Internet of Things 25 (2024) 101063

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Review article

Anomaly detection based on Artificial Intelligence of Things: A Systematic Literature Mapping

Sergio Trilles [a,b,*], Sahibzada Saadoon Hammad [a,c], Ditsuhi Iskandaryan [d,e]

[a] *Institute of New Imaging Technologies, Universitat Jaume I, Avda. Sos Baynat, s/n, 12071, Castelló de la Plana, Spain*
[b] *Facultad de Ciencias y Tecnología, Universidad Isabel I, Calle Fernán González, 76, 09003, Burgos, Spain*
[c] *ValgrAI - Valencian Graduate School and Research Network of Artificial Intelligence, Camí de Vera s/n, 46022, Valencia, Spain*
[d] *Oncology Data Analytics Program, Catalan Institute of Oncology (ICO), Avinguda de la Gran via de l'Hospitalet, 199, 08908, Barcelona, Spain*
[e] *Colorectal Cancer Group, ONCOBELL Program, Institut de Recerca Biomedica de Bellvitge (IDIBELL), Avinguda de la Gran via de l'Hospitalet, 199, 08908, Barcelona, Spain*

ARTICLE INFO

Dataset link: https://doi.org/10.5281/zenodo.8231242

Keywords:
Anomaly detection
Edge computing
Artificial Intelligence of Things
TinyML
MicroController Units

ABSTRACT

Advanced Machine Learning (ML) algorithms can be applied using Edge Computing (EC) to detect anomalies, which is the basis of Artificial Intelligence of Things (AIoT). EC has emerged as a solution for processing and analysing information on IoT devices. This field aims to allow the implementation of Machine/Deep Learning (DL) models on MicroController Units (MCUs). Integrating anomaly detection analysis on Internet of Things (IoT) devices produces clear benefits as it ensures the use of accurate data from the initial stage. However, this process poses a challenge due to the unique characteristics of IoT. This article presents a Systematic Literature Mapping of scientific research on the application of anomaly detection techniques in EC using MCUs. A total of 18 papers published over the period 2021–2023 were selected from a total of 162 in four databases of scientific papers. The results of this paper provide a comprehensive overview of anomaly detection using TinyML and MCUs. The main contributions of this survey are the fact that it aims to: (a) study techniques for anomaly detection in ML/DL and validation metrics used in the AIoT; (b) analyse data used in the estimation of models; (c) show how ML is applied in EC using hardware or software; (d) investigate the main microcontrollers, types of power supply, and communication technology; and (e) develop a taxonomy of ML/DL algorithms used to detect anomalies in TinyML. Finally, the benefits and challenges of this kind of TinyML analysis are described.

1. Introduction

In today's world, the Internet of Things (IoT) plays a significant role in our daily lives, connecting different aspects of our environment [1]. Edge Computing (EC) is a type of computing used in IoT systems, which enables more efficient devices to process data and improve performance near the source [2,3]. EC has several benefits, including faster data processing, lower transfer costs, improved information security and privacy, and increased productivity [2,4]. Sensor-based IoT devices generate valuable data that can be used to better understand and manage our surroundings, leading to better decision-making, automation and productivity [5,6]. However, low-cost data-generating devices can introduce errors, leading to unexpected system behaviour [7].

* Corresponding author at: Institute of New Imaging Technologies, Universitat Jaume I, Avda. Sos Baynat, s/n, 12071, Castelló de la Plana, Spain.
E-mail addresses: strilles@uji.es (S. Trilles), hammad@uji.es (S.S. Hammad), iskandar@uji.es (D. Iskandaryan).

https://doi.org/10.1016/j.iot.2024.101063
Received 9 October 2023; Received in revised form 2 January 2024; Accepted 3 January 2024
Available online 11 January 2024

Detecting anomalies or unexpected states in the analysis of IoT data is crucial in domains such as automated operations in industries, energy management, healthcare or assisted living [8,9]. Anomaly detection can include the identification of abnormalities, outliers or specific events. Several techniques are used for anomaly detection in order to identify incorrect data. However, these techniques require significant human involvement to activate the systems and comprehend the data produced. Even with minor procedures, manually detecting trends can be challenging. It is easier for experts to focus on a limited subset of data and manually recognise relevant patterns and trends. However, as the number of connected devices increases, the complexity of data analysis also increases. Therefore, there is growing interest in developing automated methods that allow professionals to concentrate solely on the most significant events [10].

Although the IoT realm may be considered relatively new, multiple methods are already available for detecting anomalies in both time-series and non-temporal data from various domains that can be applied in IoT-specific scenarios. Several surveys have explored this topic, including the detection of anomalies in time-series data. The early works of Hawkins [11] and Abraham and Chang [12] laid the foundation for more comprehensive investigations, such as those conducted by Markou and Singh [13,14], who led a two-part survey exploring statistical and Neural Network (NN) approaches. Further methods available in 2009 were analysed by Chandola et al. [15], while Zhang et al. [16] discussed approaches that applied to early IoT. More recently, Chalapathy and Chawla [17] studied Deep Learning (DL) methods in the broader field of anomaly detection, including time series in IoT scenarios.

Using data-driven analytical procedures efficiently is crucial for IoT applications, since various techniques are used to detect anomalies in IoT environments, as mentioned in a survey conducted in [18]. Intelligent methods such as Big Data or Artificial Intelligence (AI) are used to gain knowledge from these data. Machine Learning (ML), a subset of AI, enables machines or IoT devices to learn from previous data without explicit programming. ML is critical in building intelligent IoT applications, as it uncovers hidden patterns by analysing large volumes of data using sophisticated algorithms, according to [19]. Smart IoT devices are imperative for optimising and allocating resources, performing real-time data analysis and making decisions. These techniques fall under three broad categories: statistical, ML and DL. Statistical methods include Dissimilarity Measures, Bayesian, Principal Component Analysis (PCA), Autoregressive Integrated Moving Average (ARIMA) and Correlation-based methods. ML techniques include Support Vector Machine (SVM), Regression models, Reinforcement Learning (RL), Decision Trees, NN and others. The DL methods identified are Convolutional NN (CNN) and Autoencoder models.

In recent years, IoT devices have become more powerful and capable of performing complex computational tasks [20]. This advancement has opened up opportunities to run advanced algorithms on the network edge [21]. Conducting data analysis and ML operations at the network edge helps prevent delays when transferring data from the source to the cloud, resulting in faster computation times [22]. Combining AI capabilities with IoT devices has led to the concept of Artificial Intelligence of Things (AIoT) [23]. In a conventional IoT architecture, ML-related tasks are typically performed on the cloud side, which can result in high latency and less-than-desirable Quality of Service (QoS) [24]. IoT devices collect the data and then send them to the cloud layer for analysis, with ML algorithms applied. This limits the real-time analytics capabilities of the system. Additionally, there are also privacy, security and bandwidth challenges [25–27]. TinyML addresses these issues by applying ML algorithms on low-power devices such as MicroController Units (MCUs) with limited memory and processing resources. In recent times, TinyML has emerged as a new trend that involves integrating ML techniques into small, low-power devices powered by MCUs [28]. The significance of TinyML lies in its ability to use powerful algorithms in battery-powered devices, thereby enabling real-time data processing and improving responsiveness while reducing the energy consumption required for wireless communications. As a result, this technology is expected to undergo exponential growth in the coming years and gain widespread acceptance [29].

Different reviews have been conducted on anomaly detection techniques. The first study was [30], where authors analysed anomaly detection work using different techniques such as statistics, classification, clustering, or information theoretic. Another review [31] focused on anomaly detection techniques using DL. In the same study, a hierarchical taxonomy was proposed, classifying deep anomaly detection methods into three main categories and 11 detailed categories based on the modelling perspective. With the emergence of the IoT realm, more focused revisions have appeared to the data produced by these devices. For instance, [32] analysed algorithms and approaches to detect anomalies in time series data generated by IoT devices. Lastly, [33] presented a summary of detection methods and applications, discussing the categorisation of IoT anomaly detection algorithms.

This paper presents a comprehensive Systematic Literature Mapping (SLM) on ML and DL in AIoT, specifically focusing on their ability to detect anomalies. The review follows the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) methodology [34]. Unlike previous works, this review analyses the use of anomaly detection techniques in the AIoT paradigm, which involves applying these models in embedded IoT devices (MCUs) and using edge computing. Additionally, the review was conducted methodically following the PRISMA guidelines, which ensure a systematic and replicable approach for reporting systematic reviews and meta-analyses [35]. The survey covers the following key contributions: (a) studying techniques for anomaly detection in ML/DL and validation metrics used in the AIoT; (b) analysing data used in the estimation of models; (c) showing how ML is applied to EC using hardware or software; (d) investigating the main microcontrollers, types of power supply and communication technology; and (e) developing a taxonomy of ML/DL algorithms used for detection anomalies based on TinyML. Finally, the benefits and challenges of this kind of TinyML analysis are described.

The paper is organised as follows. In Section 2, we introduce the main concepts to provide a context for the review. Section 3 explains the methodology used to achieve this systematic review of the literature. Next, Section 4 presents the results of our analysis. In Section 5, we discuss the benefits and challenges of using TinyML in the detection of anomalies. Section 6 defines a roadmap for anomaly detection in MCU. Finally, we offer our conclusions in Section 7.

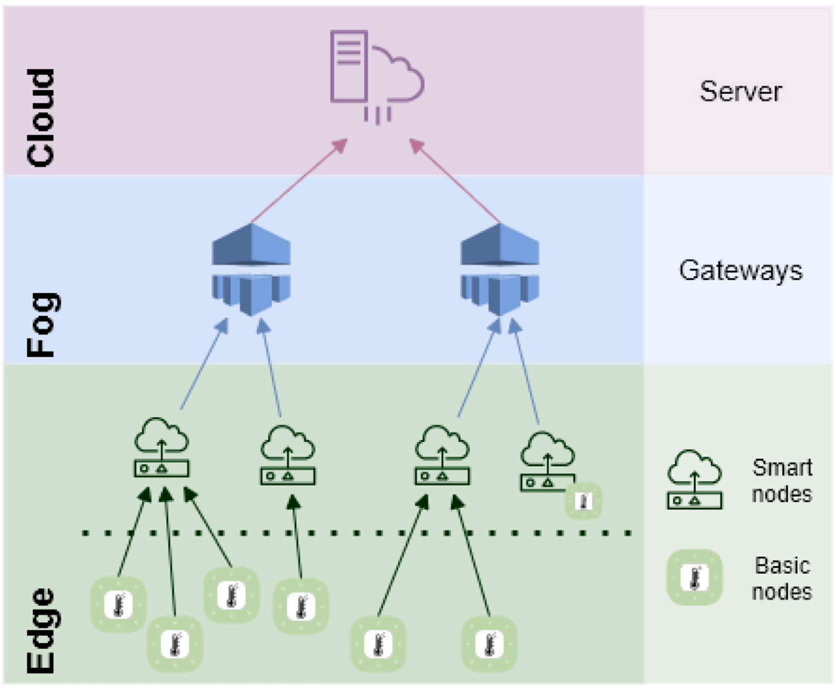

Fig. 1. Edge computing architecture defined in [39].

2. Background and concepts

This section serves as an introductory overview, elucidating key concepts essential for comprehending the intricacies of the present study.

2.1. Internet of Things and edge computing

In 1999, Kevin Ashton introduced the concept of IoT. IoT is a network of intelligent objects that allows the sharing of information and resources, while adapting to changes in the environment and being self-organising [36]. With IoT, smart devices can connect to the Internet and serve various domains, discussed in [37] and analysed in [38]. These applications are classified into three categories: smart city, industry and health. In a typical IoT architecture (as presented in Fig. 1), IoT devices with limited computing capabilities are located at the lower level (EC). This layer comprises basic sensors/actuators and smart devices [1]. Basic sensors and actuators lack computing or Internet Protocol (IP) connectivity capabilities, requiring higher-level devices to establish links. Smart devices can store and process measurements and use their computing power (EC). Distributed devices with greater capacity than smart devices are placed in the intermediate layer, or Fog Computing (FC). These Fog devices maintain communication and data flow captured from IoT devices and temporarily store data in a local cloud. The data are processed and analytical processes can be performed in real time. Traditional cloud services store data and manage all devices in the upper layer.

When computing is carried out at the edge of the network to manage downstream data for cloud services and upstream data for IoT services, it is known as EC [2]. The term "Edge" refers to network resources that are located close to users or data sources [2,40]. EC addresses the issue of high network latency in applications and services that are sensitive to lags in Cloud Computing (CC) [41]. EC boosts data security by performing computer downloads, data storage and information processing close to the user and in near real time. AI applications are increasing in IoT environments due to EC, where semantic web, ML/DL, and data mining technologies are commonly used to leverage the computational power of EC [42].

2.2. Machine learning and deep learning

Experts in the field note that ML involves the study of computational algorithms that gain experience by making decisions and predictions [43,44]. By analysing and processing data, intelligent systems can be created [45] that adapt to changing circumstances [46]. These algorithms use mathematical concepts such as statistics, optimisation and probability to operate effectively [43]. ML algorithms can be categorised into three groups: (1) supervised learning algorithms that predict the label of an object based on its characteristics and predefined class labels; (2) unsupervised learning algorithms that generate classification labels without predefined classes; and (3) RL algorithms that learn from real-world events and receive rewards or penalties based on their decisions [46–48].

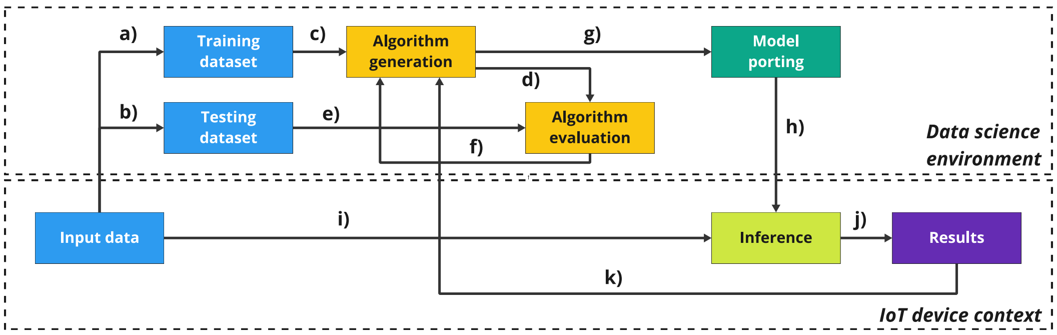

Fig. 2. TinyML workflow.

DL algorithms are a form of ML that aims to model complex patterns in data prediction using multilayer information processing modules [49,50]. The term "deep" refers to the numerous hidden layers that make up the NN [49]. The rise of DL can be attributed to advancements in ML algorithms, increased processing power, affordable hardware and the availability of large datasets [49–51]. Like ML algorithms, DL can be classified as supervised, unsupervised or RL algorithms [52].

According to research conducted by [50,53,54], there are four main categories of DL algorithms. The first category is CNN, primarily used for computer vision and image recognition. It comprises convolution and subsampling layers (pooling) and uses the convolution operation to learn higher-order characteristics. The second category is Recurrent NN (RNN), commonly used for time-series problems. RNNs use sequential information to learn patterns in the data and do not have a defined layer structure, allowing arbitrary connections between neurons. The third category is Autoencoder algorithms, which encode a dataset, reducing its dimensionality while simultaneously learning how to reconstruct it. Finally, there is the Restricted Boltzmann Machine (RBM), a stochastic and generative NN capable of learning representations and solving combinatorial problems.

AI models are generated through training and testing steps. This involves exposing them to vast amounts of data, including images and values. Through repetition and the analysis of massive databases, AI learns the characteristics of these elements and generates a model. This model is typically uploaded to a cloud server for a production stage. However, EC improves this process by generating a copy of the ML model in each IoT device where data is generated. This means that the information does not have to travel to the cloud server for each model iteration. This approach improves the speed of detecting manufacturing faults and significantly reduces traffic and bandwidth requirements.

Although EC allows for ML/DL models on devices, powerful algorithms such as Deep NN (DNNs) require significant computing power. In such cases, a technique is employed where some calculations are offloaded to devices with higher processing capabilities [55]. IoT devices transmit data to the closest server, typically located in the fog, which receives the results after performing the model [56]. A common approach to minimising data redundancy on devices is to send essential data. For example, in image analysis models with substantial data volume, IoT devices filter important frames and exclusively transmit them to the server. For more complex solutions, hierarchical structures with different computing capabilities at various levels of devices are also used. This architecture operates synchronously and divides tasks to cover DNN hyperparameters.

The deployment process for a TinyML application is depicted in Fig. 2. The design can be separated into two parts: the conventional data science environment and the IoT context (TinyML). The first part contains the standard steps for traditional ML/DL algorithms, such as data collection, algorithm selection, model building, and optimisation. The second part, which is focused on TinyML, includes the model deployment and porting stages. To create a TinyML model, it is necessary to choose an effective algorithm and gather the required data. This can be achieved by using precompiled datasets or collecting data from real-time sensors. As usual, the input data is split into two subsets: training (a) and test (b). The model generation process (c) is iterative, as the model needs to be regenerated (f) based on the evaluation (d) of the metrics (using the test dataset [e]). Once all the requirements have been met, the model learning process can begin on an IoT device. The final stages of a TinyML system are porting and deployment (g,h). When integrating a model, it must be converted into an MCU-friendly language. TinyML interpreters, such as Tensorflow Lite Micro, are used to convert a model created in a suitable scripting language to a frozen graph that is optimally represented as a C array. Finally, the ported model is incorporated into a microcontroller, where inference (j) is performed on sensor data (i) using the base ML process. The results obtained can be useful for further improving the model (k).

2.3. Anomaly detection

Several attempts have been made to describe anomalous data. According to Hawkins, an outlier is an observation that differs so much from other observations that it raises the suspicion that it was generated by a different mechanism [11]. Alternatively, Barnett and Lewis defined an outlier as an observation (or a subset of observations) that seems inconsistent with the rest of the dataset [57]. An anomaly in the context of the IoT refers to the measurable effects of an unexpected change in the state of a system that deviates from its usual behaviour, either locally or globally. This definition highlights some crucial points about the nature of IoT data. First, most of the information obtained from an IoT system is "normal" and represents the typical characteristics of that particular system. Secondly, the definition recognises that the notion of normal system operations can evolve over time due to various factors. They can be categorised into three types (Fig. 3):

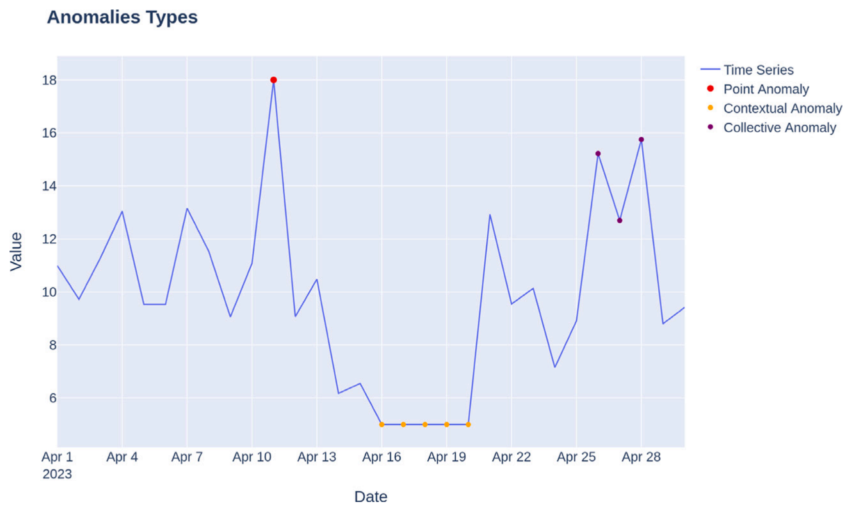

Fig. 3. Representation of the three existing types of anomalies.

- Point anomalies: the occurrence of an observation in the data that does not conform to the normal pattern of the data is a point anomaly. Point anomalies are individual in nature.
- Contextual anomalies: these refer to instances in the data that deviate from normal observations of the data in the context in which the data are being collected. Such instances could be normal values in one context, but in another they might be anomalous; hence, they are context-related.
- Collective or pattern anomalies: these are defined as a collection of data observations that do not adhere to the normal pattern of the data. As the name suggests, such an anomaly type is collective in nature, and an individual point in such anomalies might not be an anomaly.

Various techniques and algorithms have been developed to detect anomalies in time series data [58]. While some methods may combine elements from different approaches, they can generally be classified into the following categories. The choice of technique is highly dependent on various factors present in the monitored data and the environment in which the anomaly detector will be used. Anomaly detection methods fall into several categories. Statistical and probabilistic methods construct a model from historical data, identifying anomalies when new observations deviate from this model [14]. Pattern matching creates a direct model of the time series, comparing new observations to a database of labelled anomalies in supervised settings [59]. Distance-based methods use specific metrics to compare previous and new observations, flagging deviations as anomalies [60]. Clustering projects data into multidimensional space, considering data points close to dense clusters as normal and flagging others as anomalous [61]. Predictive models generate regression models based on recent trends, considering a data point anomalous if it significantly differs from predicted values [62]. Ensemble methods use multiple algorithms and a voting mechanism for anomaly detection, enhancing accuracy but increasing complexity and computational time [63].

3. Systematic literature mapping methodology

When conducting a SLM, a strict methodology is followed to obtain reliable data that can be used to address research issues. This ensures that the review is quantifiable, reproducible, systematic, clearly stated and comprehensive with respect to a specific topic [64]. We conducted a systematic review of the literature using the PRISMA methodology [34], which includes a detailed checklist that other researchers can use to replicate the review process and collect precise data for future research. Although PRISMA is commonly used in the medical and physical sciences [65], it is still in the experimental phase of computer science research. However, using a PRISMA-based approach in an SLM ensures the quality of the review, allows the reader to evaluate the strengths and weaknesses of the research and provides a reference point for other professionals in the field. It also makes it easier to replicate the review methods, structure and format using PRISMA. Overall, the PRISMA methodology ensures the accuracy and reliability of SLMs. We used the PRISMA technique to assess the existing evidence. Our aim was to examine the possibility of implementing anomaly detection approaches on limited embedded devices (MCUs) located at the edge of the network. The process of conducting this review of the literature is depicted in Fig. 4.

Defining the research questions is a vital aspect of any research. It helps to address specific issues and to ensure that biases are avoided by being concise and transparent. As the researchers stated in [66], defining the research issue is the most critical part of a systematic mapping study. The process of formulating research questions aims to identify the primary studies that will be analysed and determine the data that will be extracted to address these questions. Lastly, the development of the mapping involves the implementation of the protocol.

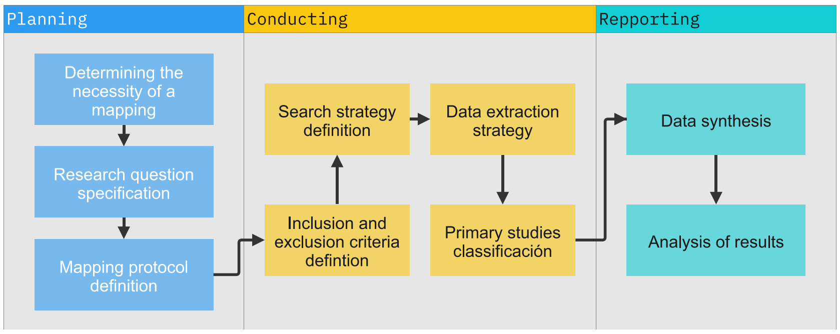
Planning
Conducting
Repporting
Determining the necessity of a mapping
Research question specification
Mapping protocol definition
Inclusion and exclusion criteria defintion
Search strategy definition
Data extraction strategy
Primary studies classificación
Data synthesis
Analysis of results

Fig. 4. Systematic literature mapping research methodology process.

Table 1
Definition of PICOC terms of the Systematic Literature M.

Term	Description
Population (P)	Artificial Intelligence of Things or Tiny Machine Learning.
Intervention (I)	Anomaly detection.
Comparison (C)	N/A.
Outcomes (O)	Solution OR Method OR Approach OR Development.
Context (C)	Internet of Things.

Table 2
List of Mapping Questions on the research.

No	Question
MQ1	How many studies have been published over the years, and how was this distribution constructed?
MQ2	Which ML/DL techniques are applied to detect anomalies through TinyML?
MQ3	Which data types are used to detect anomalies?
MQ4	Which ML/DL evaluation metrics are used for investigation?
MQ5	Which IoT devices (MCU) are used by the researchers?
MQ6	Which IoT conditions in terms of power and connectivity are used?
MQ7	In which use cases and domains are they applied?

3.1. PICOC

The authors in [67] suggest using the PICOC (Population, Intervention, Comparison, Outcome, Context) medical guideline when constructing study questions. The terms “Population” and “Intervention” refer to the group of elements under investigation, “Comparison” pertains to the intervention that was previously specified, “Outcome” indicates the expected results, and “Context” denotes the area of the study to which the intervention is applied and investigated. Table 1 displays the PICOC terms defined for the current research.

In this study, we are particularly interested in exploring the use of AI algorithms in IoT devices, specifically TinyML and MCU. We aim to use ML/DL algorithms to detect anomalies in these devices. Due to the nature of our intervention, we were unable to compare them. The results of our study include various solutions, techniques, approaches and developments. Our focus is solely on fog/cloud solutions when ML/DL algorithms are applied to EC within an IoT environment.

3.2. Research questions

Using the terms selected for the PICOC design, we can now formulate and clarify the research questions that this study aims to answer. In their publication [68], the authors highlighted the importance of having precise and focused research enquiries. As a basis for this mapping study, we have compiled a list of Mapping Questions (MQs), which can be seen in Table 2.

3.2.1. Inclusion/exclusion criteria

The objective of using inclusion and rejection criteria is to identify research papers that address the research questions. We have listed the inclusion standards to find the source documents in Table 3. As exclusion criteria are the opposite of inclusion criteria, we have not included them here.

3.3. Search strategy

This subsection outlines the core elements of our research methodology, including string search and strategic academic library selection.

Table 3
Inclusion criteria used in the research.

No.	Inclusion criteria
IC1	Articles that propose a research/empirical study and apply an ML/DL algorithm to detect anomalies;
IC2	Articles that apply an ML/DL algorithm on the EC layer;
IC3	Articles that necessarily have a title, abstract and keywords;
IC4	Articles published in English;
IC5	Articles published at peer-review events, such as journals, conferences or workshops.

Table 4
Search string used in the research.

No.	Search terms
1	“tinyml“ OR “tiny ml” OR “aiot“ OR “tiny machine learning” OR “Artificial Intelligence of Things"
2	“event detection“ OR “anomaly detection” OR “fault detection“ OR “abnormal behavior” OR “abnormal behaviour“ OR “anomaly prediction” OR “anomalous behavior recognition“ OR “anomalous behaviour recognition” OR “outlier detection"

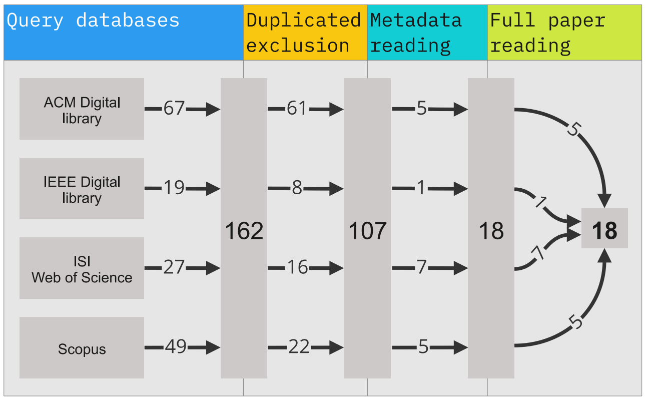

Fig. 5. Systematic literature mapping steps.

3.3.1. Sources

After describing the PICOC components, the research queries and the inclusion/exclusion criteria, we list the source libraries used to search. The four academic libraries chosen were ACM Digital Library, Web of Science, IEEExplore and Scopus. They were chosen based on the following criteria: (i) they are the main databases related to the subject of the review and with the best reputation; (ii) the ability to conduct searches using logical characters (such as “AND” and “OR”); and (iii) the ability to conduct searches in the document’s body and metadata fields (such as the title, keywords and abstract).

3.3.2. String search

Table 4 presents a list of the commonly used keywords included in the search to locate the papers that apply ML/DL models to detect anomalies in the context of TinyML or AIoT. The population displayed on PICOC, which consists of two elements, is included in the string search with all combinations (acronyms and complete words). The intervention described in Section 3.1 is referenced in the second sentence, which alludes to the element that concerns the investigation. The terms of the Outcomes are not included in order to get more results. The last line of the string refers to the specific context we are investigating. The term “Internet of Things” is also included in the subsets “TinyML” and “AIoT” to produce more relevant results for primary queries. Therefore, adding “IoT” to the initial search is unnecessary. This selection step is carried out over the whole of the step involving the reading of the paper, where only papers that present this kind of result will be added.

3.4. Data extraction process

The data extraction method has four main stages. Fig. 5 illustrates the key stages. Although we set all the libraries to search using only the metadata fields (title, keywords and abstract), we were able to search the ACM library using a field called “Anywhere”. Thus, we searched through all the documents and did not set a year limit due to the newness of this study. A total of 162 papers were found in the four scientific paper databases. The survey shows that 18 research studies were approved for an in-depth analysis.

We obtained the most recent findings within the last three years and excluded papers that were irrelevant to the scope covered by our study. We then identified and eliminated 55 duplicated files. After applying our exclusion criteria in the third step, we finally ended up with 89 articles. We performed a thorough examination of all the information, including keywords, titles and abstracts, using our inclusion/exclusion criteria. Finally, we obtained a set of 18 articles that addressed the subject matter of our research and conducted a more detailed review. Data extracted from each study include the type of publication (journal, conference, or book chapter), authors and their institutions, the country of publication, the study abstract, keywords, the URL and ISBN/ISSN/DOI of the publication (**see dataset published in Zenodo** [69]). The searches were conducted on 29 March 2023.

4. Systematic literature mapping report

Following the identification of the bounded papers, we can present the detailed results of the SLM. The thorough analysis and classification of the 18 primary studies were conducted to address the mapping questions outlined in Section 3.2.

4.1. Selected studies

The complete findings of SLM obtained from the appropriate papers that were selected can be seen in Table 5. To answer the mapping-related questions discussed in Section 3.2, a thorough analysis and categorisation of the 18 studies was carried out.

The analysed papers are divided into two groups: those that employ data from IoT sensors and those that use network data. The first group, application data, pertains to processed IoT sensor data that serve various industries such as healthcare and smart cities. On the other hand, network telemetry data are monitored to ensure the IoT network's health. A brief description of each work selected based on this categorisation is presented below.

A framework proposed by [70] applies Complex Event Processing (CEP) reasoning rules to sensor data in an industrial environment using EC devices. The approach uses a CNN, an Autoencoder NN, and multiple CEP rules on two Arduino Nano 33 Bluetooth Low Energy (BLE) boards. Meanwhile, [72] uses vibration data from sensors placed on industrial machines and analyses them using lightweight CNNs on a Raspberry Pi single-board microcontroller and perimeter inference on the ESP32 board. Furthermore, [83] presents an approach to detect anomalies in vibration data using an ARM Cortex-M4 microcontroller and an Autoencoder algorithm. Another industrial application for anomaly detection can be found in [77], which aims to detect abnormal values in the behaviour of submersible pumps by monitoring elements such as bearings, fans or similar equipment. This method uses Tiny-MLOps methodologies and an unsupervised ML technique capable of detecting anomalies. Lastly, the study in [82] presents an anomaly detection technique based on Sparse and Random NN for industrial quality inspection and predictive maintenance tasks.

In the industrial domain, some studies utilise camera-captured images to detect anomalies. One such system is presented in [75], which uses three cameras (OpenMV Cam H7 Plus) to detect artefacts and anomalies in plastic components automatically. The data processing is performed locally using TinyML with the CNN algorithm. Another system, described in [86], focuses on part recognition and defect inspection. The system is tested using a DNN algorithm and a platform of eight high-performance edge devices (Nvidia Jetson Nano) and eight low-performance devices (Raspberry Pi 4B). Finally, in [88], the authors proposed a scenario for a wastewater management plant, where a device runs an unsupervised anomaly detection algorithm based on ML to autonomously learn the expected behaviour of the industrial pump being monitored without relying on any external component.

Regarding energy consumption, [73] proposed a novel embedded solution to detect and diagnose anomalies for photovoltaic power generation systems. A fault detection and diagnosis algorithm based on Deep CNN (DCNN) and thermographic images has been developed and integrated into a Raspberry Pi 4 board. [80] proposed two different scenarios related to energy consumption where EC could be applied, including anomaly detection for Industry 4.0 based on a Hybrid Binary Automatic Encoder (HBAE). It uses two microcontrollers from the STM32 family.

The authors of [87] introduced a Typicality and Eccentricity Data Analytics (TEDA) algorithm to detect anomalies in data streams. This TinyML algorithm uses the concepts of typicality and eccentricity to achieve this. It is implemented through an OBD-II interface (On-Board diagnostics) and the Freematics ONE+ (ESP32) device, providing information on CO_2 emissions.

The field of health has seen significant application of AIoT. In a study by [71], an anomaly detection system was proposed for the Diabetes Management Control System (DMCS) using the Federated Learning-assisted CNN model. A Raspberry Pi device was used to provide patient care and maintain the normal range of glucose concentration in the body. Similarly, [78] discussed the use of microcontrollers with integrated Micro Electro-Mechanical System (MEMS) sensors in a real-life healthcare scenario. Adapted CNN models were employed to reduce power consumption.

A recent study [79] explored the use of AI to predict and detect anomalies in personal protective equipment. Helmets, bracelets and belts are equipped with sensors to collect data, which are then used to train models like CNN or Long Short-Term Memory (LSTM). A multisensory bracelet with an ESP32 microcontroller has also been developed to monitor the health of workers on construction sites, power plants and power lines [81]. LSTM models are used to detect anomalous readings.

AIoT technology has been used in the field of surveillance. A research study, referenced as [76], introduced a reliable and effective framework for identifying irregularities in extensive surveillance video data. The proposed model combines 2D-CNN and Echo State Network (ESN) for intelligent surveillance. CNN is employed as a feature extractor from input videos, which are then refined by the automatic encoder for feature enhancement, followed by ESN for sequence learning and detection of anomalous events.

An area where AI-powered technology has been implemented is in smart agriculture. Researchers at [85] have developed an AIoT monitoring system to monitor plant growth and identify nutrient deficiencies in indoor crops, such as lettuce. To achieve this, they used a DL algorithm to analyse images of hydroponic growth boxes and evaluate each lettuce plant individually.

Table 5
Extracted features from selected papers.

Paper	Domain	Data type (Dataset)	Algorithm	Metrics	Libraries	Training	Hardware	Connectiv.	Architecture
[70]	Industry	Vibration (own)	Autoencoder	N/A	TinyOL	Pre and post-trained	Arduino	WiFi and BLE	Single
[71]	Healthcare	Glucose level (own)	CNN, MLP	RMSE, accuracy, precision, recall and F1-score	TensorFlow	Pre-trained	Raspberry	Wireless	Multiple level
[72]	Industry	Bearing vibration (Case Western Reserve University [CWRU], Integrity Monitors [IMS] and own)	CNN	Accuracy and F1-score	TensorFlow Lite	Online training	Raspberry and ESP32	N/A	Single
[73]	Industry	IR images (own)	DCNNs	Precision, recall, F1-score and accuracy	TensorFlow Lite	Pre-trained	Raspberry	N/A	Single
[74]	Industry	Energy consumption (own)	RNN and LSTM	Accuracy, precision, recall, and F1-score	Original	N/A	Raspberry	WiFi	N/A
[75]	Industry	Images (own)	DNN	N/A	N/A	Pre-trained	STM32 and OpenMV Cam H7 Plus	N/A	N/A
[76]	Smart City	Video (UCF, ShanghaiTech, and Surveillance fight)	Autoencoder	AUC and accuracy	TensorFlow	N/A	Nvidia Jetson Nano	N/A	Single
[77]	Industry	IMU, temperature and vibration (own)	Isolation Forest	N/A	MicroPython	N/A	ESP32	USB	Single
[78]	Healthcare	Temperature, humidity, IMU, Proximity, pressure and sound (own)	CNN	N/A	N/A	Pre-trained	Arduino	BLE	Single
[79]	Industry and business	IMU, temperature sensor, pulse sensor and panic button (own)	GMM and LSTM	Accuracy	N/A	Pre-trained	Nvidia Jetson Nano	WiFi	Single
[80]	Industry	Bearing fault (CWRU)	HyBNN	MSE and accuracy	C implementation	Pre-trained	STM32	N/A	N/A
[81]	Healthcare	Temperature, IMU, cardiac pulse and resistive force (own)	GMM and LSTM	ROC	Scikit-learn	Pre-trained	Raspberry	WiFi	Single
[82]	Smart City	Images (MNIST, FashionMNIST, CIFAR10)	LSH	AUC	Scipy	Pre-trained	Raspberry	N/A	Single
[83]	Healthcare	IMU (own)	Autoencoder	F1-score, accuracy, specificity and sensitivity	TensorFlow Lite	Pre-trained	Arduino	BLE	Single
[84]	Industry	Network traffic (USTC-TFC2016)	CNN	N/A	TensorFlow Lite	Pre-trained	WIO	N/A	Single
[85]	Environm.	Images (own)	Isolation Forest	R2	TensorFlow	Pre-trained	Raspberry	WiFi	Nodes and central
[86]	Industry	Images (CIFAR10, NEU64)	DNN	N/A	TensorFlow	N/A	Raspberry	cable	Cluster edge
[87]	Industrial	GPS, Speed, RPM, Intake Air temperatures, Mass Air Flow	TEDA	N/A	N/A	N/A	ESP32 (Freematics ONE Plus Model A)	No	Single

Finally, in the second group, telemetry data from IoT device networks is used. In a study conducted by [84], they suggested using EC and small ML to classify network traffic and detect unusual behaviour. The same authors used TensorFlow Lite to transform a conventional CNN into a compact ML model that runs on an edge device. It is crucial to provide anomaly detection systems for Cyber–Physical Systems (CPS), and [74] achieved this by combining a One-Class Support Vector Machines (OCSVM) model with a Pairwise Self-supervised LSTM (PSLSTM) model.

4.2. Studies distribution

Fig. 6 shows the initial result of the searches carried out, in which the number of articles per year and academic library appear. We were able to identify 18 research papers that used ML/DL algorithms for anomaly detection on MCU following the steps described in Fig. 5. These papers were published between 2021 and 2023 and, despite the topic being relatively new, there has been a consistent increase in publications every year. The number of publications has doubled every year, with more than 77% of them published in

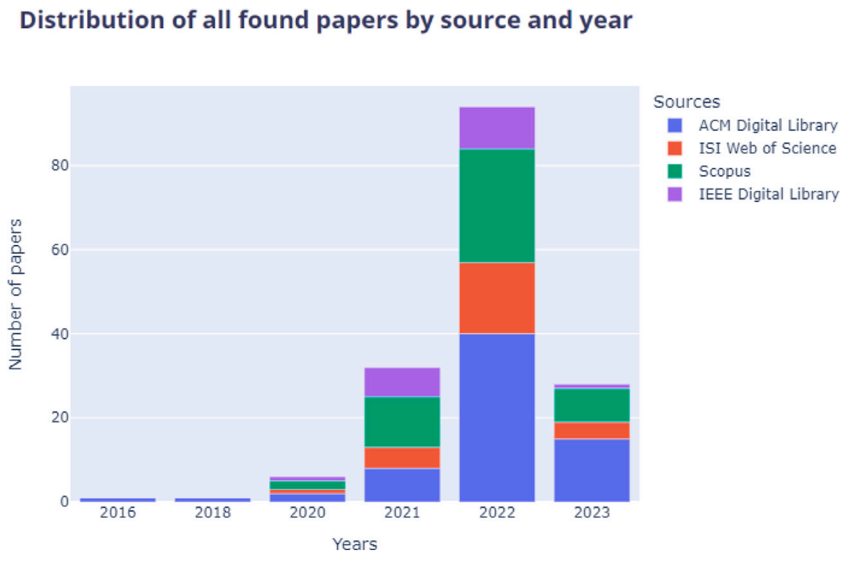

Fig. 6. Distribution of articles found by academic libraries and year of publication.

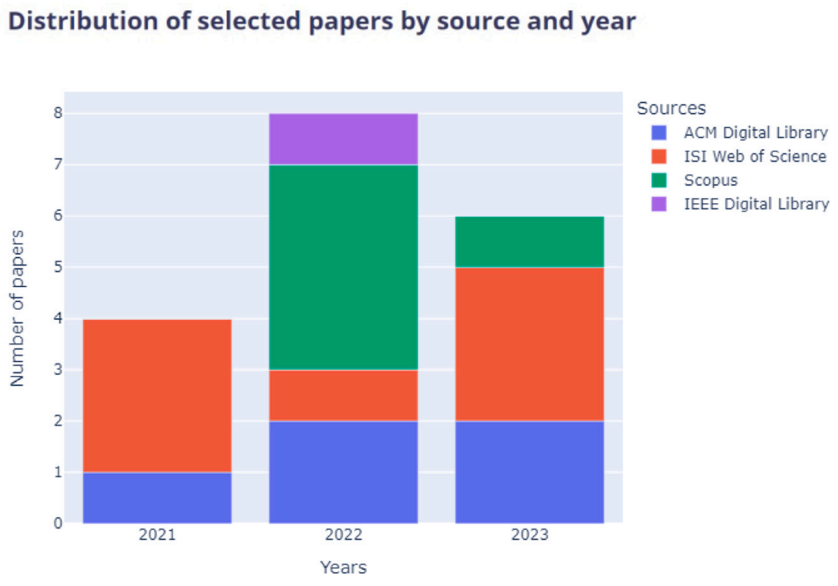

Fig. 7. Articles selected by academic libraries and year of publication.

the last two years. Although our search was conducted in March 2023, a substantial number of articles were published during the first months of this year. Furthermore, Fig. 7 shows the number of publications for each year.

Based on the data presented in Fig. 7, it can be seen that ISI Web of Science has the highest number of articles in the digital repository. In this source, seven studies account for 38.88% of the total number of articles selected. Of the chosen articles, 13 were published in journals and five were presented at conferences. Most of the selected articles (40%) were published in Sensors (MDPI), with six studies, while Future Generation Computer Systems had two articles, representing 66.66% of the journal papers.

Fig. 8 shows a map that indicates the main country associated with the author of each article. On the basis of this, we can conclude that Italy has produced 16.7% of the publications, with three studies. China and Spain, on the other hand, have each contributed two studies. Europe boasts the highest number of entries with seven papers, while Australia has only one article related to this topic.

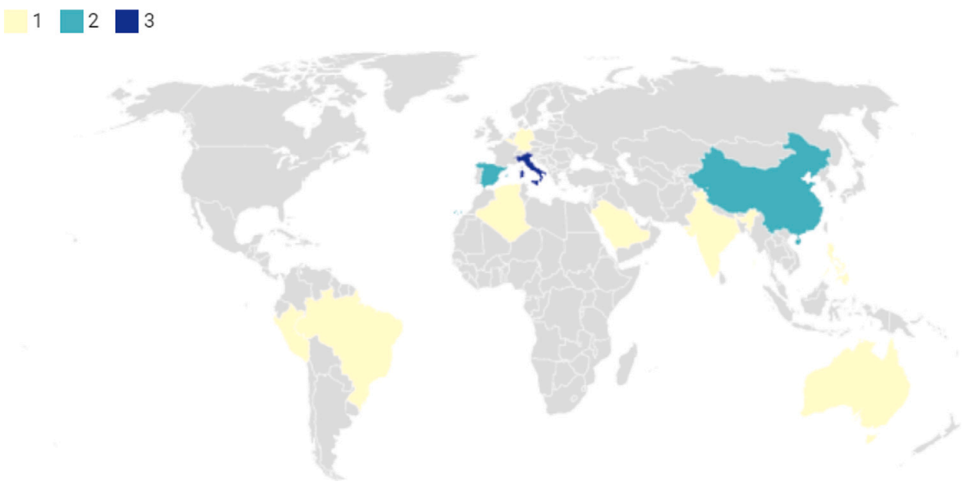

Fig. 8. Distribution of articles by country of authors' affiliations.

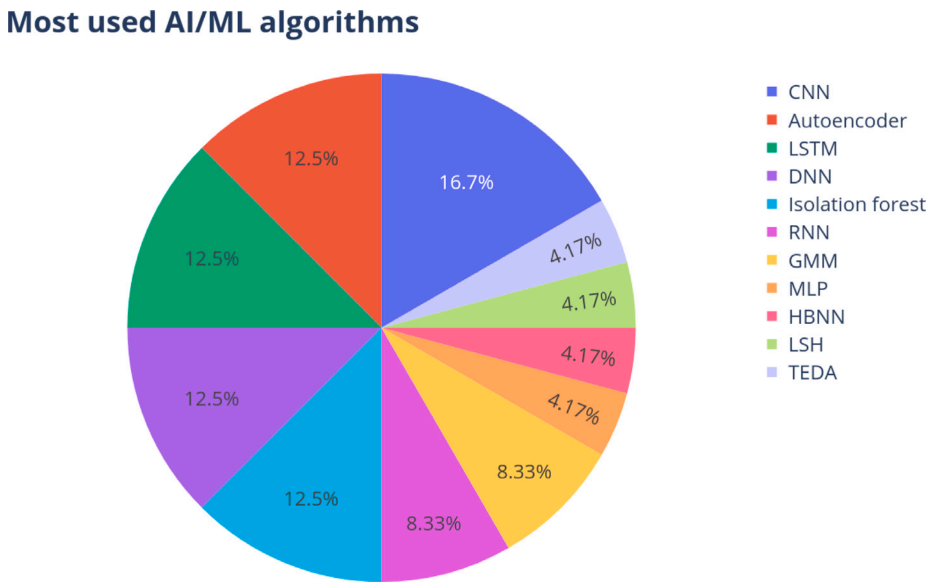

Fig. 9. Distribution of articles by ML/DL algorithms.

4.3. ML/DL techniques are applied to detect anomalies through TinyML

In the studies conducted, a total of 12 algorithms were used. Fig. 9 shows the percentage distribution of the most commonly used algorithms. CNN was the algorithm used most frequently, accounting for 18.8% of the studies. Autoencoder, LSTM, DNN, Isolation Forest, and Gaussian Mixture Model (GMM) followed closely behind, with percentages of 13.6% each. Conversely, Multilayer Perceptron (MLP), RNN, DCNN, TEDA, Hybrid Binary NN (HyBNN), and Locality Sensitive Hashing (LSH) were the least used algorithms, with only 4.55% each. If we classify the algorithms according to their typology, we can see that 42.86% were unsupervised, 52.36% were supervised and 4.76% were semi-supervised.

After examining the libraries and frameworks used in developing the techniques mentioned above, we have found a total of seven different options (see Fig. 10). The most commonly used library is TensorFlow, in any of its versions. The traditional version has been used four times, each time requiring a high-performance hardware device such as Raspberry PI (three occurrences) or Nvidia Jetson Nano (one occurrence). The TensorFlow Lite version has also been used four times, with a wide range of devices including ESP32, Arduino, Wio and Raspberry Pi. The second most popular option is MicroPython, which has been used twice, always in conjunction with a microcontroller of the ESP32 family. The TinyOL framework was adopted once by Arduino Nano 33 BLE Sense. The Scipy and Scikit-learn libraries were used only once, and on high-performance hardware such as Raspberry Pi. Finally, in the work cited in [80], an implementation was made without any external library, programmed in C for the ESP32 microcontroller.

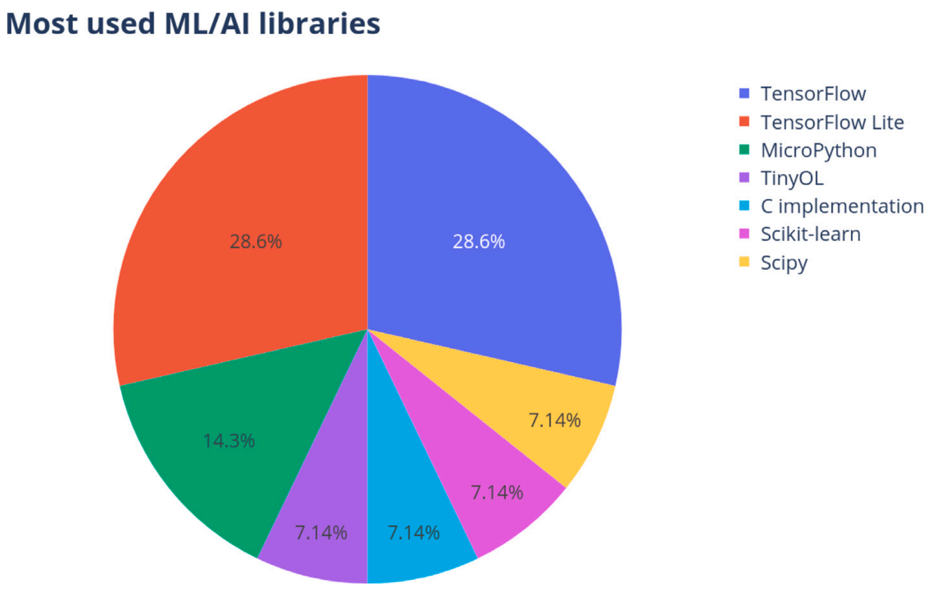

Fig. 10. Distribution of articles by ML/DL libraries.

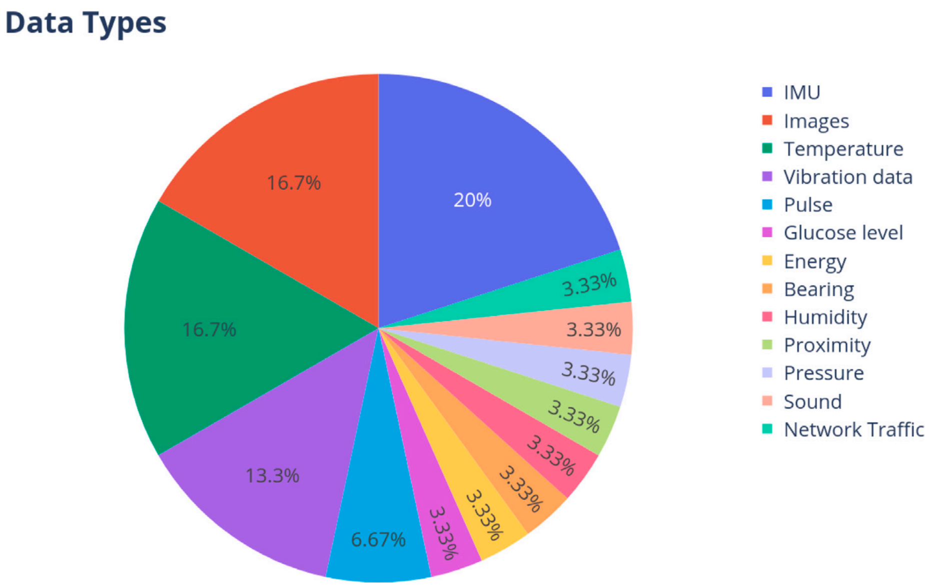

Fig. 11. Distribution of articles by data types.

4.4. *Data types used for anomaly detection*

Based on the selected research studies, the Inertial Measurement Unit (IMU) is the most widely used data type to detect anomalies in TinyML applications, accounting for 20% of the cases. Images and temperature data are each used in 16.7% of the studies. Vibration and pulse data contribute to 13.3% and 6.67%, respectively, for anomaly detection in embedded systems. Other data types, such as body glucose level (3.33%), energy (3.33%), bearing (3.33%), humidity (3.33%), proximity (3.33%), pressure (3.33%), sound (3.33%) and network traffic, are less frequently used. Fig. 11 illustrates the distribution of these data types.

Upon analysis of the source of data used in 12 works, it has been discovered that some studies employed their own datasets. Other studies, however, used previously published datasets such as CWRU (in two papers), IMS, UCE, ShanghaiTech, MNIST, FashionMNIST, CIFAC10 (in two articles) and NEU64.

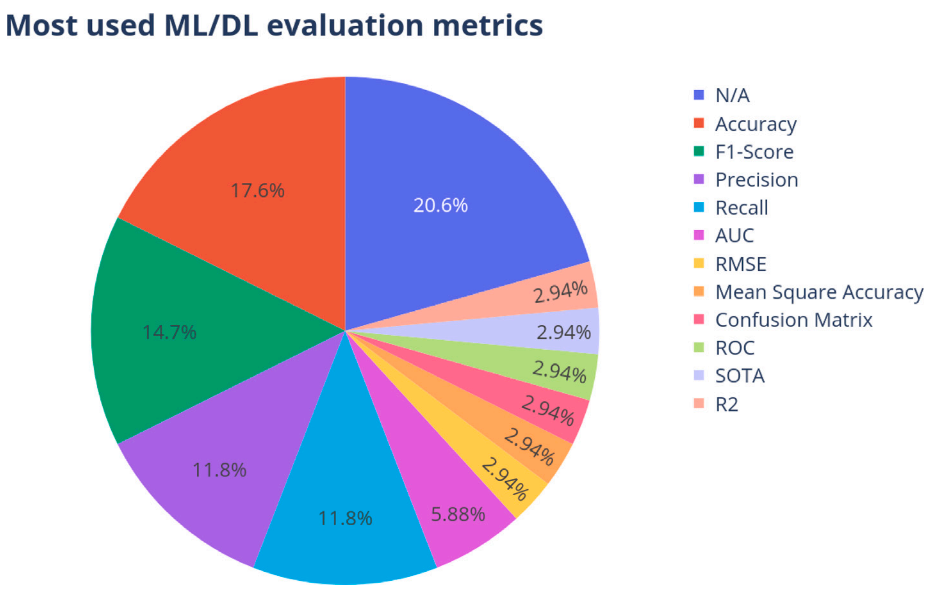

Fig. 12. Distribution of articles by the evaluation metrics.

4.5. ML/DL evaluation metrics

The evaluation metrics used to measure the applied methods are listed in Fig. 12. The chart displays the frequency of the metrics employed in various publications. Accuracy (ACC) was the most frequently used metric in eight publications. To assess the model's reliability, the F1-Score was used in five articles. Precision and recall were tested in three articles, while Root Mean Squared Error (RMSE) and Area Under the Curve (AUC) were tested in two articles, and Mean Squared Error (MSE), Coefficient of Determination (R^2), specificity and sensitivity were tested in one work. The authors did not disclose the metrics used to validate their models in seven articles.

4.6. MicroController units

Fig. 13 shows the distribution of the IoT devices used in the studies. Of the 18 studies, eight (42.1%) used Raspberry Pi as the processing system to deploy TinyML models, making Raspberry Pi the most used MCU system. It should be noted that almost all existing Raspberry Pi models have been used, including Raspberry PI 1, 2B, 3B, 3B + (two papers), 4 and 4B (two papers). The Arduino microcontroller was used in three papers (15.8%), and the Nano BLE 33 version was used in all of them. The ESP32 microcontroller was also employed in three different research studies; in two, the ESP32 DevKit V4 version was chosen and in one, the V1 version. Nvidia Jetson Nano (10.5%), STM32 (10.5%), WIO (5.26%) were used in two and one papers, respectively. The versions STM32F401RE and STM32H743 of STM32 have been used in the same study [75].

4.7. Power and connectivity

The connectivity used to transfer the results of the ML/DL model depends on the type of microcontroller, since each supports a different kind of connectivity (Fig. 14). Nine of the research studies (47.36%) use wireless connectivity, five of them use Wi-Fi, three relies on Bluetooth, and one does not specify the type of wireless connectivity. Regarding wired connectivity, one study indicates that it uses USB and another uses Ethernet. A total of eight papers do not indicate the type of connectivity used.

Concerning the power supply of IoT devices, only five research studies indicate that they are connected to the energy source by cable. None of the research studies refer to the use of batteries as a power source.

4.8. Use case and domain

The use case and application are divided into four main categories [37,89]. Fig. 15 shows the different categories. The prominent applied use case of the research works is Industry and business (61.1%) in eleven studies. The rates of application of TinyML in Healthcare, Smart City, and Environmental were found to be (22.2%), (11.1%) and (5.56%) respectively.

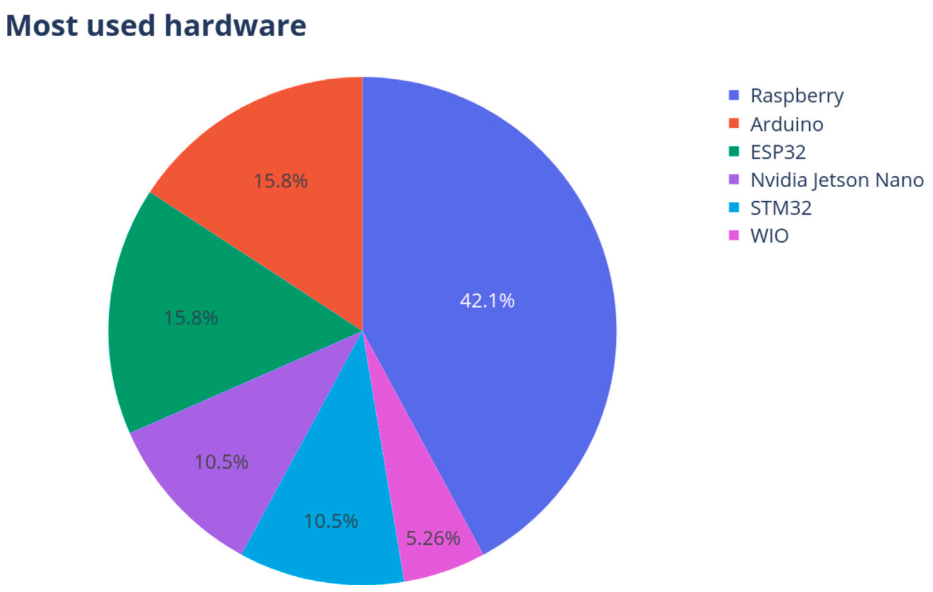

Fig. 13. Distribution of articles by hardware.

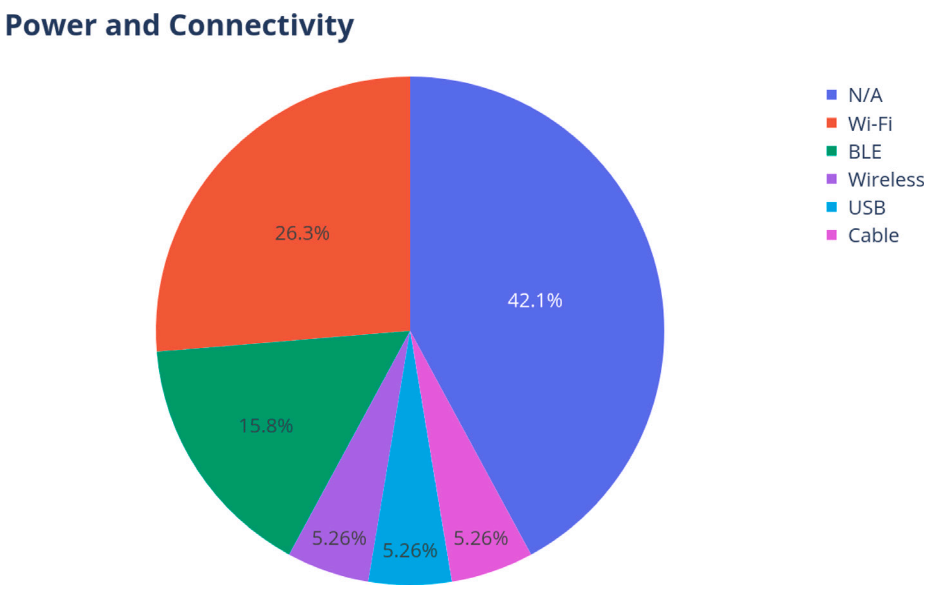

Fig. 14. Distribution of articles by power and connectivity.

5. Benefits and challenges of TinyML in anomaly detection

IoT and AI are destined to be the perfect symbiosis. As a data generator, the IoT produces the amount of data needed to feed the ML models, and ML techniques increase the potential of the IoT by making it smarter and generating decisions based on the data provided by the devices. As can be seen in Fig. 6, the topic of anomaly detection using TinyML is in a moment of splendour, despite the fact that TinyML is still in its preterm ages. In a few years, the number of articles published has undergone an exponential growth. This increase is largely due to technical advances in IoT and MCU, particularly, new hardware developments enable remarkable performance by allowing small local servers or microdata centres to be integrated into IoT devices [90]. TinyML allows us to run ML/DL models in the same place where the data is generated or the decision must be applied. The purpose is to enable IoT devices to be autonomous in decision-making, and not just messengers that deliver information to the control centre.

The utilisation of TinyML for anomaly detection is advantageous in IoT devices, as it can detect potential issues in the data quickly, thus avoiding them from spreading during the subsequent stages of data analysis and boost prompt decisions (e.g. image and

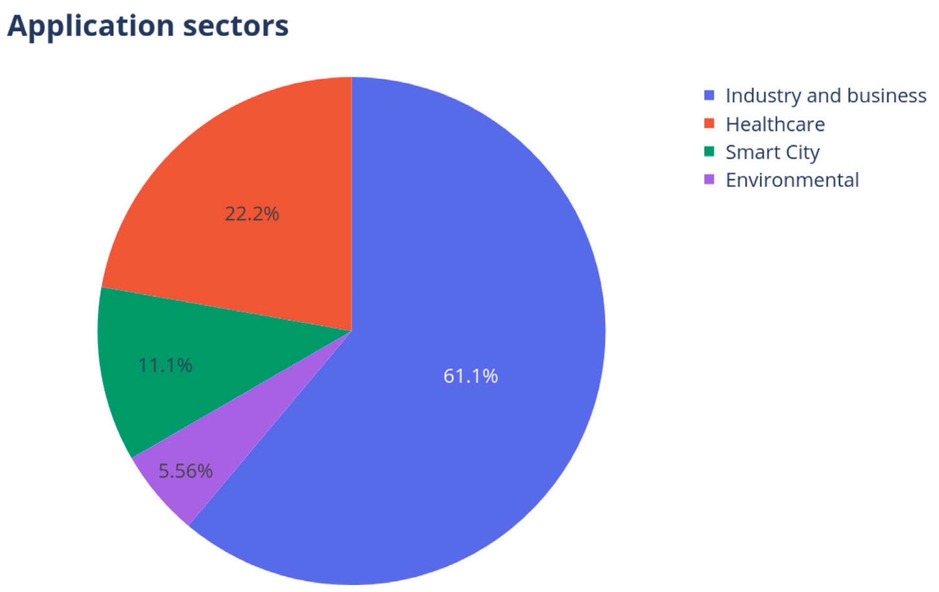

Fig. 15. Distribution of articles by use case and domain.

voice recognition). Anomaly detection with TinyML models is successfully employed in a variety of areas, such as health, agriculture, industry, and the environment. TinyML is motivated by the need to incorporate intelligence into a wide range of applications that were previously not feasible due to the computing and energy demands of traditional ML models. Here are some of the benefits that TinyML provides.

TinyML models are advantageous due to their ability to reduce latency, as they can be executed directly on the device. Processing data locally is much quicker than sending it to the cloud, especially when the data is substantial (e.g. pictures or videos). Moreover, TinyML systems provide high precision in their models, with only a minor decrease [91]. These discoveries show the capability of TinyML to reduce latency and keep up model accuracy, making it a desirable choice for deploying ML models on resource-limited devices. In the research studies analysed, twelve papers include execution performance results to carry out inference. None of them compare making an inference between the edge and the cloud. Inference times range from 2 to 534 ms.

TinyML is a form of ML that is specifically tailored to run on devices with limited computing power, such as microcontrollers found in IoT gadgets. The output of the model can be used for a variety of purposes, including controlling the device and sending data to the fog node or cloud for further analysis. To bring cloud services to the local area network, alternative methods for data analysis, such as federated computing [92] and EC, have been suggested. These technologies are used to register each IoT device on a gateway or fog node, allowing data extraction and processing from multiple sensors. In the studies analysed, the most common architecture is to apply the TinyML model on the MCU device in isolation. Twelve papers only describe the environment of the IoT device, without defining any layer architecture and executing the model locally without transferring the result to higher stages. Other papers, such as [71,85], define a multilevel architecture that transfers results to higher stages (fog or cloud). Finally, [86] establishes a cluster architecture of edge devices, in which they work together to obtain and share the results of the models.

Although EC and FC provide low latency, the main downside is that the wireless connection between the IoT device and the gateway or fog node must stay active for complex operations. TinyML models can be executed without an Internet connection, while cloud ML models necessitate such a connection. This is particularly important for applications deployed in areas without Internet access. Of the studies examined, seven do not establish an external connection or provide details about it. Six articles provide WiFi connectivity with the IP protocol, while two [78,83] use BLE to connect the device to a gateway. There are also wired works [77,85], which use a USB cable to transfer data. Despite the comprehensive search done, there is a noticeable gap in the existing scientific literature concerning the utilisation of cutting-edge connectivity technologies, such as Low Power Wide Area Network (LoraWAN) or 5G–6G, alongside TinyML.

TinyML algorithms offer a major advantage in terms of privacy and security. By keeping the data on the device, there is no need to send sensitive information to the cloud for processing. This has a significant impact on AI ethics, as it guarantees user data privacy, which is in line with data protection regulations. Eight articles have been published that demonstrate the value of using TinyML techniques to enhance the security and privacy of research.

Decreasing energy use in TinyML is essential and can be achieved by reducing the amount of data that needs to be transported and processed, as well as by employing algorithms that are computationally efficient. In addition, energy-efficient components, low-voltage and battery-powered operations, and sleep or idle modes can help reduce the power requirements of the device. Moreover, TinyML models can help to cut down on expenses related to sending data to the cloud for processing and storage, such as bandwidth and storage costs. Low energy consumption can also lead to cost reductions in TinyML. However, energy consumption is not a priority among the studies analysed, as only one provides consumption data during execution. Out of the five studies that indicate how they obtain energy, all of them directly connect to a source, while none of them use batteries or energy-saving techniques, or harvesting techniques via solar panels or other alternative sources.

Despite the benefits listed above that TinyML brings, its many limitations constitute obstacles to its widespread deployment. Some of the limitations of TinyML are presented below.

As we have seen, the hardware and software used in the different studies have great heterogeneity. Among them, we have detected seven hardware solutions and seven libraries/frameworks to facilitate the development of TinyML models. Because the embedded systems of manufacturing companies do not have a generalised method, each company has its own software to adapt the system to its devices. Therefore, no standardised method includes software as a framework to deploy and run ML/DL models on various devices from different companies without drops in accuracy. A generalised framework needs to be developed to increase the adoption and awareness of TinyML.

The difference in computing power between the different edge devices appears along with the heterogeneity of the devices. Currently, there are devices with uneven computational characteristics, from devices with high performance, such as those that include ARM Cortex processors (Raspberry Pi or Nvidia Jetson), to others with lower performance, such as ESP32. Devices with higher performance can be considered as small computers with higher consumption. These characteristics move them away from more restrictive environments where connecting to a light point is impossible.

Some IoT devices offer energy-saving functionality, and the TinyML philosophy must support this. TinyML models need a specific amount of energy to maintain the efficiency of the model at a certain level. Given the diversity in their architecture and preprocessing paths, hardware modules make it difficult to define a standard power mechanism. As sensors and other accessories are typically connected to edge devices, TinyML frames can experience poor power handling. Consequently, creating a low-power TinyML system remains a major obstacle.

At the opposite point, cloud-based systems can process massive amounts of data with large DL models. Therefore, the ability to perform these data analyses with large amounts of data could be improved by moving from the cloud to any device on the edge. Memory is also limited in edge devices, which in some cases have only a few KB, and this is one of the main motivations for creating TinyML. Traditional DL model inference requires dramatically higher maximum memory (gigabytes) than TinyML devices can provide. This poses a challenge, creating the need to innovate optimisation techniques [93] that are appropriate for each algorithm and MCU.

Reliability is an area that can be improved in EC. Variations in the process, physical and logical errors, and ageing can all be potential issues. To ensure that an edge device is suitable for any application, it is important to carry out a reliability assessment before deployment. During the article selection process, it became clear that TinyML in MCU devices still has room for improvement and is in its early stages. Different reviewed articles were not selected (see Fig. 6) as they only mentioned using TinyML algorithms without real experimentation on IoT devices (MCU). While most of these papers suggested that the models can be applied to real devices, they only provided evaluations of the simulated models.

6. Roadmap for anomaly detection in TinyML

Creating anomaly detection models individually for each IoT device is too expensive. IoT nodes and the data generated by them have certain characteristics that do not facilitate the generation of ML models. The characteristics are listed below. The first inconvenience is that the data generated by the IoT nodes are prone to errors due to various reasons, such as transmission errors. This may explain the reason why anomaly detection is one of the main research areas of IoT data. Additionally, most IoT devices are built with low-cost components and are located in hostile locations, so potential errors can occur during data capture. In addition to the problems outlined above, there are also those of degradation or poor calibration [94], which increase the uncertainty of the data produced. This instability can adversely affect the process of generating ML models and their subsequent inference. The second disadvantage is that IoT devices depend on the environment. Temporary or permanent changes in the environment can occur at any time and rapid adaptation is required. Due to the nature of ML models, their internal logic is always data-dependent. This feature is called "CACE: Change Anything Changes Everything" [95] and makes it challenging to maintain robust ML models, due to the tight dependencies of the components. The third problem concerns the dynamic nature of IoT devices. Many IoT devices have the characteristic of being mobile, so their environment will be different depending on the location. In these cases, data instability is even more critical as they can constantly vary. Another challenge, which we previously enumerated, is the wide variety of IoT devices. This means that the hardware properties of each device can cause divergences in the data generated. For example, an IoT node equipped with temperature sensors or an accelerometer may have different levels of quality (e.g. accuracy) compared to others, depending on the specifications of the sensors. The lack of initial data represents one of the most controversial points in the implementation of new IoT devices (with ML analysis), since, initially, there are no training data, or there is only a limited amount of them, so the generation of ML models will not be possible or adequate. There are also other challenges, not related to the data but affecting the applicability of the data. Another problem associated with this type of algorithm is the lack of data science experience. IoT project teams are usually made up of engineers from areas closer to electronics and hardware (but also software). These teams typically lack data science specialists. This gap makes it difficult to generate ML models that are adapted and optimised for the conditions of IoT devices. Finally, the lack of traceability and transparency in the generation of the ML model, since after its generation it is impossible to know the characteristics of how it has been generated (data, environment or IoT device, among others). This makes it difficult to reuse it on other similar devices.

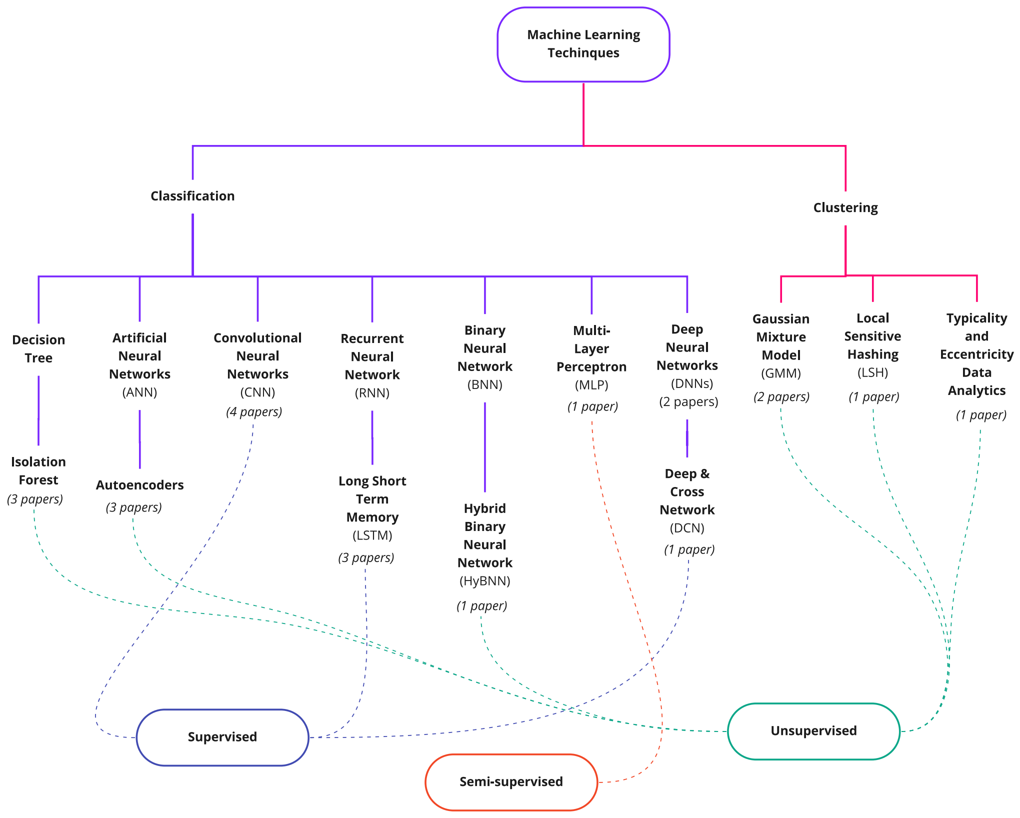
Machine Learning Techinques
Classification
Clustering
Decision Tree
Isolation Forest (3 papers)
Artificial Neural Networks (ANN)
Autoencoders (3 papers)
Convolutional Neural Networks (CNN) (4 papers)
Recurrent Neural Network (RNN)
Long Short Term Memory (LSTM) (3 papers)
Binary Neural Network (BNN)
Hybrid Binary Neural Network (HyBNN) (1 paper)
Multi-Layer Perceptron (MLP) (1 paper)
Deep Neural Networks (DNNs) (2 papers)
Deep & Cross Network (DCN) (1 paper)
Gaussian Mixture Model (GMM) (2 papers)
Local Sensitive Hashing (LSH) (1 paper)
Typicality and Eccentricity Data Analytics (1 paper)
Supervised
Semi-supervised
Unsupervised

Fig. 16. Distribution of articles by ML/DL algorithms.

In the studies analysed, it was found that ML algorithms, whether supervised or unsupervised, can effectively detect, identify, and categorise anomalies. The main advantage of both methods is their ability to learn from the vast and readily available data in the IoT environment, despite the fact that IoT data rarely includes labels indicating normal or anomalous. Due to the scarcity of labelled data, building a dependable anomaly detection model solely based on supervised learning algorithms is challenging. Moreover, the model must be capable of predicting anomalies for new and future data as the IoT data volume is continuously growing at a rapid pace, and this is a limitation of unsupervised learning algorithms.

Fig. 16 shows the taxonomy of ML/DL algorithms created based on the SLM performed. In it, algorithms have been classified according to their typology, classification or clustering. Each algorithm has been listed individually. In addition, depending on the group to which they belong (supervised, unsupervised or semi-supervised), they have been related. In this way, based on the type of problem to be solved, it can be helpful to determine which algorithms have been previously applied.

Validation and evaluation of models rely heavily on labelled data. However, datasets generated from IoT devices, such as the UCF dataset [76], lack normal and anomalous classes, which makes it difficult to establish a basic truth. This is also the case when working with your own dataset, as you will not have any labels to begin with. Although some datasets, like the CWRU dataset [72], are tagged for anomaly detection, there is still a need for labelled IoT datasets for network or perceptual information with anomaly detection, which should be addressed in future research.

Libraries specialising in TinyML for MCUs have been crucial in efficiently deploying ML models on edge devices with limited resources. One of the leading lightweight iterations of Google's TensorFlow framework tailored explicitly for MCU environments is TensorFlow Lite for Microcontrollers (TFLite Micro), which has been widely used in various studies due to its versatility [96]. uTensor is another notable contribution to the TinyML ecosystem. It is an open-source library designed specifically for microcontrollers and provides a lightweight yet powerful platform for deploying ML models [97]. ARM's CMSIS-NN (Cortex Microcontroller Software Interface Standard-Neural Network) offers a collection of efficient NN kernels optimised for ARM Cortex-M processors, enhancing the performance of ML workloads [98]. Additionally, TinyOL, an emerging library [99], has shown promise in recent studies, contributing to the diverse set of tools available for TinyML applications. Notably, selected studies have utilised TensorFlow Lite and TinyOL, underscoring their relevance and impact in the realm of TinyML for MCUs. It is crucial to continue enhancing these types of libraries by increasing their interoperability with a wide range of hardware. A powerful library could serve as a solution to the absence of a hardware standard.

In addition, it is important to perform IoT benchmarking when developing models. Benchmarking refers to evaluating and comparing solutions based on criteria such as performance or computational time using publicly available datasets. Some comparative studies on IoT, like the one in [100], do not make their data or algorithms available to the public.

7. Conclusions

Meeting the current demand requires the ability to make intelligent strategic decisions at the edge level. As upcoming application domains emerge, it becomes increasingly necessary to incorporate ML algorithms into compact embedded devices with limited resources. The frequency of publication in detecting anomalies using TinyML algorithms and MCUs shows that the field is still in its early stages. This paper performs an SLM that follows the PRISMA protocol to analyse the use of TinyML to detect anomalies. Throughout this document, the current use of TinyML algorithms for anomaly detection, application domains and difficulties in building TinyML applications are shown. It is currently believed that there is no one algorithm that is universally superior for this issue, rather, there are several methods that are tailored to specific applications. Due to the wide range and diversity of MCU devices, the widespread use of TinyML models is limited. The Raspberry microcontroller family is the hardware most commonly used to implement these algorithms. The fact that there is a wide variety of hardware devices, as we have seen, also implies a wide variety of libraries/frameworks (seven options in total) to enable the migration of ML/DL models to MCUs. The anomaly detection algorithms used in MCUs do not differ from those used in other stages, such as in the cloud. The most commonly used category of algorithms is classification, CNN being the most commonly used technique. For the type of clustering algorithms, the most widely used is GMM. Although a large number of related articles have recently appeared, no studies have been found that are compatible with the energy saving systems of IoT devices. None of the studies analysed uses more recent connectivity, such as LoraWAN or 5G, or alternative energy systems, such as solar power. So far, the most predominant use case or domain has been industry and business, healthcare, smart cities and environmental monitoring.

There are numerous challenges mentioned above, which can be explored and developed further. To overcome these difficulties, it is necessary to adopt interdisciplinary approaches that cover all the layers, including programming techniques, microarchitecture, and hardware. The following sectors have the potential for growth and require attention to address the challenges. Only three papers [71,85,86] of the 18 articles analysed make use of a multilevel architecture; that is, in most of them the results of the TinyML models are completed on the IoT device itself. For computational contexts enabled for the edge, compute task activation/offloading can be used. We can enhance the current flexible settings of edge-aware details by incorporating such loading strategies. By leveraging TinyML, we can move resource-intensive tasks from edge devices with limited resources, which can enable better connectivity between the cloud and the edge. Another aspect of improving the current state-of-the-art that denotes the degree of consolidation of this type of analysis is that only seven papers apply this model in a real-time environment. That is, the TinyML models generated are put into production for inference using their own data.

CRediT authorship contribution statement

Sergio Trilles: Conceptualization, Funding acquisition, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – original draft. **Sahibzada Saadoon Hammad:** Writing – review & editing, Visualization. **Ditsuhi Iskandaryan:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://doi.org/10.5281/zenodo.8231242.

Acknowledgments

Grant PID2019-104065GA-I00 funded by MCIN/AEI/10.13039/501100011033 and by “ERDF, a way of making Europe”, by the European Union. Grant PID2022-141813OB-I00 funded by MCIN/AEI/10.13039/501100011033 and by “ERDF, a way of making Europe”, by the European Union - Next GenerationEU/ PRTR. Sergio Trilles has been funded by the Juan de la Cierva - Incorporación postdoctoral programme of the Ministerio de Ciencia e Innovación - Spanish government (IJC2018-035017-I) funded by MCIN/AEI/ 10.13039/501100011033 and by “ERDF, a way of making Europe”, by the European Union. Sahibzada Saadoon Hammad has been funded by the ValgrAI - Valencian Graduate School and Research Network for Artificial Intelligence (Generalitat Valenciana).

References

[1] C. Granell, A. Kamilaris, A. Kotsev, F.O. Ostermann, S. Trilles, Internet of things, in: Manual of Digital Earth, Springer, Singapore, 2020, pp. 387–423.
[2] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.
[3] W. Shi, S. Dustdar, The promise of edge computing, Computer 49 (5) (2016) 78–81.
[4] W.Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, A. Ahmed, Edge computing: A survey, Future Gener. Comput. Syst. 97 (2019) 219–235.
[5] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput Netw. 54 (15) (2010) 2787–2805.
[6] S. Trilles, A. Luján, Ó. Belmonte, R. Montoliu, J. Torres-Sospedra, J. Huerta, SEnviro: A sensorized platform proposal using open hardware and open standards, Sensors 15 (3) (2015) 5555–5582.

[7] S. Trilles, Ò. Belmonte, S. Schade, J. Huerta, A domain-independent methodology to analyze IoT data streams in real-time. A proof of concept implementation for anomaly detection from environmental data, Int. J. Digit. Earth 10 (1) (2017) 103–120.
[8] A. Chatterjee, B.S. Ahmed, IoT anomaly detection methods and applications: A survey, Internet Things 19 (2022) 100568.
[9] A. Ukil, S. Bandyoapdhyay, C. Puri, A. Pal, IoT healthcare analytics: The importance of anomaly detection, in: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications, AINA, IEEE, 2016, pp. 994–997.
[10] A.A. Cook, G. Mısırlı, Z. Fan, Anomaly detection for IoT time-series data: A survey, IEEE Internet Things J. 7 (7) (2019) 6481–6494.
[11] D.M. Hawkins, Identification of Outliers, Vol. 11, Springer, 1980.
[12] B. Abraham, A. Chuang, Outlier detection and time series modeling, Technometrics 31 (2) (1989) 241–248.
[13] M. Markou, S. Singh, Novelty detection: a review—part 2:: Neural network based approaches, Signal Process. 83 (12) (2003) 2499–2521.
[14] M. Markou, S. Singh, Novelty detection: A review—part 1: Statistical approaches, Signal Process. 83 (12) (2003) 2481–2497.
[15] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. (CSUR) 41 (3) (2009) 1–58.
[16] Y. Zhang, N. Meratnia, P. Havinga, Outlier detection techniques for wireless sensor networks: A survey, IEEE Commun. Surv. Tutor. 12 (2) (2010) 159–170.
[17] R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, 2019, arXiv preprint arXiv:1901.03407.
[18] M. Fahim, A. Sillitti, Anomaly detection, analysis and prediction techniques in IoT environment: A systematic literature review, IEEE Access 7 (2019) 81664–81681, http://dx.doi.org/10.1109/ACCESS.2019.2921912.
[19] M.S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, A.P. Sheth, Machine learning for internet of things data analysis: A survey, Digit. Commun. Netw. 4 (3) (2018) 161–175.
[20] H. El-Sayed, S. Sankar, M. Prasad, D. Puthal, A. Gupta, M. Mohanty, C.-T. Lin, Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment, IEEE Access 6 (2017) 1706–1717.
[21] W. Shi, S. Dustdar, The promise of edge computing, Computer 49 (5) (2016) 78–81.
[22] K. Bajaj, B. Sharma, R. Singh, Implementation analysis of IoT-based offloading frameworks on cloud/edge computing for sensor generated big data, Complex Intell. Syst. 8 (5) (2022) 3641–3658.
[23] B. Dong, Q. Shi, Y. Yang, F. Wen, Z. Zhang, C. Lee, Technology evolution from self-powered sensors to AIoT enabled smart homes, Nano Energy 79 (2021) 105414.
[24] F.E.F. Samann, S.R. Zeebaree, S. Askar, IoT provisioning QoS based on cloud and fog computing, J. Appl. Sci. Technol. Trends 2 (01) (2021) 29–40.
[25] L. Dutta, S. Bharali, Tinyml meets iot: A comprehensive survey, Internet Things 16 (2021) 100461.
[26] M.Z. Gunduz, R. Das, Cyber-security on smart grid: Threats and potential solutions, Comput. Netw. 169 (2020) 107094, http://dx.doi.org/10.1016/j.comnet.2019.107094, URL https://www.sciencedirect.com/science/article/pii/S1389128619311235.
[27] H. Ahmetoglu, R. Das, A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions, Internet Things 20 (2022) 100615, http://dx.doi.org/10.1016/j.iot.2022.100615, URL https://www.sciencedirect.com/science/article/pii/S254266052200097X.
[28] P.P. Ray, A review on TinyML: State-of-the-art and prospects, J. King Saud Univ.-Comput. Inf. Sci. 34 (4) (2022) 1595–1623.
[29] H. Han, J. Siebert, TinyML: A systematic review and synthesis of existing research, in: 2022 International Conference on Artificial Intelligence in Information and Communication, ICAIIC, IEEE, 2022, pp. 269–274.
[30] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) http://dx.doi.org/10.1145/1541880.1541882.
[31] G. Pang, C. Shen, L. Cao, A.V.D. Hengel, Deep learning for anomaly detection: A review, ACM Comput. Surv. 54 (2) (2021) http://dx.doi.org/10.1145/3439950.
[32] A.A. Cook, G. Mısırlı, Z. Fan, Anomaly detection for IoT time-series data: A survey, IEEE Internet Things J. 7 (7) (2020) 6481–6494, http://dx.doi.org/10.1109/JIOT.2019.2958185.
[33] A. Chatterjee, B.S. Ahmed, IoT anomaly detection methods and applications: A survey, Internet Things 19 (2022) 100568, http://dx.doi.org/10.1016/j.iot.2022.100568, URL https://www.sciencedirect.com/science/article/pii/S2542660522000622.
[34] D. Moher, A. Liberati, J. Tetzlaff, D.G. Altman, P. Group*, Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement, Ann. Intern. Med. 151 (4) (2009) 264–269.
[35] A. Martín-Martín, E. Orduna-Malea, M. Thelwall, E.D. López-Cózar, Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories, J. Informetrics 12 (4) (2018) 1160–1177.
[36] S. Madakam, V. Lake, V. Lake, V. Lake, et al., Internet of things (IoT): A literature review, J. Comput. Commun. 3 (05) (2015) 164.
[37] D. Gil, A. Ferrández, H. Mora-Mora, J. Peral, Internet of things: A review of surveys based on context aware intelligent services, Sensors 16 (7) (2016) 1069.
[38] E. Borgia, The internet of things vision: Key features, applications and open issues, Comput. Commun. 54 (2014) 1–31.
[39] W. Yu, F. Liang, X. He, W.G. Hatcher, C. Lu, J. Lin, X. Yang, A survey on the edge computing for the internet of things, IEEE Access 6 (2017) 6900–6919.
[40] G. Premsankar, M. Di Francesco, T. Taleb, Edge computing for the internet of things: A case study, IEEE Internet Things J. 5 (2) (2018) 1275–1284.
[41] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, D.S. Nikolopoulos, Challenges and opportunities in edge computing, in: 2016 IEEE International Conference on Smart Cloud, SmartCloud, IEEE, 2016, pp. 20–26.
[42] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, T. Zhou, A survey on edge computing systems and tools, Proc. IEEE 107 (8) (2019) 1537–1562.
[43] M. Mohri, A. Rostamizadeh, A. Talwalkar, Foundations of Machine Learning, MIT Press, 2018.
[44] D. Michie, "Memo" functions and machine learning, Nature 218 (5136) (1968) 19–22.
[45] G. Holmes, A. Donkin, I.H. Witten, Weka: A machine learning workbench, in: Proceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference, IEEE, 1994, pp. 357–361.
[46] E. Alpaydin, Introduction to Machine Learning, MIT Press, 2020.
[47] V. Nasteski, An overview of the supervised machine learning methods, Horizons. b 4 (2017) 51–62.
[48] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, W.-Y. Lin, Intrusion detection by machine learning: A review, Expert Syst. Appl. 36 (10) (2009) 11994–12000.
[49] F. Zantalis, G. Koulouras, S. Karabetsos, D. Kandris, A review of machine learning and IoT in smart transportation, Future Internet 11 (4) (2019) 94.
[50] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: A review, Neurocomputing 187 (2016) 27–48.
[51] A. Mosavi, M. Salimi, S. Faizollahzadeh Ardabili, T. Rabczuk, S. Shamshirband, A.R. Varkonyi-Koczy, State of the art of machine learning models in energy systems, a systematic review, Energies 12 (7) (2019) 1301.
[52] Y. Hong, U. Hwang, J. Yoo, S. Yoon, How generative adversarial networks and their variants work: An overview, ACM Comput. Surv. 52 (1) (2019) 1–43.
[53] H.F. Nweke, Y.W. Teh, M.A. Al-Garadi, U.R. Alo, Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges, Expert Syst. Appl. 105 (2018) 233–261.
[54] A. Shrestha, A. Mahmood, Review of deep learning algorithms and architectures, IEEE Access 7 (2019) 53040–53065.
[55] H. Lin, S. Zeadally, Z. Chen, H. Labiod, L. Wang, A survey on computation offloading modeling for edge computing, J. Netw. Comput. Appl. 169 (2020) 102781.
[56] R. Das, M.M. Inuwa, A review on fog computing: Issues, characteristics, challenges, and potential applications, Telemat. Inform. Rep. 10 (2023) 100049, http://dx.doi.org/10.1016/j.teler.2023.100049, URL https://www.sciencedirect.com/science/article/pii/S2772503023000099.
[57] V. Barnett, T. Lewis, et al., Outliers in Statistical Data, Vol. 3. no. 1, Wiley New York, 1994.

[58] G.S. Madhuri, M.U. Rani, Anomaly detection techniques, in: 2018 IADS International Conference on Computing, Communications & Data Engineering, CCODE, 2018.
[59] M. Van Onsem, D. De Paepe, S.V. Hautte, P. Bonte, V. Ledoux, A. Lejon, F. Ongenae, D. Dreesen, S. Van Hoecke, Hierarchical pattern matching for anomaly detection in time series, Comput. Commun. 193 (2022) 75–81.
[60] L. Tran, M.Y. Mun, C. Shahabi, Real-time distance-based outlier detection in data streams, Proc. VLDB Endow. 14 (2) (2020) 141–153.
[61] R.K. Gunupudi, M. Nimmala, N. Gugulothu, S.R. Gali, CLAPP: A self constructing feature clustering approach for anomaly detection, Future Gener. Comput. Syst. 74 (2017) 417–429.
[62] X.-X. Lin, P. Lin, E.-H. Yeh, Anomaly detection/prediction for the internet of things: State of the art and the future, IEEE Network 35 (1) (2020) 212–218.
[63] A. Chohra, P. Shirani, E.B. Karbab, M. Debbabi, Chameleon: Optimized feature selection using particle swarm optimization and ensemble methods for network anomaly detection, Comput. Secur. 117 (2022) 102684.
[64] M. Weed, Sports tourism research 2000–2004: A systematic review of knowledge and a meta-evaluation of methods, J. Sport Tourism 11 (1) (2006) 5–30.
[65] L. Shamseer, D. Moher, M. Clarke, D. Ghersi, A. Liberati, M. Petticrew, P. Shekelle, L.A. Stewart, Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-p) 2015: elaboration and explanation, Bmj 349 (2015).
[66] S. Keele, et al., Guidelines for performing systematic literature reviews in software engineering, 2007.
[67] M. Petticrew, H. Roberts, Systematic Reviews in the Social Sciences: A Practical Guide, John Wiley & Sons, 2008.
[68] D. Budgen, P. Brereton, Performing systematic literature reviews in software engineering, in: Proceedings of the 28th International Conference on Software Engineering, 2006, pp. 1051–1052.
[69] S.T. Oliver, Analysis of anomaly detection for artificial intelligence of things: A systematic literature mapping, 2023, http://dx.doi.org/10.5281/zenodo.8231242.
[70] H. Ren, D. Anicic, T.A. Runkler, The synergy of complex event processing and tiny machine learning in industrial IoT, in: Proceedings of the 15th ACM International Conference on Distributed and Event-Based Systems, 2021, pp. 126–135.
[71] P.V. Astillo, D.G. Duguma, H. Park, J. Kim, B. Kim, I. You, Federated intelligence of anomaly detection agent in IoTMD-enabled diabetes management control system, Future Gener. Comput. Syst. 128 (2022) 395–405.
[72] S. Asutkar, C. Chalke, K. Shivgan, S. Tallur, TinyML-enabled edge implementation of transfer learning framework for domain generalization in machine fault diagnosis, Expert Syst. Appl. 213 (2023) 119016.
[73] A. Mellit, An embedded solution for fault detection and diagnosis of photovoltaic modules using thermographic images and deep convolutional neural networks, Eng. Appl. Artif. Intell. 116 (2022) 105459.
[74] D. Wang, F. Li, K. Liu, X. Zhang, Real-time cyber-physical security solution leveraging an integrated learning-based approach: An integrated learning-based cyber-physical security solution, ACM Trans. Sensor Netw..
[75] A. Albanese, M. Nardello, G. Fiacco, D. Brunelli, Tiny machine learning for high accuracy product quality inspection, IEEE Sens. J. 23 (2) (2022) 1575–1583.
[76] M. Islam, A.S. Dukyil, S. Alyahya, S. Habib, An IoT enable anomaly detection system for smart city surveillance, Sensors 23 (4) (2023) 2358.
[77] M. Antonini, M. Pincheira, M. Vecchio, F. Antonelli, An adaptable and unsupervised TinyML anomaly detection system for extreme industrial environments, Sensors 23 (4) (2023) 2344.
[78] O. D'Souza, S.C. Mukhopadhyay, M. Sheng, Health, security and fire safety process optimisation using intelligence at the edge, Sensors 22 (21) (2022) 8143.
[79] S. Márquez-Sánchez, I. Campero-Jurado, J. Herrera-Santos, S. Rodríguez, J.M. Corchado, Intelligent platform based on smart PPE for safety in workplaces, Sensors 21 (14) (2021) 4652.
[80] D. Pau, M. Lattuada, F. Loro, A. De Vita, G.D. Licciardo, Comparing industry frameworks with deeply quantized neural networks on microcontrollers, in: 2021 IEEE International Conference on Consumer Electronics, ICCE, IEEE, 2021, pp. 1–6.
[81] S. Márquez-Sánchez, I. Campero-Jurado, D. Robles-Camarillo, S. Rodríguez, J.M. Corchado-Rodríguez, Besafe b2. 0 smart multisensory platform for safety in workplaces, Sensors 21 (10) (2021) 3372.
[82] S. Leroux, P. Simoens, Sparse random neural networks for online anomaly detection on sensor nodes, Future Gener. Comput. Syst. 144 (2023) 327–343.
[83] K. Sai Charan, An auto-encoder based TinyML approach for real-time anomaly detection, in: 10TH SAE India International Mobility Conference, no. 2022-28-0406, 2022.
[84] E. Chen, A. Perez-Pons, Malware network traffic classification on the edge, in: 2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems, MASS, IEEE, 2022, pp. 754–758.
[85] E.A. Hacinas, W.M. Dimaculangan, S.J. Que, M.I.M. Tendido, A.S.M. Bambao, L. Tesoro, M.B.C. Raz, A.J. Cabtalan, J.A. Gonzaga, D.J.A. Rustia, AIoT-based system for indoor plant growth monitoring and early nutrient deficiency detection, in: 2022 ASABE Annual International Meeting, American Society of Agricultural and Biological Engineers, 2022, p. 1.
[86] Y. Jin, B. Huang, Y. Yan, Y. Huan, J. Xu, S. Li, P. Gope, L. Da Xu, Z. Zou, L. Zheng, Edge-based collaborative training system for artificial intelligence-of-things, IEEE Trans. Ind. Inform. 18 (10) (2022) 7162–7173.
[87] P. Andrade, I. Silva, M. Silva, T. Flores, J. Cassiano, D.G. Costa, A tinyml soft-sensor approach for low-cost detection and monitoring of vehicular emissions, Sensors 22 (10) (2022) 3838.
[88] M. Antonini, M. Pincheira, M. Vecchio, F. Antonelli, A TinyML approach to non-repudiable anomaly detection in extreme industrial environments, in: 2022 IEEE International Workshop on Metrology for Industry 4.0 & IoT, MetroInd4. 0&IoT, IEEE, 2022, pp. 397–402.
[89] N. Schizas, A. Karras, C. Karras, S. Sioutas, TinyML for ultra-low power AI and large scale IoT deployments: A systematic review, Future Internet 14 (12) (2022) 363.
[90] R. Mahmud, A.N. Toosi, Con-pi: A distributed container-based edge and fog computing framework, IEEE Internet Things J. 9 (6) (2021) 4125–4138.
[91] S.S. Hammad, D. Iskandaryan, S. Trilles, An unsupervised TinyML approach applied to the detection of urban noise anomalies under the smart cities environment, Internet Things 23 (2023) 100848.
[92] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I.M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, et al., The reservoir model and architecture for open federated cloud computing, IBM J. Res. Dev. 53 (4) (2009) 4:1–4:11.
[93] Y. Wang, R. Nahon, E. Tartaglione, P. Mozharovskyi, V.-T. Nguyen, Optimized preprocessing and tiny ML for attention state classification, 2023, arXiv preprint arXiv:2303.11371.
[94] J.M. Barcelo-Ordinas, M. Doudou, J. Garcia-Vidal, N. Badache, Self-calibration methods for uncontrolled environments in sensor networks: A reference survey, Ad Hoc Netw. 88 (2019) 142–159.
[95] S. Okuda, G. Nemoto, T. Mori, N. Ishitani, K. Nishimura, N. Uchihira, Exploitation pattern for machine learning systems, in: 2021 36th International Technical Conference on Circuits/Systems, Computers and Communications, ITC-CSCC, IEEE, 2021, pp. 1–4.
[96] ML for mobile and edge devices - TensorFlow lite, 2023, URL https://www.tensorflow.org/lite. (Accessed 10/11/2023).
[97] Utensor/utensor: Tinyml AI inference library, 2023, URL https://github.com/uTensor/uTensor. (Accessed 10/11/2023).
[98] CMSIS NN software library, 2023, URL https://www.keil.com/pack/doc/cmsis/NN/html/index.html. (Accessed 10/11/2023).
[99] H. Ren, D. Anicic, T. Runkler, TinyOL: TinyML with online-learning on microcontrollers, 2021, arXiv:2103.08295.
[100] C.R. Banbury, V.J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, et al., Benchmarking tinyml systems: Challenges and direction, 2020, arXiv preprint arXiv:2003.04821.