UDC 001.8
DOI https://doi.org/10.26661/2786-6254-2024-2-07

PROBLEMS OF MACHINE LEARNING IMPLEMENTATION ON MODERN EDGE DEVICES

Melnychuk O. S.
Postgraduate Student at the Department of CDS
Lviv Polytechnic National University
Stepana Bandery str., 12, Lviv, Ukraine
orcid.org/0009-0008-2385-1488
oleksandr.s.melnychuk@lpnu.ua

Lobur M. V.
Ph.D. Hab., Professor at the Department of CDS
Lviv Polytechnic National University
Stepana Bandery str., 12, Lviv, Ukraine
orcid.org/0000-0001-7516-1093
mykhaylo.v.lobur@lpnu.ua

Sviridova T. V.
Ph.D, Director of Engineering
Epam Systems
Uhorska str., 14, Lviv, Ukraine
orcid.org/0009-0008-4332-8109
tetyana_sviridova@epam.com

Key words: *TinyML, embedded AI, edge computing, Internet of Things, microcontrollers.*

Recently, artificial intelligence (AI) and machine learning (ML) have attracted considerable attention in both industry and academia. Since traditional ML methods are energy-demanding, they can be applied only to a limited subset of devices with significant computing capabilities. However, the rise in complexity of available processors and microcontrollers made it possible to integrate ML methods into data pipelines on mobile devices, and later – on low-power (few milliwatts) edge devices. This concept finally took shape in 2018 and was named TinyML. Its rapid spread is fueled by the huge number of manufactured microcontrollers (250 billion) and the popularity of the Internet of Things (IoT). In the future, TinyML can be used on every device that is part of the IoT, although this will not always be economically justified.
At the same time, the application of TinyML technologies is not straightforward and requires a careful assessment of the capabilities as well as selection of both the ML algorithm and the peripheral device. This is explained primarily by the lack of generally accepted benchmarks of both TinyML algorithms and hardware capabilities for the application of ML. There are several approaches to TinyML implementation: software-oriented, hardware-oriented, and hybrid. Depending on the approach, different methods of solving a specific problem are used. Some ML algorithms can be adopted to edge devices through the use of simplification and adaptation techniques tailored for neural networks (NN). However, the process is not automated and generalized as of now.
This article is devoted to an overview of the concept of TinyML, its main stages and features, the most prominent achievements in the fields of speech and image recognition, sequence classification and data compression, health diagnostics and brain interaction, prediction of equipment malfunctions and anomaly detection, autonomous transport and ecology. Unfortunately, the volume of the article does not allow us to reveal the specifics of TinyML's application in other fields and with more details. In addition, the article analyzes the problems that arise during ML implementation on low-power devices. The purpose of the article is to become a short guide and roadmap to the world of TinyML applications.

ПРОБЛЕМИ ВПРОВАДЖЕННЯ МАШИННОГО НАВЧАННЯ В СУЧАСНІ ПЕРИФЕРІЙНІ ПРИСТРОЇ

Мельничук О. С.
аспірант кафедри САП
Національний університет «Львівська політехніка»
вул. Степана Бандери, 12, Львів, Україна
orcid.org/0009-0008-2385-1488
oleksandr.s.melnychuk@lpnu.ua

Лобур М. В.
доктор технічних наук, професор кафедри САП
Національний університет «Львівська політехніка»
вул. Степана Бандери, 12, Львів, Україна
orcid.org/0000-0001-7516-1093
mykhaylo.v.lobur@lpnu.ua

Свірідова Т. В.
кандидат технічних наук, директор з інженерії
ЕПАМ Системз
вул. Угорська, 14, Львів, Україна
orcid.org/0009-0008-4332-8109
tetyana_sviridova@epam.com

Ключові слова: *TinyML, вбудований ШІ, периферійні обчислення, інтернет речей, мікроконтролери.*

Останнім часом штучний інтелект (ШІ) та машинне навчання (МН) привернули значну увагу як на виробництві, так і в академічних колах. Оскільки традиційні методи МН є енергоємними, то це обмежує їх застосування підмножиною пристроїв зі значними обчислювальними можливостями. Однак зростання потужності процесорів та мікроконтролерів дозволило інтегрувати методи МН для обробки даних спершу на мобільні пристрої, а пізніше – на малопотужні (кілька міліват) периферійні пристрої. Ця концепція остаточно сформувалась у 2018 році та отримала назву TinyML. Її стрімке поширення підживлюється велетенською кількістю вироблених мікроконтролерів (250 мільярдів) та популярністю інтернету речей (ІР). У перспективі, TinyML може бути використаний на кожному пристрої що є частиною ІР, хоча це не завжди буде економічно виправдано.

Водночас застосування технологій TinyML не є простим і потребує ретельної оцінки та вибору можливостей як алгоритму МН так і периферійного пристрою. Це пояснюється насамперед відсутністю загальноприйнятих способів порівняння як алгоритмів TinyML, так і спроможностей апаратного забезпечення щодо застосування МН. Існує кілька підходів до впровадження TinyML: орієнтований на програмне забезпечення, апаратне забезпечення та гібридний. Залежно від підходу застосовуються різні методи вирішення конкретної задачі. Деякі алгоритми МН можуть бути перенесені на периферійні пристрої досить просто, через застосування прийомів спрощення та адаптації нейронних мереж (НМ). Однак на сьогодні їх застосування не є автоматизованим та узагальненим.

Ця стаття присвячена огляду концепції TinyML, основних етапів та особливостей її застосування, найбільш помітних досягнень в галузях розпізнавання мови та зображень, класифікації послідовностей та стиснення даних, діагностики здоров'я та взаємодії з мозком, прогнозування несправностей техніки та виявлення аномалій, автономного транспорту та екології. На жаль, об'єм статті не дозволяє розкрити особливості застосування TinyML в решті галузей та більш детально. Крім того, в статті проаналізовані проблеми, які виникають при впровадженні технологій МН на малопотужних пристроях. Мета статті – стати путівником та дороговказом у світ застосувань TinyML.

Introduction. TinyML plays a pivotal role in the Industry 4.0 [1] and Industry 5.0 [2] revolutions, facilitating the integration of AI-powered computing technologies across various sectors, including smart cities, the automotive industry, and medical robotics. Over the last several decades, considerable effort has been put into advancing Machine Learning (ML) from cloud premises to mobile devices and further to the edge nodes [3]. Specifically, low-power (under a milliwatt) embedded devices based on Micro-Controller Units (MCUs) have garnered significant attention. Each of these "tiny" devices doesn't exceed few hundred Kilobytes in memory, and several Megabytes of flash storage where ML models can be stored and operated. But there are more than 250 billion MCUs in use already and the number grows every year [3]. This is largely due to their minimal power consumption and low cost, along with their versatility and reliability. By integrating with sensors, these systems gain the ability to perceive their surroundings; connecting to actuators, they can perform various actions, and through interconnectivity, they enable the spread of distributed intelligence. Embedded technologies form the backbone of the Internet of Things (IoT) ecosystem and its myriad smart applications: from smart buildings and cities to smart metering, agriculture, environmental monitoring, health, logistics, and retail. The progression into the Industrial Internet of Things (IIoT) has further enhanced the capacity for intelligent, real-time processing of vast data volumes, leading to innovations in autonomous vehicles, smart manufacturing, anomaly detection, and predictive maintenance.

The essence of intelligence in embedded technologies lies in the learning algorithms that empower devices to make informed decisions from the data they collect. However, implementing Machine Learning (ML) on tiny devices poses significant challenges due to stringent architectural, power, and latency constraints. These devices operate at milliwatt power levels nevertheless are expected to deliver real-time responses, especially in critical systems such as healthcare monitors, autonomous vehicles, or industrial human-robot collaboration. In these scenarios, any delay in decision-making can result in dire outcomes, including jeopardized patient safety, increased road hazards, or industrial operation interruptions.

Methods. The aim of the article is to serve as a guide for the latest research directions and approaches on TinyML. The objective is to gain a better understanding of the state of the art of TinyML especially in perspective of edge AI. This study seeks the newest trends and most prominent challenges in the field. For the purpose, a search on web-based resources was conducted using relevant keywords. The authors then checked the articles manually excluding any study out of scope.

Problem Statement. Since 2018, the concept of Tiny Machine Learning (TinyML) has emerged with a widely acknowledged definition: TinyML is a paradigm that facilitates running ML on the edge devices with minimal processor and memory requirements; hence, the power consumption of such systems is expected to be within a few milliwatts or less [4]. The hurdles facing TinyML developers are significant, particularly because modern neural networks, among the most advanced technologies today, require billions of parameters [5]. These larger networks yield better performance and broader applicability but at the cost of energy consumption is increased with network size. This escalating demand for energy makes the expansion of neural networks unsustainable at larger scales, underscoring TinyML as not just an attractive but a necessary area of research. Market trends underscore this shift, showing a preference for deploying less energy-intensive hardware and simplifying the complexity of learning algorithms, emphasizing the need for TinyML solutions to become even more efficient.

In developing TinyML solutions, practitioners typically follow one of two traditional approaches — ML-oriented (also known as SW-oriented) or HW-oriented — or a newer strategy known as co-design[6]. Traditional methods keep the design of the ML framework and its hardware implementation separate. In the ML-oriented approach, specialists design, train, and test a model appropriate for the application, optimize its parameters, and deploy it on a chosen device. Conversely, the HW-oriented strategy begins without a predetermined hardware platform, focusing instead on creating optimized hardware using scaled-down models and techniques. The co-design method is innovative because it involves ML experts and hardware engineers collaborating from the very beginning. They share their expertise to design a solution, with hardware engineers delving into the mathematical foundations of ML algorithms to identify efficient hardware components, and ML researchers exploring the latest technologies to possibly reconfigure their algorithms for a seamless hardware-software integration. This mutual shaping of form and function represents the cutting-edge of TinyML development [7].

Data pipeline. The standard procedure for implementing TinyML, as illustrated in a conceptual figure, involves three key stages: training, optimization, and deployment. This process can be split into two main components: conventional ML tasks (such as data collection, algorithm selection, model training, and optimization) and tasks specific to TinyML (like model porting and deployment).

The initial step in TinyML development involves choosing an appropriate algorithm and collecting data, either from pre-existing datasets or real-time sensor data. For example, the Google Speech Commands dataset can be utilized to develop and test a keyword spotting algorithm, but also possible way is to utilize sensor data to train a computer vision algorithm for autonomous micro-cars. Training typically occurs on a high-resource device, such as a server or PC, using popular ML frameworks like TensorFlow, PyTorch, or Scikit-Learn.

After training, the model undergoes optimization through methods such as Pruning, Knowledge Distillation, Quantization, and Encoding to meet specific performance criteria. Pruning removes unnecessary neural network parameters; Knowledge Distillation involves teaching a smaller model to replicate the output of a larger, pre-trained one; and Quantization reduces the numerical precision of the network to make it more efficient, though care must be taken to avoid significant performance drops.

The final stages involve porting the optimized model to a language compatible with MCUs (commonly C/C++), using tools like TensorFlow Lite Micro to translate the model into a format suitable for embedded systems. This step prepares the model for deployment in an MCU, where it can perform inference tasks directly on sensor data, leveraging the machine learning model embedded within [8].

Deployments. The growing popularity of embedded technologies has spurred ongoing research into low-resource technologies, aligning with the objectives of TinyML. A vital aspect of TinyML applications involves embedded inference mechanisms, which have seen a significant amount of research. It is important to highlight that models developed using mainstream ML frameworks like TensorFlow, PyTorch, and Scikit-Learn are typically not directly compatible with TinyML due to their substantial memory demands. To address this, various runtimes have been introduced to facilitate ML/DL implementation on microcontroller units (MCUs). Yet, there are essentially three main strategies identified for deploying TinyML solutions: Manual Programming, Code Generators, and ML Interpreters [9].

Manual Programming is often considered the most effective approach because it allows for detailed customization and optimization. However, this method’s drawback is that optimizations are usually proprietary and not widely shared, limiting knowledge dissemination. Additionally, the diverse nature of MCUs complicates the standardization of Embedded ML techniques. While Manual Programming can enhance performance, it sacrifices ease of replication and requires significant time investment, making it less suitable for individuals outside the field, such as hobbyists. The absence of tools for on-the-fly model updates further positions manual programming as a necessary, albeit challenging, option. On the other hand, Code Generation tools stand out for their convenience, offering a way to achieve optimal solutions without the need for manual coding. This approach significantly simplifies the deployment process, making it more accessible to a broader audience. Code Generation tools offer a convenient and efficient pathway to achieving optimal TinyML solutions without the necessity for manual programming. Tools like EdgeImpulse and Imagimob leverage AutoML can provide comprehensive Software-as-a-Service (SaaS) solutions, while others serve as repositories for locating third-party TinyML libraries. Despite their advantages, a significant limitation of Code Generation approaches is the

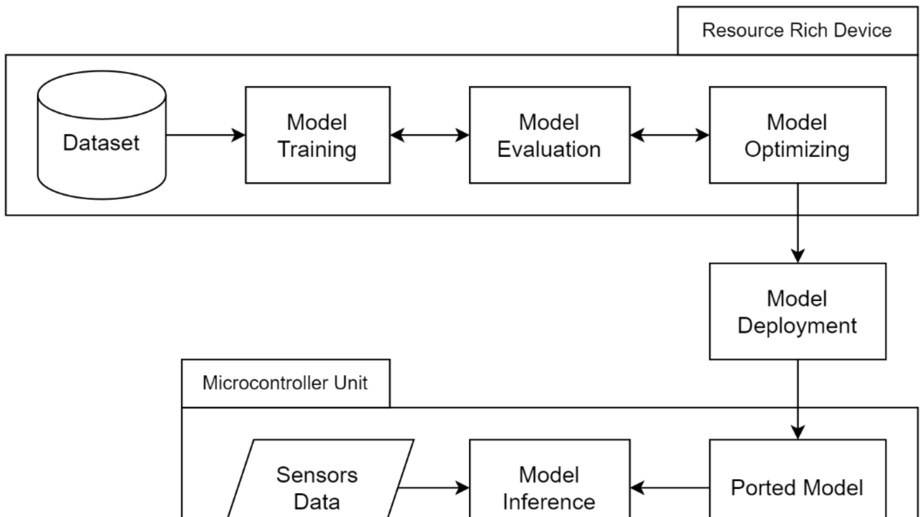

Fig. 1. Generic TinyML Pipeline

fragmented market. Many vendors use proprietary toolsets and compilers, leading to challenges with interoperability and portability across different platforms.

Interpreters present a contrasting advantage by ensuring superior portability, as their architecture remains consistent across various devices. This uniformity allows for the easy porting of ML/DL models without tying the model's architecture to a specific framework. Models can also be individually tailored and optimized to meet the specific requirements of different devices. However, this approach may incur slight performance and memory usage overheads. The distinction between the model and system-level processes under this scheme enhances generalizability and simplifies the introduction of benchmarks, facilitating a broader application of embedded machine learning.

To make embedded machine learning more universally accessible, leading technology companies have initiated open-source projects aimed at standardizing TinyML approaches. Google's TensorFlow Lite Micro (TFLM) and Microsoft's Embedded Learning Library (ELL) are prime examples of such efforts.

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

Speech-Based Applications. Key applications in today's context include speech detection and recognition, online education platforms, and purpose-driven communication, as illustrated in Figure 2. These applications typically demand substantial data processing and power consumption on the host device. To address these challenges, the TinySpeech library has been developed, offering a solution that minimizes computational demands and storage requirements through the use of deep convolutional networks.

In the context of Speech Enhancement, the researchers tackled the challenge of optimizing the size of the speech enhancement model due to limitations in hardware resources. The study utilized structured pruning and integer quantization techniques on a Recurrent Neural Network (RNN) model dedicated to speech enhancement. Their findings indicated a significant reduction in the model's size by approximately 11.9 times and a decrease in the number of operations by about 2.9 times. Additionally, the research highlighted the effectiveness of neural speech enhancement methods in hearing aid devices, notably improving battery performance. The necessity for efficient resource use in energy-restricted edge devices running voice recognition tasks was further emphasized by the findings presented in another study. This research proposed a strategy for splitting the process and suggested a co-design approach tailored for TinyML-based voice recognition systems. By implementing a windowing operation, the researchers managed to segregate hardware and software tasks, enabling preliminary voice data processing [10]. This approach led to reduced energy consumption on the hardware side. Furthermore, the study laid out the groundwork for future research on optimizing the division between hardware and software through co-design, aiming to enhance efficiency even further.

Vision-Based Applications. TinyML is pivotal for processing computer vision datasets, especially when such tasks need to be executed on edge devices for swift outcomes. A study highlighted in [11] tackled the hurdles of training models on the OpenMV H7 microcontroller board. The researchers developed a system capable of identifying American Sign Language alphabets using an ARM Cortex-M7 microcontroller, equipped with just 496 KB of frame-buffer RAM. This effort primarily aimed at overcoming the significant issue of high generalization errors observed in convolutional neural networks (CNNs),

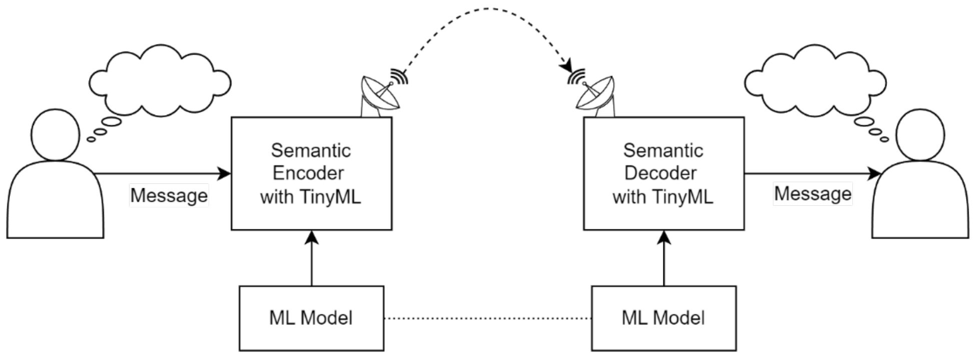

Fig. 2. Voice recognition

which, despite high training and testing accuracy, struggled to effectively adapt to new scenarios and backgrounds with noise interference. The team applied interpolation augmentation to address this, achieving a notable 98.80% accuracy in testing and 74.59% in generalization. This approach notably mitigated the accuracy loss typically associated with quantization in hand sign classification tasks.

In a study detailed in [12], researchers conducted a case study to create a gesture recognition device that could attach to a cane for use by visually impaired individuals. The primary design considerations included affordability, precise gesture recognition, and efficient battery usage. Data for this project was gathered from a gesture dataset, and the ProtoNN model was employed alongside a classification algorithm for training purposes. The study underscored the importance of comprehensively understanding gestures and their safety implications, as well as exploring potential integration with Android and other devices as areas for future investigation.

Data Pattern Classification and Compression. The adaptation of TinyML models to process real-time data has become a focal point for researchers. In [13], a pioneering approach was introduced, named TinyML with Online Learning (TinyOL), which facilitates the incremental online training of models on microcontroller units (MCUs) and allows for the models to be updated directly on IoT edge devices. This system was developed in C++, incorporating an extra layer into TinyOL. The methodology was applied to the auto-encoder on the Arduino Nano 33 BLE Sense board, training the model to recognize new data patterns. The research highlighted the need for creating efficient and optimized neural network algorithms that can accommodate online training on devices. Meanwhile, [14] addressed the challenge posed by the number of activation layers in memory-restricted AI devices on the edge. To tackle this, Tiny-Transfer-Learning (TinyTL) was devised to make better use of memory on edge devices by bypassing the use of intermediate layers for activation.

Health Diagnosis. In response to the COVID-19 pandemic, there's an increased necessity for continuous monitoring of cough-related respiratory symptoms. Researchers introduced a scalable CNN-based model called Tiny RespNet in [15], designed to function in multi-modal environments. This model is implemented on the Xilinx Artix-7 100T FPGA, offering the advantages of parallel processing, low power usage, and high energy efficiency. The Tiny RespNet framework is capable of processing various types of input, including audio recordings and speech from patients, as well as demographic information, facilitating effective classification. It successfully classifies cough detection and related respiratory symptoms across three datasets.

Predictive Maintenance. Of special importance for industry applications is the ability of TinyML solutions to predict when maintenance should be performed for machines of various nature. A study [16] introduces a self-contained low-power on-device PdM (LOPdM) system based on the cutting-edge self-powered sensor (SPS) and tiny machine learning (TinyML) techniques. The piezoelectric sensor measures vibration of simulated equipment. Result data forms input for six different AI models. Two of the models, namely the random forest (RF) and the deep neural network (DNN) are able to identify malfunctions with precision up to 99%.

Study [17] leverages a two-dimensional CNN deployed to STMF767ZI microcontroller by means of X-CUBE-AI tool to predict remaining useful life (RUL) of turbofan engines. Enhanced by L1 norm weight pruning and Adam optimization algorithm retraining, this method still achieves acceptable accuracy comparing to Cloud AI deployment.

Brain-Computer Interface. In the healthcare industry, TinyML offers immense potential, notably in areas such as tumor and cancer detection, emotional intelligence, and predicting health conditions using EEG and ECG signals [18]. TinyML technologies empower Adaptive Deep Brain Stimulation (aDBS) systems [19], which are on the brink of achieving significant advancements in clinical applications. aDBS is crucial for pinpointing specific biomarkers and symptoms related to diseases by directly recording brain signals. Given that healthcare often involves gathering vast amounts of data and processing this information to devise timely interventions for patients, creating a system that is both highly accurate and secure is essential. This integration of IoT with TinyML within the realm of medical science gives rise to what is known as the Healthcare Internet of Things (H-IoT) [20]. H-IoT's primary uses include monitoring, diagnostics, controlling the spread of diseases, logistics, and support systems. For remote health status monitoring of patients, it's critical to develop a system that is not only reliable but also features minimal latency and is globally accessible. Such a system can be realized by combining H-IoT with TinyML and leveraging 6G-enabled internet service [21].

Autonomous Vehicles. Autonomous vehicles play a pivotal role in various emergency scenarios, including military operations, human tracking, and industrial uses. These vehicles require advanced navigation capabilities for the effective detection and tracking of targeted objects. The challenge of enabling autonomous navigation becomes particularly pronounced with smaller-scale vehicles. TinyML technology has been applied to enhance the autonomous operation of such mini-vehicles, as demonstrated in a study using the GAP8 MCI [22], which incorporates a convolutional neural network

(CNN) framework. This technology was also tested on the STM32L4 and NXP k64f platforms, revealing that TinyML integration can decrease processing delays by up to 13 times while achieving an energy efficiency improvement of about 90%.

Furthermore, the exploration of automatic traffic scheduling has gained attention as a potential area for TinyML application to enhance real-time traffic management systems [23]. One innovative approach involves using piezoelectric sensors embedded across various lanes on a road, employing a two-point time ratio technique to detect vehicles through piezo-sensor data. This method extends to vehicle classification and the prediction of green light timing, using a random forest regressor to determine signal duration based on the vehicle count in each lane. This system was implemented on an Arduino Uno, utilizing the m2gen library compatible with Scikit Learn.

Phenomics and conservation of ecology. Phenomics is the examination of how an organism's phenotypes evolve in response to genetic variations throughout its life. In the realm of plant phenomics, this field is particularly focused on identifying significant germplasm by leveraging genetic advancements across various plant species. A study referenced in [24] explored phenomics through image analysis, specifically targeting the classification of tomato leaf diseases and spider mite infestations. This research utilized the Plant-Village tomato dataset alongside the YOLO3 algorithm, powered by the DarkNet-53 architecture, for the automated detection of tomato leaves. Additionally, the study applied the SegNet algorithm for pixel-wise image segmentation. It also reviewed various data analysis tools to assess their compatibility with TinyML for phenomics research.

In the sphere of ecological conservation, AI-driven analytics have seen substantial advancement. A notable study [25] implemented TinyML in small payload satellites (SmallSats) to enhance the conservation efforts for sea turtles through real-time, vision-based TinyML techniques. TinyML's application extends to environmental monitoring as well; for instance, [26] detailed a compact deep neural network designed for weather prediction. This system, built on the STM32 microcontroller unit (MCU) and employing the X-CUBE-AI toolchain within the Miosix operating system, requires 45.5 KB of flash memory and 480 Bytes of onboard RAM to function.

Anomaly detection. An anomaly refers to an occurrence that deviates from the norm or the bulk of events. The study cited in [27] explores the suitability of TinyML for identifying anomalies in specific tasks. Employing a conventional Artificial Neural Network (ANN), along with an auto-encoder and a variational auto-encoder, the research utilizes the Arduino Nano 33 BLE Sense module and a Kenmore top load washing machine model to pinpoint anomalies during the unbalanced spin dry cycle. The findings indicate a high level of precision and accuracy, reaching around 90%.

In [28] low-cost thermal infrared sensor with 32×24 pixels is used to detect anomalies on heat maps of different machines. CNN running on an ESP-WROOM-32 MCU device was able to reach accuracy as high as 99.73%.

Challenges. TinyML faces significant challenges that impede its growth and development. To summarize them and evaluate their importance, most cited review articles and surveys on the topic were identified in Google Scholar database [1; 3; 8; 29; 30; 31; 32; 33; 34]. Further on, most cited review papers from 2024 [6; 7; 35; 36; 37] were added to include the latest insights as well. Only studies in the English language were considered.

Most prominent problems and challenges are presented and discussed below ranked from most to least mentioned. Some of them are renamed for unification.

1. Limited Memory Capacity.

Most edge devices feature no more than a few hundred KB of RAM memory, and several MB of flash (permanent) storage. The volume is consumed by ML model as well as by the sensor measurements gathered through time. The measurements might be useful to overcome possible concept drift: evolution of the environment that makes input data dissimilar to those used in training set. As a result, the performance of ML model degrades. One possible solution is to leverage online, on-device learning [13] but this will consume more energy as well as processing capacity. Another solution is to store data on a neighbor edge server and access them on-demand, but this can increase network overhead which consumes energy as well [34].

2. Energy Efficiency.

TinyML accommodates the energy efficiency limitation in its paradigm. Minimizing energy consumption is crucial for battery-powered or solar-powered edge devices, which have strict energy quota. Energy is consumed by MCU, by network modules, by sensors, etc. Ideally, each of the components must be flexible enough to lower power consumption according to current workload and operating context. That is, MCU must dynamically switch to lower frequency when performing non-critical tasks, network module must adapt network protocol and transmission power according to the distance to the furthest receiver, sensors must adjust the sampling frequency according to the measurements change over time. These are future directions for inquiry. Today an MCU can be put to deep sleep till the next awake signal either from sensor or another peripheral module. Otherwise, it normally operates at maximum frequency. The same principle is for other components: most of them can be turned on or off, can work on pre-defined settings but are unable to adapt automatically. Another common approach for the problem is task offloading or federated execution [38].

3. Processor Power.

Most edge devices operate within a clock speed range of 10–1000 MHz. This range can be limiting for the execution of complex learning models directly on edge devices, affecting the performance and responsiveness of TinyML applications. To overcome these limitations, alternative architectures are researched such as PULP (parallel ultra- low power) [39]. It is demonstrated that PULP test implementation is 12.87x faster in comparison to ARM Cortex-M4. Nevertheless, in many practical cases such speed is extensive. Clear tendency to use low-power off-the-shelf devices such as Arduino Nano 33 BLE (ARM Cortex M4, 64 MHz) is shown in [6].

4. Device Heterogeneity.

This problem refers to the variability in the characteristics of different edge devices used for ML. It's caused by several reasons: different manufacturers adhere to different architectures and even more architectures and frameworks are proposed every year in new studies. Furthermore, Hardware Heterogeneity usually causes Software Heterogeneity as device drivers and operating systems are incompatible in most cases. Another side of it is the existence of multiple ML frameworks and inference engines which generate incompatible ML models, i.e. cause Models Heterogeneity. As of now, there are no realistic solutions proposed for the problem.

5. Benchmarking.

Hardware and Software Heterogeneity causes natural need to compare device performance when it comes to ML inference. And memory and processor limitations restrict usage of existing ML benchmarks which target more powerful computers. There were several benchmarks created to fill the gap: MLPerf Tiny Benchmark[6], "TinyML benchmark"[40] and BiomedBench[41]. Still, together they cover only a small fraction of devices, datasets and ML models used today. Also, they don't include more advanced algorithms such as transfer learning or online learning.

6. Lack of Datasets and Models.

There is large number of datasets mentioned in the papers. But only some of them are made public. And only part of those demonstrate good balance, diversity and are properly annotated. Addressing the lack of datasets problem in TinyML involves strategies such as data augmentation, synthetic data generation, transfer learning, and few-shot learning techniques.

Lack of Accepted Models problem refers to the absence of widely accepted public ML models. And it's closely related to Lack of Datasets and Device Heterogeneity problems. Which means that such a model must be trained on a good-quality public dataset and must be deployable to most popular devices without considerable loss of model metrics. One of the explanations of the problem presence is that the entire TinyML field is still too young and immature to construct proper tools and implement the solution.

Conclusions. TinyML emerged as a response to the growing need for bringing machine learning capabilities to resource-constrained devices, such as those used in IoT applications. It addresses the challenge of implementing AI in environments where computing power, memory, and energy availability are limited. By optimizing ML models to run efficiently on small, low-power devices, TinyML enables intelligent data processing at the edge, reducing the need for constant cloud connectivity and thus enhancing privacy, response times, and operational efficiency.

Still there are serious challenges to consider before deploying ML models to edge devices. This leaves multiple opportunities for further research directions in the field.

REFERENCES

1. Dutta, L., & Bharali, S. (2021). TinyML meets IoT: A comprehensive survey. Internet of Things, 16, 100461.
2. Fraga-Lamas, P., Lopes, S. I., & Fernández-Caramés, T. M. (2021). Green IoT and edge AI as key technological enablers for a sustainable digital transition towards a smart circular economy: An industry 5.0 use case. Sensors, 21(17), 5745.
3. Schizas, N., Karras, A., Karras, C., & Sioutas, S. (2022). TinyML for ultra-low power AI and large scale IoT deployments: a systematic review. Future Internet, 14(12), 363.
4. Warden, P., & Situnayake, D. (2019). TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers. O'Reilly Media.
5. Pramod, A., Naicker, H. S., & Tyagi, A. K. (2021). Machine learning and deep learning: Open issues and future research directions for the next 10 years. Computational analysis and deep learning for medical care: Principles, methods, and applications, 463–490.
6. Capogrosso, L., Cunico, F., Cheng, D. S., Fummi, F., & Cristani, M. (2024). A machine learning-oriented survey on tiny machine learning. IEEE Access.
7. Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S., & López, O. L. (2024). TinyML: Tools, applications, challenges, and future research directions. Multimedia Tools and Applications, 83(10), 29015-29045.
8. Rajapakse, V., Karunanayake, I., & Ahmed, N. (2023). Intelligence at the extreme edge: A survey on reformable TinyML. ACM Computing Surveys, 55(13s), 1-30.

9. Banbury, C. R., Reddi, V. J., Lam, M., Fu, W., Fazel, A., Holleman, J., ... & Yadav, P. (2020). Benchmarking TinyML systems: Challenges and direction. arXiv preprint arXiv:2003.04821.
10. Kwon, J., & Park, D. (2021). Hardware/software co-design for TinyML voice-recognition application on resource frugal Edge Devices. Applied Sciences, 11(22), 11073.
11. Paul, A. J., Mohan, P., & Sehgal, S. (2020, December). Rethinking generalization in american sign language prediction for edge devices with extremely low memory footprint. In 2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS) (pp. 147–152). IEEE.
12. Patil, S. G., Dennis, D. K., Pabbaraju, C., Shaheer, N., Simhadri, H. V., Seshadri, V., ... & Jain, P. (2019, October). Gesturepod: Enabling on-device gesture-based interaction for white cane users. In Proceedings of the 32nd Annual ACM symposium on User Interface Software and technology (pp. 403–415).
13. Ren, H., Anicic, D., & Runkler, T. A. (2021, July). TinyOL: TinyML with online-learning on microcontrollers. In 2021 international joint conference on neural networks (IJCNN) (pp. 1–8). IEEE.
14. Cai, H., Gan, C., Zhu, L., & Han, S. (2020). TinyTL: Reduce activations, not trainable parameters for efficient on-device learning. arXiv preprint arXiv:2007.11622.
15. Rashid, H. A., Ren, H., Mazumder, A. N., & Mohsenin, T. (2021). Tiny RespNet: a scalable multimodal tinyCNN processor for automatic detection of respiratory symptoms. In TinyML Research Symposium (pp. 1–8).
16. Chen, Z., Gao, Y., & Liang, J. (2023). LOPdM: A Low-power On-device Predictive Maintenance System Based on Self-powered Sensing and TinyML. IEEE Transactions on Instrumentation and Measurement.
17. Athanasakis, G., Filios, G., Katsidimas, I., Nikoletseas, S., & Panagiotou, S. H. (2022, September). TinyML-based approach for remaining useful life Prediction of Turbofan Engines. In 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1-8). IEEE.
18. Pai, K., Kallimani, R., Iyer, S., Maheswari, B. U., Khanai, R., & Torse, D. (2022). A Survey on Brain-Computer Interface and Related Applications. arXiv preprint arXiv:2203.09164.
19. Merk, T., Peterson, V., Köhler, R., Haufe, S., Richardson, R. M., & Neumann, W. J. (2022). Machine learning based brain signal decoding for intelligent adaptive deep brain stimulation. Experimental Neurology, 351, 113993.
20. Bharadwaj, H. K., Agarwal, A., Chamola, V., Lakkaniga, N. R., Hassija, V., Guizani, M., & Sikdar, B. (2021). A review on the role of machine learning in enabling IoT based healthcare applications. IEEE Access, 9, 38859-38890.
21. Padhi, P. K., & Charrua-Santos, F. (2021). 6G enabled tactile internet and cognitive internet of healthcare everything: Towards a theoretical framework. Applied System Innovation, 4(3), 66.
22. de Prado, M., Rusci, M., Capotondi, A., Donze, R., Benini, L., & Pazos, N. (2021). Robustifying the deployment of TinyML models for autonomous mini-vehicles. Sensors, 21(4), 1339.
23. Roshan, A. N., Gokulapriyan, B., Siddarth, C., & Kokil, P. (2021, March). Adaptive traffic control with TinyML. In 2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET) (pp. 451-455). IEEE.
24. Nakhle, F., & Harfouche, A. L. (2021). Ready, Steady, Go AI: A practical tutorial on fundamentals of artificial intelligence and its applications in phenomics image analysis. Patterns, 2(9).
25. Curnick, D. J., Davies, A. J., Duncan, C., Freeman, R., Jacoby, D. M., Shelley, H. T., ... & Pettorelli, N. (2022). SmallSats: a new technological frontier in ecology and conservation?. Remote Sensing in Ecology and Conservation, 8(2), 139–150.
26. Alongi, F., Ghielmetti, N., Pau, D., Terraneo, F., & Fornaciari, W. (2020, September). Tiny neural networks for environmental predictions: An integrated approach with miosix. In 2020 IEEE International Conference on Smart Computing (SMARTCOMP) (pp. 350-355). IEEE.
27. Lord, M., & Kaplan, A. (2021, December). Mechanical anomaly detection on an embedded microcontroller. In 2021 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 562–568). IEEE.
28. Oliveira, V. M., & Moreira, A. H. (2021). Edge AI System Using a Thermal Camera for Industrial Anomaly Detection. In International Summit Smart City 360° (pp. 172-187). Cham: Springer International Publishing.
29. Sanchez-Iborra, R., & Skarmeta, A. F. (2020). Tinyml-enabled frugal smart objects: Challenges and opportunities. IEEE Circuits and Systems Magazine, 20(3), 4-18.
30. Ray, P. P. (2022). A review on TinyML: State-of-the-art and prospects. Journal of King Saud University-Computer and Information Sciences, 34(4), 1595-1623.
31. Saha, S. S., Sandha, S. S., & Srivastava, M. (2022). Machine learning for microcontroller-class hardware: A review. IEEE Sensors Journal, 22(22), 21362-21390.

32. Su, W., Li, L., Liu, F., He, M., & Liang, X. (2022). AI on the edge: a comprehensive review. Artificial Intelligence Review, 55(8), 6125-6183.
33. Alajlan, N. N., & Ibrahim, D. M. (2022). TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications. Micromachines, 13(6), 851.
34. Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. (2023). A comprehensive survey on tinyml. IEEE Access.
35. Tsoukas, V., Gkogkidis, A., Boumpa, E., & Kakarountas, A. (2024). A Review on the emerging technology of TinyML. ACM Computing Surveys.
36. Jouini, O., Sethom, K., Namoun, A., Aljohani, N., Alanazi, M. H., & Alanazi, M. N. (2024). A Survey of Machine Learning in Edge Computing: Techniques, Frameworks, Applications, Issues, and Research Directions. Technologies, 12(6), 81.
37. Laskaridis, S., Venieris, S. I., Kouris, A., Li, R., & Lane, N. D. (2024). The future of consumer edge-ai computing. IEEE Pervasive Computing.
38. Zhou, H., Jiang, K., Liu, X., Li, X., & Leung, V. C. (2021). Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. IEEE Internet of Things Journal, 9(2), 1517-1530.
39. Tabanelli, E., Tagliavini, G., & Benini, L. (2023). DNN is not all you need: Parallelizing non-neural ML algorithms on ultra-low-power IoT processors. ACM Transactions on Embedded Computing Systems, 22(3), 1-33.
40. Sudharsan, B., Salerno, S., Nguyen, D. D., Yahya, M., Wahid, A., Yadav, P., ... & Ali, M. I. (2021, June). Tinyml benchmark: Executing fully connected neural networks on commodity microcontrollers. In 2021 IEEE 7th World Forum on Internet of Things (WF-IoT) (pp. 883-884). IEEE.
41. Samakovlis, D., Albini, S., Álvarez, R. R., Constantinescu, D. A., Schiavone, P. D., Peón-Quirós, M., & Atienza, D. (2024). BiomedBench: A benchmark suite of TinyML biomedical applications for low-power wearables. IEEE Design & Test.