, Y.-H.; Yang, T.-J.; Emer, J.S. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proc. IEEE* **2017**, *105*, 2295–2329. [CrossRef]
9. Zhao, Z.; Zhang, L.; Xu, Y.; Liang, W. Machine learning-based networking: Concepts, applications and challenges. *IEEE Netw.* **2018**, *32*, 78–85.
10. Lane, N.D.; Bhattacharya, S.; Mathur, A.; Georgiev, P.; Forlivesi, C.; Kawsar, F. Squeezing Deep Learning into Mobile and Embedded Devices. *IEEE Pervasive Comput.* **2015**, *14*, 82–88. [CrossRef]
11. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016.
12. Warden, P.; Situnayake, D. *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*; O'Reilly Media: Sebastopol, CA, USA, 2019.
13. Ren, H.; Anicic, D.; Runkler, T.A. Tinyol: Tinyml with online-learning on microcontrollers. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021.
14. Giordano, M.; Baumann, N.; Crabolu, M.; Fischer, R.; Bellusci, G.; Magno, M. Design and Performance Evaluation of an Ultralow-Power Smart IoT Device With Embedded TinyML for Asset Activity Monitoring. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–11. [CrossRef]
15. Sudharsan, B.; Salerno, S.; Yadav, P.; Breslin, J.G. Approach for Remote, On-Demand loading and Execution of TensorFlow Lite ML Models on Arduino IoT Boards. In Proceedings of the 2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Milano, Italy, 4–6 May 2022.
16. Google AI Blog. *Introducing Edge TPU: Our New Edge Computing Platform for Machine Learning*; Google AI Blog: Mountain View, CA, USA, 2018.
17. Intel Corporation. *Intel Movidius Myriad X Vision Processing Unit (VPU)*; Intel: Santa Clara, CA, USA, 2019.
18. Wu, X.; Wang, M.; Lin, J.; Wang, Z. Amoeba: An Efficient and Flexible FPGA-Based Accelerator for Arbitrary-Kernel CNNs. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2024**, *32*, 1086–1099. [CrossRef]
19. Zaman, K.S.; Reaz, M.B.I.; Ali, S.H.M.; Bakar, A.A.A.; Chowdhury, M.E.H. Custom hardware architectures for deep learning on portable devices: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *33*, 6068–6088. [CrossRef]
20. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *arXiv* **2017**, arXiv:1712.05877.
21. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
22. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2019**, arXiv:1905.11946.
23. TensorFlow. *TensorFlow Lite for Microcontrollers*; TensorFlow: Mountain View, CA, USA, 2020.
24. Edge Impulse. *Edge Impulse Documentation*; Edge Impulse: San Jose, CA, USA, 2021.
25. Charan Sai, K. An auto-encoder based TinyML approach for real-time anomaly detection. *SAE Int. J. Adv. Curr. Pract. Mobil.* **2022**, *5*, 1496–1501. [CrossRef]
26. Diab, M.S.; Rodriguez-Villegas, E. Embedded machine learning using microcontrollers in wearable and ambulatory systems for health and care applications: A review. *IEEE Access* **2022**, *10*, 98450–98474. [CrossRef]
27. Zhang, S.; Li, Y.; Zhang, S.; Shahabi, F.; Xia, S.; Deng, Y.; Alshurafa, N. Deep learning in human activity recognition with wearable sensors: A review on advances. *Sensors* **2022**, *22*, 1476. [CrossRef] [PubMed]

28. Kumar, M.; Zhang, X.; Liu, L.; Wang, Y.; Shi, W. Energy-efficient machine learning on the edges. In Proceedings of the 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), New Orleans, LA, USA, 18–22 May 2020.
29. Zhou, X.; Xu, K.; Wang, N.; Jiao, J.; Dong, N.; Han, M.; Xu, H. A secure and privacy-preserving machine learning model sharing scheme for edge-enabled IoT. *IEEE Access* **2021**, *9*, 17256–17265. [CrossRef]
30. Rahman, M.A.; Asyhari, A.T.; Leong, L.S.; Satrya, G.B.; Tao, M.H.; Zolkipli, M.F. Scalable machine learning-based intrusion detection system for IoT-enabled smart cities. *Sustain. Cities Soc.* **2020**, *61*, 102324. [CrossRef]
31. Profentzas, C.; Almgren, M.; Landsiedel, O. Performance of deep neural networks on low-power IoT devices. In Proceedings of the Workshop on Benchmarking Cyber-Physical Systems and Internet of Things, Virtual, 3 May 2021.
32. Ignatov, A.; Malivenko, G.; Plowman, D.; Shukla, S.; Timofte, R. Fast and accurate single-image depth estimation on mobile devices, mobile ai 2021 challenge: Report. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021.
33. Hymel, S.; Banbury, C.; Situnayake, D.; Elium, A.; Ward, C.; Kelcey, M.; Baaijens, M.; Majchrzycki, M.; Plunkett, J.; Tischler, D.; et al. Edge impulse: An mlops platform for tiny machine learning. In Proceedings of the Machine Learning and Systems 5, Dalian, China, 21–23 July 2023.
34. Cartesiam. NanoEdge AI Studio: The Leading Tool for Anomaly Detection. 2021. Available online: https://blog.st.com/nanoedge-ai-studio/ (accessed on 3 September 2024).
35. Pauzi, A.S.B.; Mohd Nazri, F.B.; Sani, S.; Bataineh, A.M.; Hisyam, M.N.; Jaafar, M.H.; Ab Wahab, M.N.; Mohamed, A.S.A. Movement estimation using mediapipe blazepose. In Proceedings of the Advances in Visual Informatics: 7th International Visual Informatics Conference, IVIC 2021, Kajang, Malaysia, 23–25 November 2021; Proceedings 7; Springer International Publishing: Berlin/Heidelberg, Germany, 2021.
36. Alajlan, N.N.; Ibrahim, D.M. TinyML: Enabling of Inference Deep Learning Models on Ultra-Low-Power IoT Edge Devices for AI Applications. *Micromachines* **2022**, *13*, 851. [CrossRef] [PubMed]
37. de Prado Escudero, M. On Automation for Optimised and Robust Deployment of Neural Networks on Edge Devices. Ph.D. Thesis, ETH Zurich, Zurich, Switzerland, 2021.
38. Fedorov, I.; Stamenovic, M.; Jensen, C.; Yang, L.-C.; Mandell, A.; Gan, Y.; Mattina, M.; Whatmough, P.N. TinyLSTMs: Efficient neural speech enhancement for hearing aids. *arXiv* **2020**, arXiv:2005.11138.
39. Kwon, J.; Park, D. Hardware/Software Co-Design for TinyML Voice-Recognition Application on Resource Frugal Edge Devices. *Appl. Sci.* **2021**, *11*, 11073. [CrossRef]
40. Paul, A.J.; Mohan, P.; Sehgal, S. Rethinking Generalization in American Sign Language Prediction for Edge Devices with Extremely Low Memory Footprint. In Proceedings of the 2020 IEEE Recent Advances in Intelligent Computational Systems, RAICS 2020, Thiruvananthapuram, India, 3–5 December 2020; pp. 147–152.
41. Mohan, P.; Paul, A.J.; Chirania, A. A tiny cnn architecture for medical face mask detection for resource-constrained endpoints. In *Advances in Intelligent Systems and Computing*; Springer: Berlin/Heidelberg, Germany, 2021; Volume 657, pp. 657–670.
42. Dutta, D.L.; Bharali, S. TinyML Meets IoT: A Comprehensive Survey. *Internet Things* **2021**, *16*, 100461. [CrossRef]
43. Guleria, C.; Das, K.; Sahu, A. A survey on mobile edge computing: Efficient energy management system. In Proceedings of the 2021 Innovations in Energy Management and Renewable Resources (52042), Kolkata, India, 5–7 February 2021; pp. 1–4.
44. de Prado, M.; Rusci, M.; Capotondi, A.; Donze, R.; Benini, L.; Pazos, N. Robustifying the deployment of tinyML models for autonomous mini-vehicles. *Sensors* **2021**, *21*, 1339. [CrossRef]
45. González Rivero, R.A.; Morera Hernández, L.E.; Schalm, O.; Hernández Rodríguez, E.; Alejo Sánchez, D.; Morales Pérez, M.C.; Nuñez Caraballo, V.; Jacobs, W.; Martinez Laguardia, A. A low-cost calibration method for temperature, relative humidity, and carbon dioxide sensors used in air quality monitoring systems. *Atmosphere* **2023**, *14*, 191. [CrossRef]
46. Ometov, A.; Shubina, V.; Klus, L.; Skibińska, J.; Saafi, S.; Pascacio, P.; Flueratoru, L.; Gaibor, D.Q.; Chukhno, N.; Chukhno, O.; et al. A survey on wearable technology: History, state-of-the-art and current challenges. *Comput. Netw.* **2021**, *193*, 108074. [CrossRef]
47. Abadade, Y.; Temouden, A.; Bamoumen, H.; Benamar, N.; Chtouki, Y.; Hafid, A.S. A Comprehensive Survey on TinyML. *IEEE Access* **2023**, *11*, 96892–96922. [CrossRef]
48. Ahmed, A.A.; Echi, M. Hawk-Eye: An AI-Powered Threat Detector for Intelligent Surveillance Cameras. *IEEE Access* **2021**, *9*, 63283–63293. [CrossRef]
49. Loizzi, G. Development of a Wearable Device for Breathing Rate Monitoring Using Stretchable Sensors. Ph.D. Thesis, Politecnico di Torino, Turin, Italy, 2021.
50. Kurniawan, A. *IoT Projects with Arduino Nano 33 BLE Sense*; Apress: Berkeley, CA, USA, 2021; Volume 129.
51. Reidy, B.C.; Mohammadi, M.; Elbtity, M.E.; Zand, R. Efficient deployment of transformer models on edge tpu accelerators: A real system evaluation. In *Architecture and System Support for Transformer Models (ASSYST@ ISCA 2023)*; ISCA 2023 Workshop: Orlando, FL, USA, 2023.
52. Jolles, J.W. Broad-scale applications of the Raspberry Pi: A review and guide for biologists. *Methods Ecol. Evol.* **2021**, *12*, 1562–1579. [CrossRef]
53. Goudarzi, M.; Palaniswami, M.S.; Buyya, R. A Distributed Deep Reinforcement Learning Technique for Application Placement in Edge and Fog Computing Environments. *IEEE Trans. Mob. Comput.* **2021**, *22*, 2491–2505. [CrossRef]
54. Muhammad, G.; Hossain, M.S. Emotion Recognition for Cognitive Edge Computing Using Deep Learning. *IEEE Internet Things J.* **2021**, *8*, 16894–16901. [CrossRef]

55. Shafique, M.; Theocharides, T.; Reddy, V.J.; Murmann, B. TinyML: Current Progress, Research Challenges, and Future Roadmap. In Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 5–9 December 2021; pp. 1303–1306.
56. Xu, K.; Zhang, H.; Li, Y.; Zhang, Y.; Lai, R.; Liu, Y. An Ultra-Low Power TinyML System for Real-Time Visual Processing at Edge. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 2640–2644. [CrossRef]
57. Bi, M.; Wang, Y.; Cai, Z.; Tong, X. A privacy-preserving mechanism based on local differential privacy in edge computing. *China Commun.* **2020**, *17*, 50–65. [CrossRef]
58. Belcastro, L.; Marozzo, F.; Orsino, A.; Talia, D.; Trunfio, P. Edge-Cloud Continuum Solutions for Urban Mobility Prediction and Planning. *IEEE Access* **2023**, *11*, 38864–38874. [CrossRef]
59. Immonen, R.; Hämäläinen, T. Tiny Machine Learning for Resource-Constrained Microcontrollers. *J. Sens.* **2022**, *2022*, 7437023. [CrossRef]
60. Han, H.; Siebert, J. TinyML: A Systematic Review and Synthesis of Existing Research. In Proceedings of the International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Republic of Korea, 21–24 February 2022; pp. 269–274.
61. Bao, G.; Guo, P. Federated learning in cloud-edge collaborative architecture: Key technologies, applications and challenges. *J. Cloud Comput.* **2022**, *11*, 94. [CrossRef]
62. Koufos, K.; EI Haloui, K.; Dianati, M.; Higgins, M.; Elmirghani, J.; Imran, M.A.; Tafazolli, R. Trends in intelligent communication systems: Review of standards, major research projects, and identification of research gaps. *J. Sens. Actuator Netw.* **2021**, *10*, 60. [CrossRef]