Hybrid Anomaly Detection Model on Trusted IoT Devices

Paul D. Rosero-Montalvo, Zsolt István, Pınar Tözün, and Wilmar Hernandez, *Senior Member, IEEE*

***Abstract*—Most machine learning proposals in the Internet of Things (IoT) are designed and evaluated on preprocessed data sets, where data acquisition and cleaning steps are often considered a black box. In addition, IoT environments have numerous challenges related to acquiring data from sensors, where sensitive data can be threatened by malicious users who seek to interfere with the communication channel or storage. Additionally, sensor data can also be affected by noise. Therefore, differentiating the type of threat/anomaly requires additional energy and computational resources. We propose to carry out data cleaning/anomaly techniques on the IoT device itself, not in the cloud servers but closer to the data source. Therefore, the IoT device sends trusted data to the Cloud. Among the benefits of this is the considerable reduction in the cost of implementation due to less movement of data between IoT devices and the Cloud. Consequently, we define three anomaly detection steps using smoothing filters, unsupervised learning, and deep learning techniques (i.e., hybrid model) to detect different variations of anomalies and threats while focusing on a small computational/memory footprint. The deployment of the hybrid model on AVR, Tensilica, and ARM microcontrollers showed that the last ones are an adequate target to implement the model because they best satisfy the necessary hardware requirements. The proposed model consumes 50 kB of Flash and 12 kB of RAM and processes data locally, achieving a bandwidth reduction of 60%. Finally, the hybrid model was tested in external data sets.**

***Index Terms*—Data tampering, hybrid model, Internet of Things (IoT), outlier detection.**

I. Introduction

NOWADAYS, many research fields use machine learning (ML) techniques due to their capability to identify and describe patterns [1]. Often, the data sets from the Internet of Things (IoT) applications are time-series data since the source of the data set is sensors and similar devices "interacting" with the real world, sending their collected data to high-performance servers located in the cloud [2], [3]. These servers are the ones used to run ML algorithms because such algorithms are computationally complex—naturally, much effort in ML proposals goes into optimizing the computations in the cloud and making such servers run more efficiently. Nevertheless, many proposals that apply ML ideas on top of IoT/sensor data assume that the data set has been "cleaned" and is typically treated as a black box [4].

Manuscript received 23 September 2022; revised 3 December 2022 and 2 January 2023; accepted 3 February 2023. Date of publication 7 February 2023; date of current version 7 June 2023. This work was supported in part by Novo Nordisk Fonden under Award NNF20OC0064411; in part by the Corporacion Ecuatoriana para el Desarrollo de la Investigacion y la Academia (CEDIA), Ecuador (Research Project) under Grant CEPRA XII-2018-13; in part by the Universidad de Las Americas (UDLA), Quito, Ecuador (Research Project) under Grant IEA.WHP.21.02; and in part by the LOEWE Initiative (Hesse, Germany) within the emergenCITY Center. *(Corresponding author: Wilmar Hernandez.)*

Paul D. Rosero-Montalvo and Pınar Tözün are with the Computer Science Department, IT University of Copenhagen, 2300 Copenhagen, Denmark (e-mail: paur@itu.dk; pito@itu.dk).

Zsolt István is with the Computer Science Department, Technical University of Darmstadt, 64293 Darmstadt, Germany (e-mail: zsolt.istvan@cs.tu-darmstadt.de).

Wilmar Hernandez is with the Facultad de Ingenieria y Ciencias Aplicadas, Universidad de Las Americas, Quito 170124, Ecuador (e-mail: wilmar.hernandez@udla.edu.ec).

Digital Object Identifier 10.1109/JIOT.2023.3243037

In the IoT space, the ability of electronic devices to work close to the end users or interact directly with the environment could generate incorrect sensor readings because they are used in indoor and outdoor environments, often with low/no maintenance [5], [6]. Additionally, extreme weather conditions can impair performance and cause component wear [7]. Therefore, IoT-based solutions can also help to detect hardware damage due to environmental conditions. For example, IoT systems can use the borescope inspection technique in monitoring and maintaining the health of aero-engine blades and, with the help of deep learning techniques, can produce an intelligent diagnosis. This being a topic of upmost importance, because it involves the safety of human beings [8], [9], [10]. As a result, IoT devices periodically send collected data to the cloud for further processing. By some estimates, 80% of the battery consumption of IoT devices is related to wireless communication with the edge/cloud [11]. Moreover, in some scenarios, IoT devices send data to the cloud without bandwidth restriction/limitation carrying on bottlenecks and unnecessary storage on the server when data is not processed for a specific target [12].

Since IoT devices gather data from different scenarios, data protection is commonly needed to acquire, manage, send, and process them [13]. Therefore, sensitive data could be threatened or compromise their integrity, and final users or companies can lose control when they are stored in cloud servers, causing unpredictable outputs from the ML model [14]. Consequently, the main security/privacy concern is managing the high quantity of generated data to protect storage and communication. Therefore, detecting anomalies/threats must be done before sending data to the cloud [15]. Thus, cloud computing could consider that the IoT device is trusted since the sensor data is secure to manage/process [16]. Consequently, anomaly detection techniques allow recognizing if data is statistically different from the rest, and such differences can be divided into intrinsic error (genuinely fault sensor), sensor event (an actual event that increases or

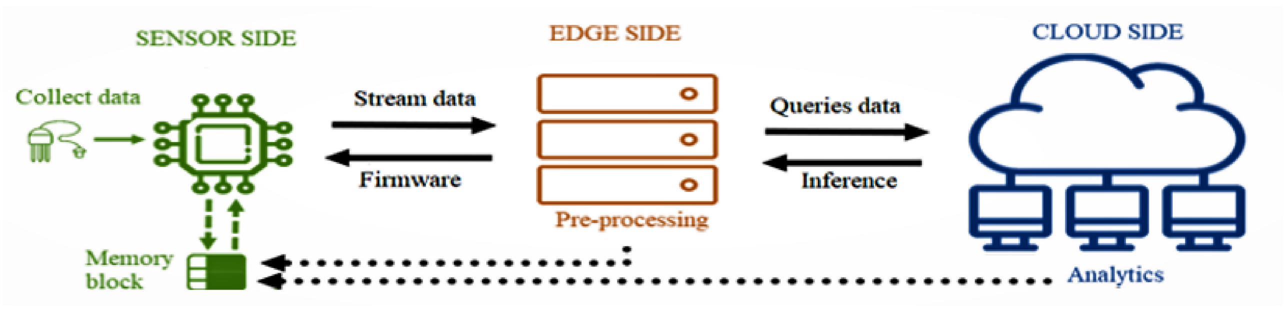


Fig. 1. *Solid black arrows*, traditional data movement; *dotted arrows*, lightweight ML models exported to the IoT device from different locations; and *square dotted arrows*, hybrid model allocated in memory. In addition, the sensor side comprises the devices along with microcontrollers. Furthermore, the edge side consists of gateways with capabilities to deploy data preprocessing models, payload errors, empty cells, and manning the wireless communication architecture. Finally, the cloud side is a powerful server with the ability to train heavy ML models.

decreases its value dramatically), and intermittent sensor error (data tampering) [17].

The most representative anomaly detection techniques fall into four categories: 1) statistical; 2) unsupervised; 3) supervised; and 4) deep learning. However, single models have yet to prove exceptional performance [4], [18], [19]. Moreover, several works focused on deploying a single anomaly detection model show that such setups do not prevent the possibility of sending erroneous data to the cloud [5]. Therefore, in recent years, hybrid models are getting more attention and demonstrate prominent advantages when a set of anomaly detection techniques are merged into one single solution [11]. Nevertheless, a hybrid system with several anomaly detection algorithms is typically deployed outside the IoT device due to computational constraints, limited storage capacity, and battery power limitations [20].

This article aims to reduce the cost of processing/detecting anomalies in the cloud and ensure data integrity by processing data locally. Following novelty decentralized computational architectures [19], this work achieves this goal by designing and deploying a hybrid ML model that detects anomalous data and threats by a threat model on IoT devices. However, there are open challenges, such as the IoT device might recognize either data is compromised by sensor fault or is a security threat that someone is tampered with the sensor to inject errors. Besides determining the optimum steps to deploy a hybrid anomaly detection model on the device. Related to the literature, the most representative anomaly detection techniques are as follows: noise reduction [3], [21], outlier detection [22], [23], [24], and tamper/threat detection [17], [25]. Based on [18] and [26], we conducted an extensive literature review focused on determining the underlying algorithms and how these could be ported to emerging microcontrollers with better computational capacity. Therefore, we explore a range of possible algorithms to determine how well they work on small IoT devices.

To prove our proposal on how to develop the hybrid model locally, Fig. 1 shows in solid black arrows how traditionally data is sent to the Edge where it is stored. Then, from the cloud side, computer scientists query data to develop ML models [27]. Next, the inference goes back to the Edge to update the firmware to send it back to the IoT device. In contrast, with the computational power of new microcontrollers, the dotted arrows show how some ML algorithms that typically work outside the device can be quantized to be processed locally in the IoT memory (shown in square-dotted arrows in Fig. 1). As a result, our proposal aims to reduce the data movement, latency, and risk of compromising data sending them through communication networks. The contributions of this work can be summarized as follows.

1) We propose a three-step hybrid anomaly detection model targeting to run on an IoT device considering the most relevant anomaly detection approaches (noise reduction, outlier detection, and tamper/threat detection) and evaluate them in terms of accuracy and metrics like memory consumption and bandwidth on the IoT device.
2) Our results demonstrate that smoothing filters, specifically the Gaussian algorithm, are the most effective for improving samples for the first noise reduction step. Then, unsupervised learning methods are suitable solutions for the second outlier detection step. Especially, the one-class support vector machine (OSVM) algorithm reduces scatter significantly when obtaining cleaned samples. Finally, for the third tamper detection step, we design a threat model to define assumptions where the IoT device must classify either a tamper/threat situation or regular sensor reading. As a result, a model based on deep neural networks can detect data tampered adequately whether the algorithm is trained with the tamper possibilities.
3) After determining the most suitable technique for each step, the hybrid anomaly detection model is tested in real environments with three microcontroller families, demonstrating that ARM microcontrollers are an adequate target to deploy this hybrid model satisfying the necessary hardware requirements and reducing the bandwidth need between IoT devices and the Edge/Cloud.

The remainder of this article is structured as follows. Section II shows related works. Then, Section III presents the hybrid anomaly detection model proposed. Next, results are discussed in Section IV. Finally, Section V presents the conclusions of this article and future work.

II. Related Works

There are several works with different approaches to detecting anomalies. Consequently, works, such as Ghosh et al. [28], Gaddam et al. [29], Choi et al. [1], and Cook et al. [30], presented earlier state-of-the-art studies showing the most relevant techniques used for detecting anomalies. These works offer statistical-based approaches, which assume some probabilistic data distribution categorized into either parametric or nonparametric methods.

A. Traditional Models

Anomaly detection in IoT environments got attention since Zhang et al. [31] presented the first intuition about it and how data with errors can impair the actuation of control systems. This work opens opportunities to other researchers like Shen and Wang [32], which presents the importance of the data validation stage before processing them. Later, Chacko and Bharati [33] presented the risks of bringing incorrect information to the cloud and how lightweight ML algorithms can be compiled in IoT environments. Besides, Yu et al. [5] presents a principal component analysis-based data outlier detection and sensor data. This work defines an architecture for sensor aggregation in the cloud. Then, Nesa et al. [22] considers different techniques to reach outlier detection. As a result, this work presented metrics to compare those algorithms with classification and statistical approaches to determine the performance of each technique. Wang et al. [34] used several algorithms, such as isolation forest (ISO) and local outlier factor (LOCAL), and their mathematical approach to present their isolation nearest neighbor ensemble (iNNE) technique. Conversely, Sood et al. [17] presented the first initiative to define a high-level tamper detection combining some algorithms with different approaches: statistical, supervised, and unsupervised techniques. Therefore, the model complexity increases at the same time as its robustness.

B. Hybrid Models

Recently, anomaly detection has been studied based on new trends when the flow of information processed by the IoT data has grown exponentially. For these trends, Zhu et al. [11] proposed a novel framework named grid-based approximate average outlier detection (GAAOD) to support KNN-based outlier detection over IoT streaming data. Furthermore, Zhang et al. [4] proposed a three-level hybrid model based on: median filter (MF), empirical mode decomposition (EMD), classification and regression tree (CART), autoregression (AR), and exponential weighted moving average (EWMA) methods (called MF-EMD-CARTAR-EWMA) to detect outliers in sensor data. The hybrid model was evaluated using a real hydrometeorological observation network data set. Then, Wu et al. [3] defined a long short-term memory (LSTM)-Gauss-Bayes method, which is a synergy of the LSTM neural network (LSTM-NN) and the Gaussian Bayes model for outlier detection in the IoT devices. Next, regarding the new tendency to use decentralized computational systems, Garmaroodi et al. [19] presented an intelligent IoT system applied in pharmaceutical systems to detect anomalies in water purifying pumps. In addition, Liu et al. [37], and Wang et al. [38] demonstrated an application of federated learning in anomaly detection in industrial environments using hybrid models. Furthermore, Liu et al. [26] and Wang et al. [38] deployed a decentralized solution where edge devices collaborate to train anomaly detection models. Besides, Garmaroodi et al. [19] presented two anomaly detection approaches running on the device. Finally, Zhang et al. [36] proposed a data edge verification mechanism based on blockchain to ensure that data is not tampered with. As illustrated by these works, a new trend exists to deploy a decentralized computational architecture where IoT devices are part of the ML pipeline running the inference of the model.

Summarizing the most relevant works and their approach, Table I shows the applied algorithm, the used algorithm, and the location where the ML algorithm was developed. Also, Table I shows relevant information about the algorithm selection for the following sections presented in this research. However, there are open challenges even when previous works presented several solutions to face anomaly detection in sensor data and IoT devices. For example, bringing these solutions into the IoT device due to the new computational capabilities of the microcontrollers. Moreover, several works present anomaly detection hybrid models using different ML methods. Nevertheless, tamper detection techniques are not deployed as part of hybrid models, even with the increasing concerns about data confidentiality.

III. Hybrid Anomaly Detection Model

ML applications in IoT environments start with the data collection by sensors, the next step is data anomaly detection. However, even though usually just the model inference step is deployed on the IoT device [39], only some techniques can be exported to IoT devices due to their expensive hardware requirements. In addition, these anomaly detection techniques are applied in different ways to reach individual purposes, such as noise reduction (improved samples), outlier detection (cleaned samples), and tamper detection (processed samples). Therefore, it is necessary to present a hybrid model merging the mentioned methods to detect erroneous, unpredictable, and unexpected data [40]. However, this hybrid model must be compiled with the minimum computational footprint [41]. As a result, the IoT device stores in memory and runs in sequence different ML models related to detecting all the anomaly possibilities. Fig. 2 shows the following steps of the proposed hybrid model and the corresponding data flow, starting with the original samples, which have noise (represented by variations), outliers (represented by red circles), and tampered data (represented by red dots).

A. Threat Model

The hybrid anomaly detection model defines tamper detection as the final stage after sending data to the Edge/Cloud. Therefore, the threat model proposed identifying the potential vulnerability as follows.

TABLE I
OVERVIEW OF HYBRID ANOMALY DETECTION MODELS TECHNIQUES: STATISTICAL (★), SUPERVISED (♦), UNSUPERVISED (■), AND DEEP LEARNING (▲)

Reviewed Works	Technique	Algorithms	Anomaly method	Model Location
Wu et al. [3]	♦ ▲	LSTM and Gaussian Naive Bayes	Noise reduction	Edge
Mansoor et al. [21]	♦ ■	SVM, k-NN, Bayes, Local outlier	Noise reduction	Edge
Sood et al. [17]	★ ♦	Regression trees, Random forest and SVM	Tamper detection and attacks	Cloud
Nesa et al. [22]	★	Regression Trees, Gradient Boosting Machine and Linear Discriminant Analysis	Outlier detection	Edge
Zhang et al. [4]	★ ♦	Filters, SVR, k-NN	Tamper/Noise detection	Server
Awawdeh et al. [23]	★ ■	Regressions model, iForest and Standard deviation	Outlier detection	Cloud
Zidi et al. [25]	♦	SVM and Naive Bayes	Tamper detection	Cloud
Deng et al. [24]	■	One class support machine	Outlier detection	Edge
Wang et al. [34]	■	Nearest-neighbor assembles and One-class SVM	Outlier detection	Cloud
Zhang et al. [35]	♦ ▲	Neural Network	Outlier/noise detection	Cloud
Zhang et al. [36]	★ ▲	Filters and Neural Network	Outlier/noise detection	Edge
Liu et al. [37]	▲	Convolutional neural network-long short-term memory	Outlier/noise detection	Edge
Alghanmi et al. [18]	♦ ■	Decision Tree and Hierarchical Affinity Propagation	Outlier detection	On-device
Garmaroodi et al. [19]	♦ ■	SVM and Hierarchical Affinity Propagation	Noise detection	On-device

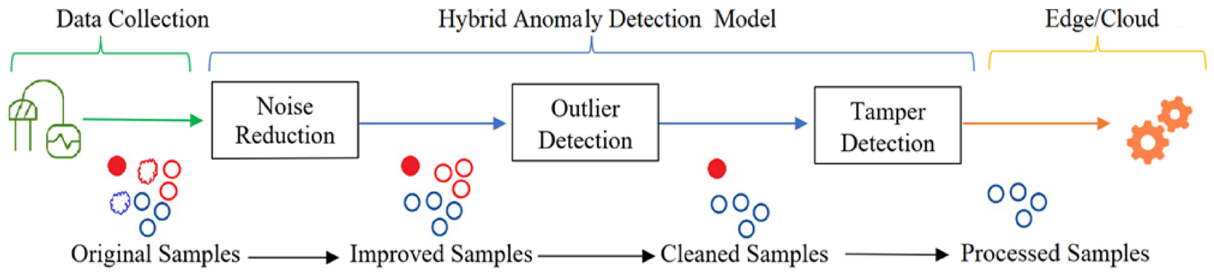


Fig. 2. Hybrid anomaly detection model (→); data flow (→); sending the decision made by the IoT device to the edge/cloud (→); errors ∽/∽; outliers ○; and tampered data ●.

1) Sensors could be damaged or worn by harsh conditions and this can affect their calibration. Therefore, the noise reduction by the smoothing filter removes the noise and drift of the samples.
2) Since the IoT device is placed in this specific environment, the sensor could be relocated to confuse the data acquisition stage. Therefore, the outlier detection stage faces this security flaws in detecting data with different distributions.
3) Whether a malicious user tries to insert erroneous data into the IoT device, the tamper detection stage has previous information on all tamper possibilities to detect them of regular readings and send an alert.

B. Data Collection

We propose a controlled data acquisition environment where data manipulation is doable and the hybrid model can be tested. Therefore, we employ a well-known CO_2, temperature, and humidity sensor `SCD-30`, which is applied in several indoor and outdoor system applications [42]. First, the electronic system has one microcontroller, the SCD30 sensor, and battery, which receives data in normal conditions in an office where people work for around 7 h in groups of two of three people (Label 1). Second, people try to tamper with the sensor by blowing on it (Label 2). Third, people smoke near the sensor to activate the fire alarm (Label 3). Finally, we simulate a fire by putting fireballs in a basket near the sensor (Label 4).

However, because the data needs a couple of hours to get tiny variations and the security of the building can be affected if malicious users try to tamper with the sensor, we create a test box (scaled size of a standard office) to accelerate the data acquisition process. Fig. 3 shows the design with a computer fan to cool the box, an incandescent bulb to warm up the box, and a small door that the user can tamper the sensor directly. To avoid noisy samples for past tests, the test box can be disassembled to restore the settings for new tests. The sample rate for label 1 in offices was 10 min. Then, the trend was calculated to define the same seasonality behavior in the text box, which is to take samples every 5 s. As a result, the data was stored into a matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, where m is the number

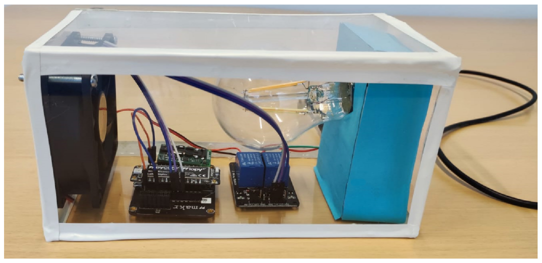


Fig. 3. Test-box designed for data acquisition process: 1) computer fan, 2) light bulb, and 3) electronic circuit.

TABLE II
External Data Sets' Description

Dataset	No. Instances	No. Attributes	Labels
EDS1	26909	11	3
EDS2	243	11	2
EDS3	125	10	5
EDS4	4417	2	2

of samples and n amounts the number of variables (that is to say, the number of attributes representing each sample). Meanwhile, $\boldsymbol{L} \in \mathbb{R}^{m\times 1}$ is a vector holding the sample labeling. In this case, $m = 1500$ and $n = 3$.

C. External Data Sets

External sensors data sets were considered to prove this hybrid model proposed. Therefore, the first data set (EDS1) comprises eight MOX gas sensors and a temperature and humidity sensor to collect data from home activity with two stimuli: 1) wine and 2) banana. Therefore, when the system detects background activity, this is considered a threat situation since users attempt to tamper with sensors by using them without stimuli and get erroneous readings. Then, the second data set (EDS2) detects fire detection in Algeria. It uses ten different sensors to classify fire and nonfire. However, there are erroneous samples when people burn paper or there are strong winds that affect the sensor calibration. As a result, this issue is considered an anomaly behavior. Next, the third data set (EDS3) uses five different QCM gas sensors to recognize five different gas measurements (i.e., 1-octanol, 1-propanol, 2-butanol, 2-propanol, and 1-isobutanol). Thus, isobutanol is a flammable liquid combined gas, reflecting machinery malfunctions. Finally, to compare with a previous hybrid anomaly detection model, the fourth data set (EDS4) is a real data set gained from physical sensors (indoor and outdoor). Two scenarios for each type of sensor (indoor and outdoor) are applied: single-hop and multihop cases. The single-hop case consists of a sensor and head station, plus a router in the multihop case. Also, this data set is used in [18]. Finally, Table II describes the data sets.

D. Noise Reduction

A phenomenon can be described by static data, time series, and images. Therefore, sensors measure the magnitude of a quantity and convert it into an electric signal at a specific sample rate (original samples) [4]. Then, the signal is discretized/digitalized to be understandable to the microcontroller. However, errors like voltage fluctuations, nonlinearity response, and vibrations insert noise into the electric signal, confusing the ML algorithm in the feature extraction stage [43]. For this reason, signal smoothing is a filter that reduces these noise components when the phenomenon does not have high sampling frequencies getting a cleaner signal [44]. Consequently, Moving Average, Mean, Gaussian filter, and Savi–Golay are the most relevant smoothing algorithms [45]. In this hybrid model, the output of this step is called *improved samples*.

E. Outlier Detection

This stage is responsible for detecting data with a different distribution than the rest. This process is carried out through an unsupervised analysis [4]. As a result, a refined data subset called *cleaned samples* is obtained. Outlier detection methods used for this stage related to the literature review are as follows.

1) *The standard deviation method (STD)* quantifies the variation or spread of a numerical data set in which two standard deviations represent 95% of the data. Therefore, any data point outside of this boundary is removed [21].
2) *The LOCAL* measures the local deviation of the density of a given sample concerning its neighbors. It is local in that the anomaly score depends on how isolated the object is from the surrounding neighborhood [23].
3) *ISO* is based on modeling the expected data in such a way as to isolate anomalies that are both few and different in the feature space [22].
4) *Elliptic envelope (ELLI)* detects outliers in a Gaussian distributed data set. It provides the *Contamination* argument that defines the expected ratio of outliers observed in practice [23].
5) *OSVM* captures the density of the majority class and classifies examples on the extremes of the density function as an outlier [24].

F. Tamper Detection

In some scenarios, the information acquired by the IoT device can be compromised by malicious users trying to steal or modify data [27]. Therefore, the threat model works to identify and detects threats regarding defined assumptions and scenarios to avoid security flaws [46]. Consequently, since the IoT device aims to send less data to the Edge/Cloud, and its memory stores one application (i.e., the IoT device does not allow multitenancy), we consider sensors could be tampered with. With this assumption, on the one hand, malicious users can modify the incoming data when the ML model is learning, and its rules could be inaccurate. On the other hand, the IoT device may have the proper model, whereas users try to insert erroneous inputs creating manipulated environments (e.g., activate the fire alarm by putting a lighter near the smoke sensor) [47].

Due to the above-mentioned statements, the tamper-detection stage requires knowing all the manipulation possibilities to mitigate them, which needs to deploy supervised learning [48]. Therefore, the IoT device was also trained when a malicious user could modify the standard conditions of sensors by blowing on them when collecting data (label 2). Besides, a malicious user could attempt to trigger the fire alarm by smoking at the sensor (label 3). As a result, when the IoT device detects a potential threat, it will decide to eliminate the sensor reading (label 2) or send an alert to the Edge/Cloud (label 3). This procedure is the last step of the data flow called *processed samples*. Finally, the algorithms used to reach this goal are presented as follows.

1) *k-nearest neighborhood* associates a new instance with the training base and assigns it to the closest group according to its distance [4].
2) *Naive Bayes* describes the probability of an event happening based on previous knowledge of the event [25].
3) *Support vector machine (SVM)* has kernel functions that put the data in hyperplanes to improve the decision boundaries [21].
4) *Decision tree* is a heuristic model based on rules [18].
5) *Neural network* is a recursive training model [2].

IV. Results

This section evaluates the choice of algorithms listed for each step in Section III and determines a target microcontroller family to deploy the final version of the proposed hybrid model.

A. Experimental Setup

ML models are trained on an external computer [core i7 with 32 GB of random access memory (RAM)] with the previously described data sets. Therefore, regression analysis is used to observe how smoothing algorithms soft the signal in the noise reduction step, which improves samples (according to the data flow presented in Fig. 2). Then, the ANOVA statistical metric determines the improved variability of the samples and how unsupervised models reduce the scatter, obtaining cleaned data. Next, the classification performance determines which classification method is adequate for detecting tampering in the cleaned data to send to the cloud just processed data. Finally, selected algorithms are exported as a `.h` file (i.e., header file referenced by a document written in C) to determine the memory consumption. It is important to mention that the original samples were divided ten times randomly to avoid bias in selected algorithms.

B. Improved Samples

The data-gathering process is affected by noise injection from the abovementioned sources. Therefore, time series analysis provides many techniques to understand a data set better. Perhaps, the most useful of them are Level (i.e., the base value if it were in a straight line), Trend (i.e., the linear increasing or decreasing behavior), Seasonality (i.e., the repeating patterns), and Noise (i.e., the variability of the observations). In this way, Fig. 4 shows these components from the CO_2 variable as an example. Therefore, smoothing filters are required, where noise samples cancel each other out.

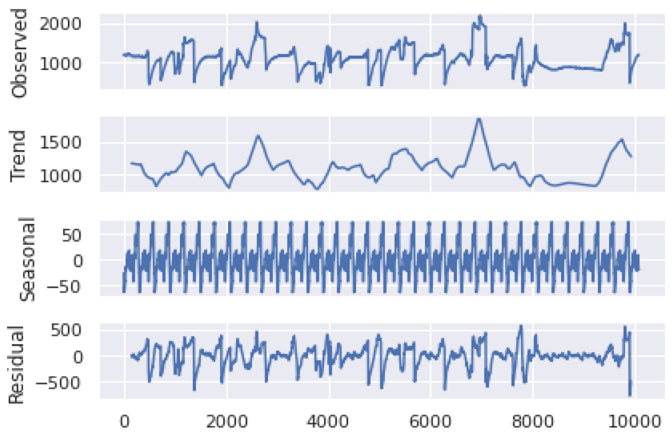


Fig. 4. Main components of CO_2 signal. y-axis: ppm values (0–10 000), x-axis: number of samples.

TABLE III
SMOOTH FILTER STATISTICAL ANALYSIS

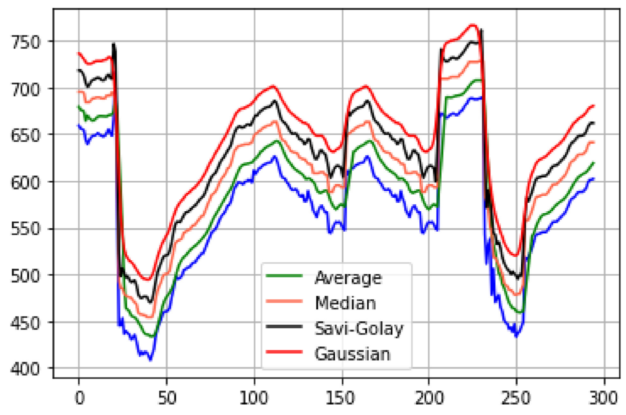
	Average k=30	Median k=30	Savi-Golay sigma=7	Gaussian k=9, poly=4	Stastistic Metrics
CO_2	6.67	5.61	5.87	6.41	**RMSE**
	0.99	0.97	0.99	0.96	**R^2 score**
	8.40	7.99	7.97	8.26	**SNR**
TEMP	0.73	0.04	0.07	0.156	**RMSE**
	0.98	0.99	0.99	0.99	**R^2 score**
	15.37	14.78	14.74	15.17	**SNR**
HUM	1.74	0.14	0.23	0.41	**RMSE**
	0.98	0.97	0.97	0.90	**R^2 score**
	8.75	8.65	8.65	8.72	**SNR**

The above-mentioned smoothing filters are deployed to reduce the data set noise and get improved samples. Hence, regression analysis such as the root-mean-square error (RMSE) determines how close the observed data points are to the filter-smoothed values. In addition, the R2 metric presents the fitness of the filter, and the signal-to-noise ratio (SNR) shows improvement in the quality of the relevant signal. Therefore, the metrics mentioned above were used to select a suitable smoothing filter. As a result, even though the MF has a better RMSE metric for each signal (i.e., minor value), its output is very similar to the signal input and does not reduce noise adequately. Conversely, for the CO_2, temperature, and humidity, the Moving average filter has both better SNR and R^2 metric. Therefore, this filter is selected for scoring higher in the evaluated metrics. Fig. 5 shows graphically how the moving average filter refines the CO_2, and Table III summarizes the values obtained from each metric where k stands for windows size.

Once the average filter proves to be the suitable solution to smooth signals, it is applied to the external data sets. Fig. 6 shows the result of one smoothed variable of each data set.

C. Clean Samples

The data set maintains outliers with different distributions. Therefore, statistical methods can describe the variability of



(—) Original samples (—) Gaussian filter, (—) Median filter, (—) Savi-Golay filter, (—) Average filter

Fig. 5. Smoothing filter applied to CO_2 signal. y-axis: analog-to-digital conversion values (0–1023); x-axis: number of samples.

the entire data set, and ANOVA analysis can be used for this purpose. This analysis splits the data set into systematic factors (i.e., data with normal distribution) and random factors (i.e., outliers). Then, unsupervised ML methods can prune these random factors and increase the decision edges distance between labels. Indeed, these outliers detection methods can reduce the data complexity, and the ML model can reach high throughput.

The mentioned-above outlier detection approaches are applied to the data set to get different improved/pruned subsets (all methods prune data around 15%) and test them with ANOVA analysis. Consequently, the standard deviation and LOCAL algorithms keep the data set scattered, which can cause confusing decision edges between labels. On the other hand, the ISO algorithm concentrates the data due to the central point (mean). Furthermore, ELLI and OSVM have similar results in maintaining data characteristics while reducing scatter. For this reason, we selected the standard deviation, Isolation Forest, and OSVM criteria, representing improvements in reducing data scatter to determine in the next step which of them better eliminates outliers improving the classification algorithm performance. Finally, Fig. 7 shows the variability of each method obtained by ANOVA analysis. Given that OSVM is the suitable solution to remove outliers, this method was applied to the external data sets.

D. Processed Samples

This stage aims to detect whether the sensor has been tampered with. Therefore, it is necessary to detect labels 2 (people blowing on the sensor) and 3 (people smoke near the sensor to active fire alarm). Hence, classification algorithms were trained with clean subsets (obtained in the previous section) and the original samples to determine improvements in the classification decision boundaries.

Table IV shows the classification performance of each applied algorithm. As a result, the original samples demonstrate outliers because ML models have issues classifying incoming data correctly. In addition, despite the subset

TABLE IV
CLASSIFICATION PERFORMANCE FOR EACH ANOMALY DETECTION TECHNIQUE

ML model	Datasets				
	Orig. Samp	Im. Samp	STD	ISO	OSVM
SVM	0.61	0.54	0.70	0.61	0.75
k-NN	0.84	0.86	0.89	0.90	0.93
Naive Bayes	0.78	0.78	0.80	0.80	0.81
Decision Tree	0.88	0.90	0.90	0.96	0.95
Neural Network	0.94	0.95	0.93	0.98	0.98

obtained by the Standard Deviation, reducing outliers has similar classification results to the original samples data set. In contrast, ML models have high classification performance when they are trained with data sets obtained by ISO and OSVM algorithms. Hence, OSVM is adequate to be implemented in the IoT device due to its capacity to reduce outliers significantly and lightweight memory footprint.

The next step was to test the classification algorithms with the improved samples (i.e., Average filter) and cleaned samples (i.e., One-SVM) of the external data sets. Therefore, Table V shows the classification results of each data set. It demonstrates that the SVM algorithm and Naive Bayes struggled to classify between labels. In addition, Decision Trees and Neural networks maintained similar results. Furthermore, using this hybrid approach by smoothing samples and detecting outliers, the chance to find a tamper situation is very close to 100% on each data set. Moreover, for other classification purposes, this model increases by at least ten points of accuracy whether the classifiers were trained with the original samples.

Once the ML models are trained, the decision tree algorithm and neural network show higher classification results. However, in real test scenarios, the decision tree algorithm has strict rules which reduce the ability to recognize light changes in incoming data. Therefore, the neural network model is an adequate solution to detect whether the sensor has been tampered with. Nevertheless, neural networks can achieve close to 100% accuracy in all the evaluated scenarios. However, its complexity (i.e., the number of layers and neurons) is considerably reduced. For example, the optimal neural network should have three hidden layers of 32, 16, and 8 neurons with the original database, respectively. However, with the database obtained by OSVM, only two hidden layers of 24 and 8 neurons are necessary, significantly improving memory consumption with similar accuracy (Table VI). Conversely, when the neural network is trained with pruned samples from the external data set EDS1, which is considered by its complexity and size, it improves the accuracy (from 95% to 98%). Besides, the model complexity reduces by several neurons in hidden layers (Table VII). The model architecture uses a sequential model and dense layers, which means each layer is deeply connected with its preceding layer. In addition, rectifier linear unit (ReLU) is used for sparse activation of neurons, efficient computation, and better gradient propagation. Finally, a dense layer with softmax as an activation function is added. Applying softmax makes the output vector in the interval $I = [0, 1]$ such that the components will add up to 1. Therefore, they can be interpreted as probabilities.

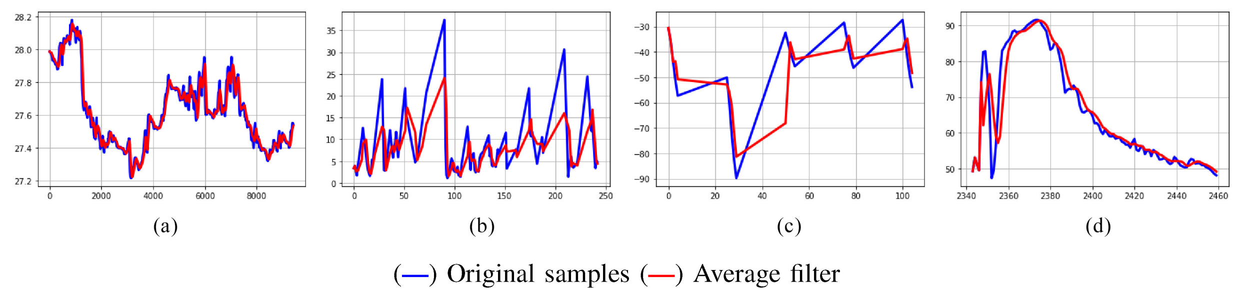


Fig. 6. Smoothed signals from the external data sets. (a) EDS1: temperature. (b) EDS2: fine fuel moisture code (FFMC). (c) EDS3: QCM3 sensor. (d) EDS4: humidity.

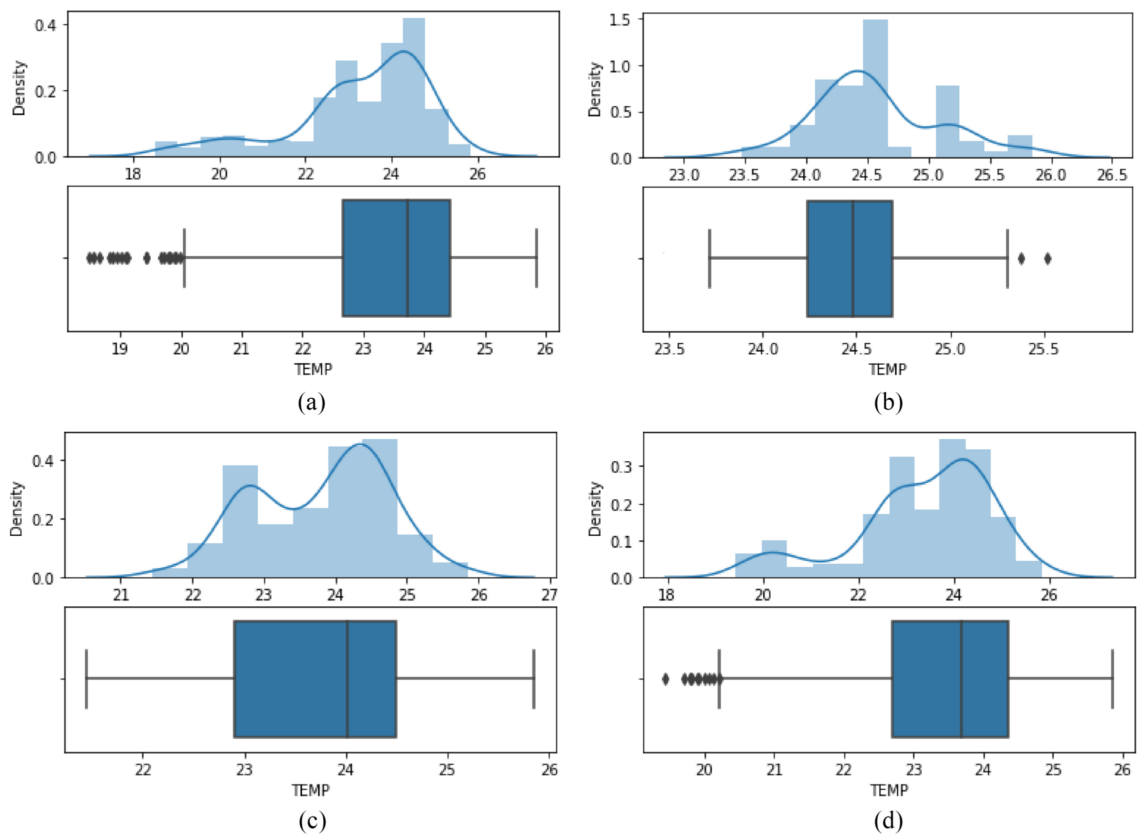


Fig. 7. Variance data distribution by ANOVA analysis for the temperature variable. (a) Original samples. (b) One-SVM. (c) Isolation forest. (d) Standard deviation.

TABLE V
CLASSIFICATION PERFORMANCE FOR EACH ANOMALY DETECTION TECHNIQUE.
ORG: ORIGINAL SAMPLES, IMPR: IMPROVED SAMPLES, AND CLEN: CLEANED SAMPLES

	External datasets											
ML model	**EDS1**			**EDS2**			**EDS3**			**EDS4**		
	Org.	Impr.	Clen	Org.	Impr.	Clen	Org.	Impr.	Clen	Org.	Impr.	Clen
SVM	0.40	0.55	0.6	0.76	0.76	0.76	0.55	0.65	0.66	0.95	0.97	0.97
k-NN	0.88	0.90	0.93	0.97	0.94	0.97	0.97	0.94	0.94	0.98	1.00	0.99
Naive Bayes	0.78	0.82	0.85	0.86	0.87	0.87	0.82	0.85	0.85	.86	0.86	0.86
Decision Tree	0.90	0.95	0.96	0.99	1.00	1.00	0.84	0.92	0.95	1.00	1.00	1.00
Neural Network	0.95	0.95	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

E. Memory Consumption

One essential part of a microcontroller is its memory, which stores information temporarily or permanently. Hence, memory blocks are described as arrays and divided into cells that can store data and be accessed using a unique identifier, representing its address or position relative to the memory array. Consequently, memory blocks can be either volatile or nonvolatile, and the control processing unit "CPU" accesses them concerning the computer architecture; this can be either Von Neumann or Harvard architecture [49]. In this scenario,

TABLE VI
PROPOSED ARCHITECTURE OF THE NEURAL NETWORK USING THE ORIGINAL SAMPLES

Layer (Type)	Output Shape	# Param
Input (Dense)	(None,6)	24
Layer 1 (Dense)	(None,24)	168
Layer 2 (Dense)	(None,8)	200
Output (Dense)	(None,4)	36

TABLE VII
PROPOSED ARCHITECTURE OF THE NEURAL NETWORK USING THE EXTERNAL DATA SET EDS1

Layer (Type)	Output Shape	# Param
Input (Dense)	(None,11)	72
Layer 1 (Dense)	(None,12)	84
Layer 2 (Dense)	(None,8)	104
Output (Dense)	(None,3)	27

the proposed hybrid model is tested in AVR, Tensilica, and ARM microcontrollers. While AVR and Tensilica family microcontrollers are based on the Harvard architecture model, ARM family microcontrollers can be based on either von Neuman or Harvard architectures. These microcontroller families divide memory units into two. First is the Flash memory, in which the firmware is stored to be executed. Second is the RAM, where microcontrollers typically use SRAM memory, which is a type of RAM that uses flip-flops to store one bit of data.

The electronic system deployed uses an Arduino Nano from the **AVR family** with 32 kB in Flash and 2 kB in RAM. An Arduino Nano 33 IoT from **ARM family** with 256 kB in Flash and 32 kB in RAM, and NodeMCU from **Tensilica family** with 4 MB in Flash and 128 kB in RAM. Each level of the hybrid model was compiled and tested to determine the memory consumption.

Tables VIII and IX summarize the memory consumption of each microcontroller. As relevant results, AVR microcontrollers do not have enough memory to compile this application because the Flash needed is 35 kB, and SRAM is fully utilized (2 kB). Besides, the outlier detection model uses half of the RAM. ARM microcontrollers, specifically with microcontrollers M0, M4, and recently M7, are designed to allow ML models with higher computational resources than the AVR family. Arduino Nano IoT uses its Flash around 11 kB in the bootloader, 12 kB in the main program, 2 kB to allocate the smoothing filter, 20 kB for the outlier detection model, and 5 kB for the tamper detection. Furthermore, Flash and SRAM utilization were 56 and 12 kB, respectively. Therefore, Nano IoT has at least 200-kB free memory, and the 20-kB RAM is available. On the other hand, since Tensilica microcontrollers need external libraries to be programmed in the Arduino environment, the bootloader needs 260 kB, which is a high memory requirement. Hence, the Flash memory was used at 310 kB for the hybrid model; it is around 13% of the total Flash memory. Nevertheless, these boards struggle with some WiFi libraries that are inefficiently compiled.

F. Bandwidth Consumption

To determine the impact of performing the anomaly detection on the IoT device instead of the cloud on bandwidth consumption, we calculate the expected bandwidth consumption for each case. First, sending streaming data to the cloud, the payload uses 20 bytes, and the data packet (i.e., control header) is 30 bytes sending data to the cloud each 30 ms by the MQTT protocol, which means the bandwidth is around 1 kb/s. Second, we define sending data to the cloud when the system detects a tamper/threat situation since the outlier detection can be processed locally by its outputs (fan) takes 15 ms. Hence, the same data packet and protocol use just 300 bps (worst scenario) since the frequency of sending messages decreases significantly. Besides, we determine that outliers appear when the IoT device wakes up from sleep modes, even when the main program waits until the sensor responds that is calibrated. Therefore, this anomalous behavior is corrected locally. In addition, data is not compromised because packages sent to the cloud can be just alert messages reducing, even more, the bandwidth needed.

TABLE VIII
FLASH CONSUMPTION ANALYSIS. OD: OWN DATA SET, EDS1: EXTERNAL DATA SET

Block memory	**Microcontrollers**					
	AVR		ARM		Tensilica	
	OD	EDS1	OD	EDS1	OD	EDS1
Boot-loader	450B	450B	11kB	11kB	260kB	256kB
Main program	12kB	15kB	14kB	20kB	20kB	30kB
Noise reduction	1kB	2kB	2kB	2kB	2kB	10kB
Outlier detection	15kB	20kB	30kB	20kB	20kB	30kB
Tamper detection	5kB	5kB	7kB	7kB	7kB	7kB
Total	≈ 33kB	≈ 43kB	≈ 56kB	≈ 70kB	≈ 310kB	≈ 320kB
Free memory	≈ 0B	≈ 0B	≈ 200kB	≈ 185kB	≈ 3690kB	≈ 3680kB

TABLE IX
RAM CONSUMPTION ANALYSIS. OD: OWN DATA SET, EDS1: EXTERNAL DATA SET

Block memory	**Microcontrollers**					
	AVR		ARM		Tensilica	
	OD	EDS1	OD	EDS1	OD	EDS1
Boot-loader	10B	100B	2kB	2kB	27kB	27kB
Main program	400B	600B	3kB	4kB	3kB	4kB
Noise reduction	500B	1kB	1kB	2kB	1kB	2kB
Outlier detection	500B	1kB	2kB	3kB	2kB	5kB
Tamper detection	1.5kB	2kB	3kB	3kB	3kB	3kB
Total	≈ 3kB	≈ 4kB	≈ 11kB	≈ 13kB	≈ 37kB	40kB
Free memory	≈ 0B	≈ 0B	≈ 20kB	≈ 18kB	≈ 25kB	≈ 22kB

V. CONCLUSION AND FUTURE WORKS

In this article, the proposed three-step anomaly detection hybrid model, which integrates data preprocessing (reducing errors), and outlier detection tasks, achieves excellent performance in the data tamper detection scenarios for data collection from IoT devices. Besides, the hybrid model runs on IoT devices to process data locally with a minimal memory footprint. In the first step, to achieve noise reduction, smoothing algorithms demonstrate a suitable solution for canceling voltage fluctuations produced by sensors. In the second step, we demonstrate that it is necessary to implement an outlier detection step to increase the robustness of detecting abnormal distribution in data. Therefore, OSVM prunes outliers and reduces the scatter better. Finally, in the third step, the tamper detection step needs previous knowledge to train models and detect anomalous sensor behavior. Thus, neural networks can recognize those changes in sensor functionalities and detect whether if the work environment is compromised. The anomaly detection hybrid model also proves the new functionalities that emerging microcontrollers may have processing ML models close to the user. This reduces latency with fewer data movements between devices with adequate time response. In addition, processing data locally reduce the opportunity for malicious users to steal data. Furthermore, neural networks can recognize 98% of tamper situations.

This research provides a new perspective on anomaly detection. The proposed hybrid model was evaluated in time-series data from different sensor data sets. In future work, the proposed method will be further expanded and optimized for different sensor data with extensive power consumption analysis.

References

[1] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120043–120065, 2021.

[2] Y. Djenouri, D. Djenouri, A. Belhadi, G. Srivastava, and J. C.-W. Lin, "Emergent deep learning for anomaly detection in Internet of Everything," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3206–3214, Feb. 2023.

[3] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "LSTM learning with Bayesian and Gaussian processing for anomaly detection in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5244–5253, Aug. 2020.

[4] M. Zhang, X. Li, and L. Wang, "An adaptive outlier detection and processing approach towards time series sensor data," *IEEE Access*, vol. 7, pp. 175192–175212, 2019.

[5] T. Yu, X. Wang, and A. Shami, "Recursive principal component analysis-based data outlier detection and sensor data aggregation in IoT systems," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2207–2216, Dec. 2017.

[6] H. Cheng, X. Liu, H. Wang, Y. Fang, M. Wang, and X. Zhao, "SecureAD: A secure video anomaly detection framework on convolutional neural network in edge computing environment," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 1413–1427, Apr.–Jun. 2022.

[7] R. Kaur and J. K. Sandhu, "A study on security attacks in wireless sensor network," in *Proc. Int. Conf. Adv. Comput. Innov. Technol. Eng. (ICACITE)*, 2021, pp. 850–855.

[8] T. Li, C. Sun, S. Li, Z. Wang, X. Chen, and R. Yan, "Explainable graph wavelet denoising network for intelligent fault diagnosis," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 28, 2022, doi: 10.1109/TNNLS.2022.3230458.

[9] H. Shang, J. Wu, C. Sun, J. Liu, X. Chen, and R. Yan, "Global prior transformer network in intelligent borescope inspection for surface damage detection of aero-engine blade," *IEEE Trans. Ind. Informat.*, early access, Nov. 15, 2022, doi: 10.1109/TII.2022.3222300.

[10] H. Shang, C. Sun, J. Liu, X. Chen, and R. Yan, "Deep learning-based borescope image processing for aero-engine blade in-situ damage detection," *Aerosp. Sci. Technol.*, vol. 123, Apr. 2022, Art. no. 107473. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S127096382200147X

[11] R. Zhu et al., "KNN-based approximate outlier detection algorithm over IoT streaming data," *IEEE Access*, vol. 8, pp. 42749–42759, 2020.

[12] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," 2019, *arXiv:1909.09145*.

[13] Y. Liu, Z. Ma, X. Liu, S. Ma, and K. Ren, "Privacy-preserving object detection for medical images with faster R-CNN," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 69–84, 2022.

[14] A. K. Pathak, S. Saguna, K. Mitra, and C. Åhlund, "Anomaly detection using machine learning to discover sensor tampering in IoT systems," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[15] D. C. G. Valadares, N. C. Will, J. Caminha, M. B. Perkusich, A. Perkusich, and K. C. Gorgônio, "Systematic literature review on the use of trusted execution environments to protect cloud/fog-based Internet of Things applications," *IEEE Access*, vol. 9, pp. 80953–80969, 2021.

[16] K. Govindan and P. Mohapatra, "Trust computations and trust dynamics in mobile adhoc networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 279–298, 2nd Quart., 2012.

[17] K. Sood, M. R. Nosouhi, N. Kumar, A. Gaddam, B. Feng, and S. Yu, "Accurate detection of IoT sensor behaviors in legitimate, faulty and compromised scenarios," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 1, pp. 288–300, Jan./Feb. 2023.

[18] N. Alghanmi, R. Alotaibi, and S. M. Buhari, "HLMCC: A hybrid learning anomaly detection model for unlabeled data in Internet of Things," *IEEE Access*, vol. 7, pp. 179492–179504, 2019.

[19] M. S. S. Garmaroodi, F. Farivar, M. S. Haghighi, M. A. Shoorehdeli, and A. Jolfaei, "Detection of anomalies in industrial IoT systems by data mining: Study of CHRIST Osmotron water purification system," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10280–10287, Jul. 2021.

[20] M. Keerthika and D. Shanmugapriya, "Wireless sensor networks: Active and passive attacks—Vulnerabilities and countermeasures," *Global Transitions Proc.*, vol. 2, no. 2, pp. 362–367, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666285X2100073X

[21] M. A. Bhatti, R. Riaz, S. S. Rizvi, S. Shokat, F. Riaz, and S. J. Kwon, "Outlier detection in indoor localization and Internet of Things (IoT) using machine learning," *J. Commun. Netw.*, vol. 22, no. 3, pp. 236–243, 2020.

[22] N. Nesa, T. Ghosh, and I. Banerjee, "Outlier detection in sensed data using statistical learning models for IoT," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2018, pp. 1–6.

[23] M. Awawdeh, T. Faisal, A. Bashir, and A. Sheikh, "Application of outlier detection using re-weighted least squares and R-squared for IoT extracted data," in *Proc. Adv. Sci. Eng. Technol. Int. Conf. (ASET)*, 2019, pp. 1–6.

[24] X. Deng, P. Jiang, X. Peng, and C. Mi, "An intelligent outlier detection method with one class support tucker machine and genetic algorithm toward big sensor data in Internet of Things," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4672–4683, Jun. 2019.

[25] S. Zidi, T. Moulahi, and B. Alaya, "Fault detection in wireless sensor networks through SVM classifier," *IEEE Sensors J.*, vol. 18, no. 1, pp. 340–347, Jan. 2018.

[26] Y. Liu, L. Dang, S. Li, K. Cai, and X. Zuo, "Research progress on models, algorithms, and systems for remote sensing spatial-temporal big data processing," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 5918–5931, 2021, doi: 10.1109/JSTARS.2021.3085893.

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

[28] N. Ghosh, K. Maity, R. Paul, and S. Maity, "Outlier detection in sensor data using machine learning techniques for IoT framework and wireless sensor networks: A brief study," in *Proc. Int. Conf. Appl. Mach. Learn. (ICAML)*, 2019, pp. 187–190.

[29] A. Gaddam, T. Wilkin, M. Angelova, and J. Gaddam, "Detecting sensor faults, anomalies and outliers in the Internet of Things: A survey on the challenges and solutions," *Electronics*, vol. 9, no. 3, p. 511, 2020. [Online]. Available: RAhttps://www.mdpi.com/2079-9292/9/3/511

[30] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for IoT time-series data: A survey," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6481–6494, Jul. 2020.

[31] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 2, pp. 159–170, 2nd Quart., 2010.

[32] Z. Shen and Q. Wang, "Data validation and confidence of self-validating multifunctional sensor," in *Proc. IEEE SENSORS*, 2012, pp. 1–4.

[33] V. Chacko and V. Bharati, "Data validation and sensor life prediction layer on cloud for IoT," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Physical Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, 2017, pp. 906–909.

[34] Z.-M. Wang, G.-H. Song, and C. Gao, "An isolation-based distributed outlier detection framework using nearest neighbor ensembles for wireless sensor networks," *IEEE Access*, vol. 7, pp. 96319–96333, 2019.

[35] K. Zhang, K. Yang, S. Li, D. Jing, and H.-B. Chen, "ANN-based outlier detection for wireless sensor networks in smart buildings," *IEEE Access*, vol. 7, pp. 95987–95997, 2019.

[36] W. Zhang, J. Wang, G. Han, S. Huang, Y. Feng, and L. Shu, "A data set accuracy weighted random forest algorithm for IoT fault detection based on edge computing and blockchain," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2354–2363, Feb. 2021.

[37] Y. Liu et al., "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348–6358, Apr. 2021.

[38] X. Wang et al., "Toward accurate anomaly detection in Industrial Internet of Things using hierarchical federated learning," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7110–7119, May 2022.

[39] J. Park et al., "Distilling on-device intelligence at the network edge," 2019, *arXiv:1908.05895.*

[40] L. Ravaglia, M. Rusci, D. Nadalini, A. Capotondi, F. Conti, and L. Benini, "A TinyML platform for on-device continual learning with quantized latent replays," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 11, no. 4, pp. 789–802, Dec. 2021.

[41] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2833–2849, Dec. 2020.

[42] "SCD30 sensor." Sensiron. Accessed: Jul. 1, 2022. [Online]. Available: https://sensirion.com/products/catalog/SCD30/

[43] X. Li et al., "CAN bus messages abnormal detection using improved SVDD in Internet of Vehicles," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3359–3371, Mar. 2022.

[44] J. Jeong, "Real-time whitening application to two microphone sensors for comb filtering and smoothing," in *Proc. IEEE SENSORS*, 2015, pp. 1–4.

[45] P. Kowalski and R. Smyk, "Review and comparison of smoothing algorithms for one-dimensional data noise reduction," in *Proc. Int. Interdiscip. Ph.D. Workshop (IIPhDW)*, 2018, pp. 277–281.

[46] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2483–2495, Aug. 2018.

[47] V. Vakhter, B. Soysal, P. Schaumont, and U. Guler, "Threat modeling and risk analysis for miniaturized wireless biomedical devices," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13338–13352, Aug. 2022.

[48] J. Shafi and A. Waheed, "K-means clustering analysing abrupt changes in air quality," in *Proc. 4th Int. Conf. Electron., Commun. Aerosp. Technol. (ICECA)*, 2020, pp. 26–30.

[49] A. E. Giakoumis, C. K. Volos, I. N. Stouboulos, H. M. Polatoglou, and I. M. Kyprianidis, "Chaos generator device based on a 32 bit microcontroller embedded system," in *Proc. 7th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, 2018, pp. 1–4.

Paul D. Rosero-Montalvo received the engineering degree in electronics and telecommunications from the Universidad Tecnica del Norte (UTN), Ibarra, Ecuador, in 2013, the master's degree in data management systems from the Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador, in 2018, and the Ph.D. degree from the University of Salamanca, Salamanca, Spain, in November 2020, under the supervision of Prof. V. Lopez-Batista.

He was a Research Assistant Professor with the Applied Science Department, UTN, for seven years. At the same time, he was a part-time Lecturer with the TI Department, Instituto Tecnologico 17 de Julio, Ibarra. He has been a Postdoctoral Fellow with the IT University of Copenhagen, Copenhagen, Denmark, since June 2021. His research focuses on emerging microcontrollers to run machine learning models in decentralized networks.

Zsolt István received the Ph.D. degree from the Systems Group, ETH Zurich, Zürich, Switzerland, in 2018.

He is a Full Professor with the Computer Science Department, Technical University of Darmstadt, Darmstadt, Germany, where he leads the Distributed and Networked Systems Group. Earlier, he held positions with the IT University of Copenhagen, Copenhagen, Denmark, and the IMDEA Software Institute, Madrid, Spain.

Pınar Tözün received the Ph.D. degree from the École polytechnique fédérale de Lausanne, Lausanne, Switzerland, in 2014.

She has been an Associate Professor with the IT University of Copenhagen (ITU), Copenhagen, Denmark, since 2018. Her research focuses on hardware-conscious machine learning, performance characterization of data-intensive systems, and scalability and efficiency of data-intensive systems on modern hardware. Before joining ITU, she was a Research Staff Member with IBM Almaden Research Center, San Jose, CA, USA, for three years, where she was part of the team that led the development of IBM Db2 Event Store.

Dr. Tözün received the ACM SIGMOD Jim Gray Doctoral Dissertation Award Honorable Mention in 2016 for her Ph.D. thesis.

Wilmar Hernandez (Senior Member, IEEE) received the bachelor's degree in electronics engineering and the specialist degree in microelectronics from the Instituto Superior Politecnico Jose Antonio Echeverria (ISPJAE), Havana, Cuba, in 1992 and 1994, respectively, and the M.S. degree in signal treatment and the Ph.D. degree in electronic engineering from Enginyeria La Salle, Universitat Ramon Llull, Barcelona, Spain, in 1997 and 1999, respectively.

From 1992 to 1995, he was a Lecturer with the Electrical Engineering Faculty, ISPJAE, and a Researcher with the Microelectronics Research Center, ISPJAE. From 1999 to 2003, he was with the Department of Electronics and Instrumentation, University Institute for Automobile Research, Universidad Politecnica de Madrid (UPM), Madrid, Spain, where he was the Technical Director of the Department of Electronics and Instrumentation from January 2003 to January 2004. From January 2004 to March 2013, he was an Associate Professor of Circuits and Systems with the Department of Circuits and Systems, EUIT de Telecomunicacion, UPM. From September 2014 to September 2015, he was a Researcher with SENESCYT, Azogues, Ecuador, under the Prometeo Fellowship Program. From December 2015 to November 2017, he was a Professor with the Universidad Tecnica Particular de Loja, Loja, Ecuador. Since January 2018, he has been a Professor with the Universidad de Las Americas, Quito, Ecuador.