Measurement 164 (2020) 108042

Contents lists available at ScienceDirect

Measurement

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

Detecting abnormal sensors via machine learning: An IoT farming WSN-based architecture case study

Paulo Silas Severo de Souza [a,*], Felipe Pfeifer Rubin [a], Rumenigue Hohemberger [c], Tiago Coelho Ferreto [a], Arthur Francisco Lorenzon [b], Marcelo Caggiani Luizelli [b], Fábio Diniz Rossi [c]

[a] Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil
[b] Federal University of Pampa, Alegrete, Brazil
[c] Federal Institute Farroupilha, Alegrete, Brazil

ARTICLE INFO

Article history:
Received 23 January 2020
Received in revised form 21 May 2020
Accepted 25 May 2020
Available online 20 June 2020

Keywords:
Anomaly detection algorithms
Outlier detection
Wireless sensor network

ABSTRACT

Precision Agriculture is a broad, systemic, and multidisciplinary subject, dealing with an integrated information and technology management system, based on the concepts that the variability of space and time influence crop yields. Precision farming aims at more comprehensive management of the agricultural production system as a whole. It uses a set of tools, instruments, and sensors to measure or detect parameters or targets of interest in the agroecosystem. Sensors are distributed in the environment and are usually communicated through a Wireless Sensor Network (WSN). Due to this dispersion of the sensors, errors could occur in Byzantine form or could be caused by safety factors, which can lead to a misinterpretation by the system of data analysis and actuation over the environment. Anomaly detection algorithms can detect such problem sensors by allowing them to be replaced, or the wrong data is ignored. Therefore, this work presents a reference architecture and a heuristic algorithm that aid the decision of which anomaly detection to use based on the demands of agricultural environments. We performed a preliminary evaluation, analyzing different anomaly detection algorithms regarding execution time, accuracy, and scalability metrics. Results show that the decision-making supported by the proposed architecture reduces edge devices' power consumption by 18.59% while minimizing the device's temperature in up to 15.94% depending on the application workload and edge device characteristics.

1. Introduction

The current moment of agriculture demands fast, accurate information that uses real and reliable references. However, these data are not always easily accessible, and for this arise technologies that transform the properties of the environment into physical quantities, and can transmit them in the form of information useful to the producer. One such device is precision farming sensors, devices capable of detecting, reading, and recording a series of changes that can quickly be translated by people or computers [1].

Wireless Sensor Networks (WSNs) is a technology that allows practicing Precision Agriculture (PA) with low costs. Before PA, farmers were only able to make use of imagery or other map-based systems to precisely target their growing areas. PA offers the benefit of real-time feedback through a variety of variables of rural property and crops practiced. Thus, PA gives detailed information, not only on the size of the planted area monitored, but also on the applied amounts of fertilizer, water, and so on [2].

Based on the above, it is possible to identify and monitor from a plant to an entire area of hundreds of square meters. In this way, data collection, monitoring, and application of crop inputs, result in lower costs and higher yields while lowering the effects on the environment. The WSN used in agriculture is very similar to those used in industrial controls. The required components of a WSN ecosystem are a central control unit with a user interface, power elements, communication gateways, router and, most importantly, sensors.

However, since these sensors are sparsely scattered around the environment, they may suffer some Byzantine failure and send erroneous values that can directly influence production [3] [4]. In soil moisture control environments, the water blade can be changed; in temperature control environments, the environment's temperature can be set to a degree that is not ideal. If any of these errors occur, all products can be lost.

* Corresponding author.
E-mail address: paulo.severo@edu.pucrs.br (P.S.S. de Souza).

https://doi.org/10.1016/j.measurement.2020.108042
0263-2241/

In order to detect these abnormal behaviors, anomaly detection algorithms based on machine learning techniques can be used for identifying failed sensors [5]. Through the use of these algorithms, a data analysis service can discard the data gathered from faulty sensors and also mark them for further analysis and replacement [6]. Although several of these algorithms are proposed in the literature, selecting which to use depends on the data to be processed [7].

Regardless of the nature of the data coming from a WSN in an agricultural environment, the computational architecture is very homogeneous. In general, only one sensor's data is sent during a certain period, allowing us to evaluate several algorithms on the same agronomic data set and to infer a better algorithm for detecting anomalies for the most diverse applications which exhibit similar behavior. Accordingly, we propose a conceptual architecture for evaluating, selecting, and tuning anomaly detection algorithms based on the varied demands of agricultural IoT applications.

The rest of the article is presented as follows. Section 2 presents a description of the problem solved in this paper. Section 3 describes an architecture that evaluates and selects the anomaly detection algorithms that deliver the best performance based on specific demands of agricultural IoT applications. Section 4 states the test scenarios, results achieved, and discussion of results. Section 5 presents the related work. Section 6 provides our conclusions and directions for future work.

2. Related work

Ukil et al. [8] present challenges and the relevance of anomaly detection techniques in several IoT health-care scenarios. The authors argue that anomaly detection plays an important role in IoT health-care scenarios. Abnormal data coming from compromised sensors could heavily affect the effectiveness of health-care IoT systems or even put its users lives in danger.

Narayanan, Mittal, and Joshi [9] developed OBDSecure Alert, an anomaly detection mechanism that detects abnormal activities in new and 'on-road' vehicles. The data collected from different vehicle components is converted into a sequence of observation vectors. Every time a new observation is available, a sliding window containing previous observations moves, and the posterior probability of the new observation sequence is determined. If the estimated probability is below a threshold, the observation is perceived as an anomalous state.

On Trilles et al. [10], an environmental anomaly detection system is proposed. The system consists of three layers: content, services, and application. The content layer consists of multiple field sensors that are used to obtain air quality data. The sensors' data are sent to the services layer where anomalies may be detected separately on each data stream using the CUSUM algorithm. The application layer provides a web interface were each sensor status is displayed along with any anomalous warning.

Due to the popularization of areas such as machine learning, several algorithms to detect anomalous sensors data have been proposed. However, choosing the wrong algorithm may delay or even compromise the detection process. Some studies also compare anomaly detection algorithms [11] [12]. However, their analysis does not cover important aspects such as the impact of running anomaly detection algorithms in edge devices regarding temperature and power consumption. Both of which are relevant since edge devices usually operate in strict conditions.

This work complements these related studies as it provides a reference architecture for evaluating and refining the performance of anomaly detection algorithms according to the needs of agricultural IoT applications. Besides, our proposal has a generalist design which allows the addition of anomaly detection algorithms with minimum effort.

3. Problem formulation

The massive growth of devices available in the consumer market capable of network communications has brought about the Internet of Things (IoT). The Internet of Things regards any object (thing) which can interact with another through compatible networking capabilities. This interaction is one of the pillars of the IoT realization [13]. Through this communication medium, different devices are able to generate and exchange data that they would not be able to otherwise.

IoT devices are designed for specific goals to reduce manufacturing costs, energy requirements, and others, in contrast to general-purpose solutions. Due to such a particular design, these devices have particular constraints applied to their resources.

Common IoT solutions for agriculture such as livestock monitoring and soil analysis, harvest data from sensors dispersed in the environment, relaying the obtained information to another device or cloud infrastructure. This data must be consistent, correct, and nonetheless informative, otherwise harvesting it would be meaningless.

A possible approach for data validation is sending it to cloud servers as they are capable of much more processing power, albeit transmission latency might rise as problem [14]. On the other hand, processing all the obtained data on an IoT device using common filtering methods might not only be unfeasible due to limited resources but also interfere with the device's main task.

Considering this scenario, in order to properly validate the obtained data, more advanced methods must be used. Anomaly detection algorithms are able to detect anomalous behavior on data sets by applying different machine learning and pattern matching techniques.

A piece of data is considered to be anomalous when its behavior differs from others, i.e., commonly observed patterns are not present in it [15]. These anomalies must be removed from the data set they reside, otherwise any attempt to analyze the information within such set would be affected.

An important consideration that should be taken before employing anomaly detection algorithms is to have a deep understanding of the IoT system it will be employed on. Most of these algorithms were not created specifically for the IoT, but for cloud computing, whereas vast amounts of resources can be provided. Thus, the selected anomaly detection algorithm must be cable of being implemented and executed on a highly constrained device.

A WSN maintains vast amounts of sensors connected, usually to a central server, via a wireless communication channel. Due to various causes, which may be environmental, electrical, or even sabotage, sensors can send erroneous values to the management system, and thereby affect decision making over the entire environment.

In this work, we consider a set of sensors $S = \{s_i : i = 1, \ldots, n\}$ where the distribution of the measurements is homogeneous (all of them performed at the same time). Based on this, all sensors are synchronized, and at each new measurement time interval of t, each sensor measures a characteristic vector x_t^i from the environment.

Each of these vectors of environmental characteristics present measurement values vec_{xj}^i, where $x_t^i = \{vec_{xj}^i : j = 1, \ldots, r\}$ and $x_t^i \in \Re^r$. After a set of moments t with several measurement sm, each of the sensors s_i has already collected a set of characteristics of the environment $D_i = \{x_t^i : t = 1, \ldots, sm\}$.

Anomaly detection techniques intend to detect a sensor presenting anomalous measurements. In other words, sensors pre-

senting a value or subset of measured values that are inconsistent with the rest of the data set. Therefore, we intend to find an anomalous measurement $am \subset D_i$ within an entire set of measurements $D = \cup_{i=1,\ldots,n} D_i$.

4. Proposed architecture

Due to the granularity of data generated by agricultural IoT applications, a wide variety of anomaly detection algorithms can be used to perform pre-processing.

While many algorithms employ similar machine learning techniques, the choice must be based on how the data is organized in terms of variables (multivariate or univariate), analysis model (online, offline, or both) and dimensionality (spatial, temporal or spatiotemporal). Besides, nowadays, there are several classes of anomaly detection algorithms that implement different detection strategies, and as a consequence, they may have different levels of accuracy and resource usage.

Therefore, in this paper, we present an architecture that envisions the diverse needs of agricultural IoT environments, and infer, within a list of anomaly detection algorithms, the one that best satisfies the requirements of each IoT application on the environment in terms of power consumption, heat generated onto the embedded device, accuracy, and execution time. Fig. 1 depicts the proposed architecture.

Efficiently detecting abnormal behaviors of IoT applications on-the-fly is not a trivial problem since these applications typically present highly variable profiles. IoT applications' behavior can change dramatically throughout short periods and in different ways, not only in volume but also regarding the content it holds. Therefore, deciding which technique to use for processing data from an IoT application should not be a permanent decision. Our architecture seeks to ensure that data is processed efficiently despite the highly variable nature of IoT applications.

Once measurements are received, our architecture consults the Knowledge Base component, which is responsible for centralizing the information regarding which algorithm leads to the best results for processing the data of a given application. If this application was not previously before, the sensor's data is forwarded to the Benchmark module, which evaluates a set of anomaly detection algorithms to identify which approach achieves the finest results in accordance with user-defined goals (e.g., energy efficiency or accuracy).

Once the evaluation is over, the Benchmarking module updates the Knowledge Base with the up-to-date information about the fittest algorithm for processing the IoT application's data. After, the Anomaly Detection module is triggered, and the resulting analysis is returned as the final product of our architecture.

To ensure the Knowledge Base stays up-to-date regarding which algorithm delivers the best results while detecting abnormal behavior for applications, all evaluations carried by the Benchmark module hold an α attribute that represents its expiration date. From time to time, when the decision-making is triggered, the architecture checks if the period from the last time the benchmarking for the application was triggered (which is represented by Δb) is higher than the α variable. As this expiration date is passed, the Benchmark module is triggered once again to verify which algorithm should be selected for processing data from the same application in the future. Since benchmarking all anomaly detection algorithms from the database may require a considerable amount of processing, operators may choose for longer benchmarking intervals by giving the α variable a higher value.

Given the strict amount of resources available for Edge devices and the potential overhead of benchmarking anomaly detection algorithms too often, our architecture introduces the Hyperparameter Optimization module, which is called from time to time, within each benchmarking execution. Through its use, the results from anomaly detection algorithms can be enhanced at a lower computation cost compared to picking other algorithms. However, since the need for refining the parameters of each anomaly detection algorithm may change regarding the IoT application workload, we define a δ variable (which is meant to get a value between 0 and 1) that dictates how often the Hyperparameter Optimization module is called. More specifically, the architecture generates a random value between 0 and 1 and only triggers the hyperparameter tuning task if this value is higher than the δ variable.

5. Evaluation and discussion

We did perform an empirical evaluation of the proposed architecture using six unsupervised anomaly detection algorithms with support to the analysis of multivariate data sets [16]. An overview of these algorithms is presented next and a summary categorizing each algorithm is provided in Table 1. Details on the evaluation and the data set are provided further.

Isolation Forest (iForest) [17]. Considers that anomalies are less frequent than normal observations and present different values. Therefore, uses random trees to detect anomalies based on the premise that after constructing isolation trees for a given data set, anomalies are isolated closer to the root of the tree while normal points are isolated in the further nodes.

K-Nearest Neighbors (kNN) [18]. Computes for each point of a data set the distance from its K^{th} nearest neighbor. Then, it selects the top N points with the maximum distances as outliers. In order to minimize the computational cost, a clustering algorithm is used to partition in disjoint subsets the input data. Partitions that cannot contain outliers are pruned. On the remaining partitions, outliers are computed.

Local Outlier Factor (LOF) [19]. Applies the concept of local density to determine if a point is an outlier. The local density of a point is calculated using the distance of its k-nearest neighbors. A point is considered an outlier if, by comparison, it has a lower density than its neighbors.

FindCBLOF [20]. Uses the Squeezer algorithm [21] on a given data set to obtain a set of clusters. Using the obtained clusters and two numerical parameters that are used to define a boundary, two new sets of clusters are derived: LC (Large Clusters) and SC (Small Clusters). For each record of the data set, if the record's cluster belongs to LC, the CBLOF value is calculated using the distance between the record and its cluster, otherwise (its cluster belong to SC) the distance is calculated using the minimal distance between the record and a cluster belonging to LC.

The Histogram-Based Outlier Score (HBOS) [22]. Firstly creates a univariate histogram for each feature on the data set. The histogram is created using different techniques considering the type of data that was provided (categorical or numerical). Then it normalizes the maximum height of each histogram to 1.0 which ensures an equal weight of each feature to the outlier score. Finally, the HBOS value (outlier score) of each instance of the data set is calculated using a formula that may be interpreted as the inverse of a discrete Naive Bayes probability model.

Angle-Based Outlier Detector (ABOD) [23]. Considers the variance of the angles between the different vectors of data objects as a property to measure if an object is an outlier. If the spectrum of observed angles for a point is broad, the point is surrounded by others and is probably located inside a cluster. Otherwise, it is believed to be outside of grouped sets of points, thus it is considered an outlier.

For our evaluation, we selected the ForestCover [24] data set, which is a well-known multivariate data set that contains forest

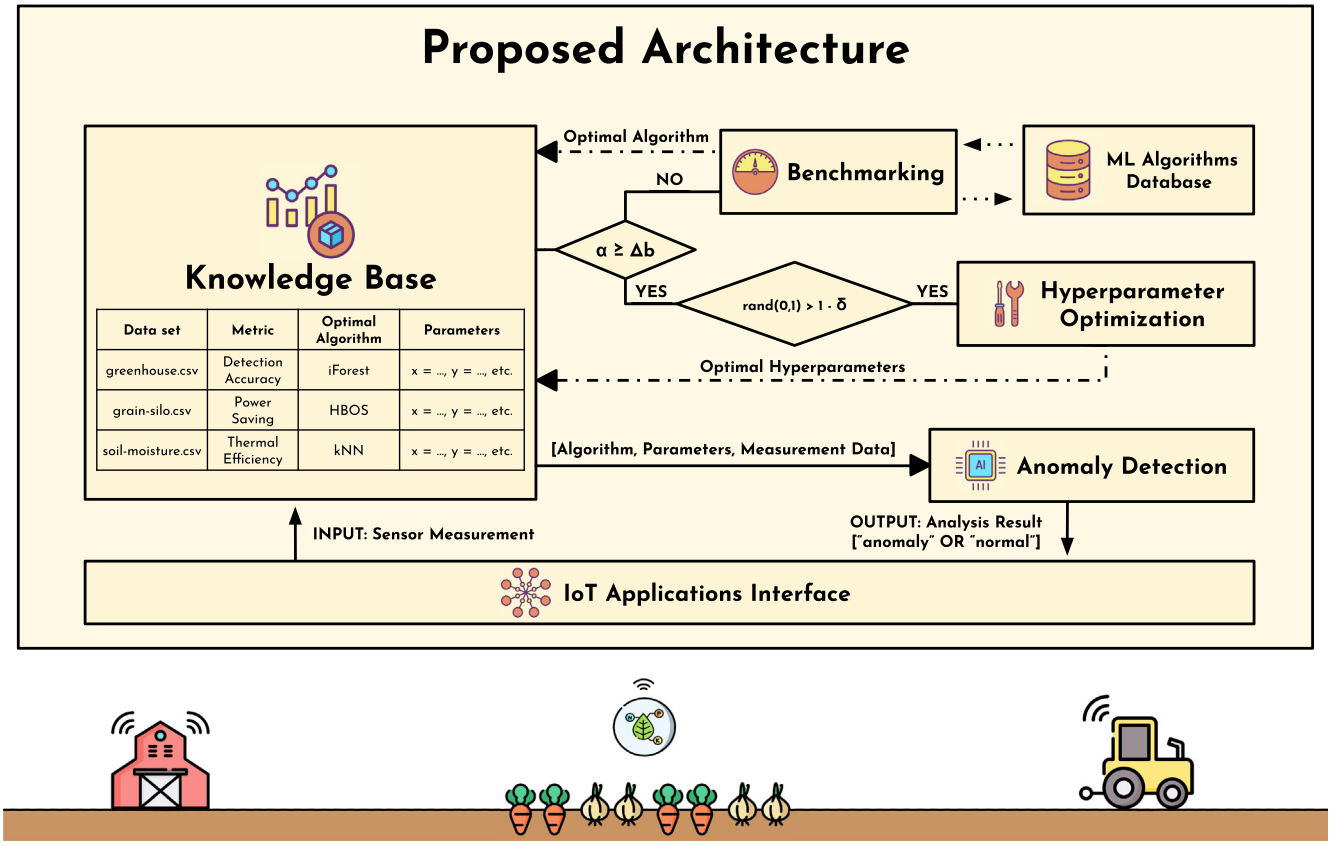

Fig. 1. Proposed architecture for detecting anomalous instances in WSN-based agriculture scenarios.

Table 1
Details on time complexity and detection approach adopted by the anomaly detection algorithms used in the preliminary evaluation of the proposed architecture.

Algorithm	Type	Time Complexity
Angle-Based Outlier Detector	Probabilistic	$O(n^3)$
Local Outlier Factor	Proximity-Based	$O(n^2 \times k)$
K-Nearest Neighbors	Proximity-Based	$O(n^2)$
Isolation Forest	Outlier Ensembles	$O(n)$
Cluster Based Local Outlier Factor	Proximity-Based	$O(n)$
Histogram-Based Outlier Score	Proximity-Based	$O(n)$

Table 2
Technical specifications of the NVIDIA Jetson TX2 Developer Kit.

Components	Specifications
Processors	HMP Dual Core Denver 2/2MP L2 Quad Core ARM A57/2 MB L2
GPU	NVIDIA PASCAL 256 CUDA Cores
Memory	8 GB 128bit LPDDR4 59.7 GB/s
Networking	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth
Operating System	Linux Ubuntu 16.04.5 LTS (kernel v4.4.38-tegra)

cover type from cartographic variables from four areas with minimal human-caused disturbances of the Roosevelt National Forest (United States). Its data is useful for researchers interested in aspects such as biodiversity. Regarding this data set composition, it contains 10 quantitative variables and 286048 data points, from which 2747 (0.9%) are considered outliers (anomalies).

Our experimental edge computing platform comprises an NVIDIA Jetson TX2 Developer Kit[1], a power-efficient embedded AI computing device suitable for several IoT applications such as robots and drones [25]. Besides, Jetson TX2 has thermal and power consumption sensors that make it suitable for our experimentation. Table 2 presents the specifications of its hardware components.

This embedded platform comprises two processors (Denver 2 and Cortex-A57) and a Pascal GPU that hold a set of functionalities to suit general-purpose tasks as well as to handle specific demands such as deep learning applications. This architecture is depicted in Fig. 2. The GPU shares the main memory with the other components instead of having its own dedicated memory to keep the

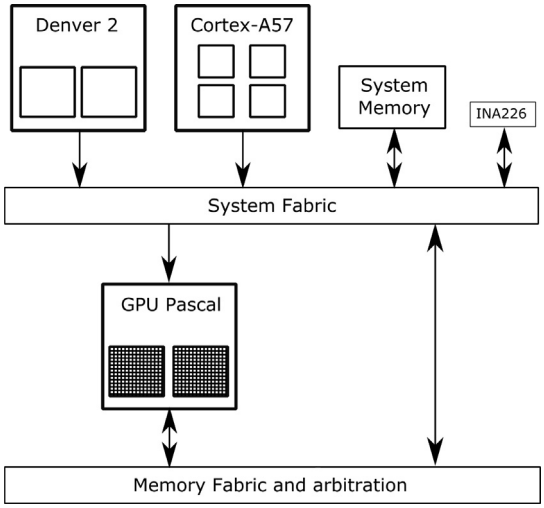

Fig. 2. NVIDIA Jetson TX2 Architecture.

[1] https://developer.nvidia.com/embedded/jetson-tx2>.

architecture as lightweight as possible. Even with this constraint, the NVIDIA Jetson TX2 is capable of leveraging the possibilities of extending AI capabilities from the cloud to the edge.

The results presented in Fig. 3 are the average of 5 executions of each algorithm with a standard deviation lower than 3%. The assets of our experimentation are available in our GitHub repository[2]. We also monitored CPU and memory usage during the execution of the algorithms. Besides, we used the Pearson Correlation Coefficient to verify how resource usage impacted the performance, power consumption, temperature and EDP. Table 3 summarizes the correlations among each of these metrics.

5.1. Precision @ Rank N

We considered Precision at rank n ($P@n$) as one of the indicators for accuracy. This metric is given by Eq. 1 and shows the number of relevant instances among the retrieved results. The algorithm that presented the best accuracy results was Isolation Forest, with $P@n = 0.11$. This result shows the effectiveness of Isolation Forest in handling problems widely discussed in anomaly detection landscape such as Swamping and Masking effects [17]. Swamping consists of identifying normal instances as anomalies. It usually occurs when normal points are too close to the anomalous ones. Masking refers to considering anomalous instances as normal data and often occurs when there are too many anomalies.

$$Precision = \frac{|\text{Relevant Data} \cap \text{Retrieved Data}|}{|\text{Retrieved Data}|} \quad (1)$$

Unlike other algorithms, such as Local Outlier Factor that presented the worst accuracy result, ($P@n = 0.56$), Isolation Forest focuses on identifying anomalies rather than profiling normal points. Since anomalies are less frequent and different from normal observations, Isolation Forest randomly partitions data in decision trees (called isolation trees), so anomalies can be identified as the instances closer to the root of the tree since fewer splits are necessary to isolate them. Fig. 3(a) presents the accuracy results of the evaluated algorithms.

5.2. Area under the ROC curve

When evaluating the performance of classification models, ROC curves can be used for plotting models' TPRs (true positive rates) against FPRs (false positive rates), at different thresholds. It provides an implicit view on the classification's trade-off between sensitivity and specificity.

Selecting a higher threshold increases specificity, leading to fewer false positives; however, it also decreases sensitivity, resulting in fewer true positives. Lowering the threshold increases sensitivity and also true positives, but reduces specificity, which increases false positives. The Area under the ROC curve (AUC) measures the model's classification performance across all possible thresholds. The AUC can also be seen as the model's probability of classifying a random positive sample higher than a random negative sample.

The AUC measurement is extremely significant for comparing the performance of different anomaly detection algorithms. Since data gathered from different sensors might not contain any anomalies, the higher an algorithm can rank these anomalies from normal values, fewer false-positive detections will occur. This is the reason iForest has the highest AUC among the selected algorithms (Fig. 3(b). Instead of focusing on detecting normal instances, iForest detects the anomalies.

[2] https://github.com/paulosevero/outlier_detec_comp

5.3. Execution time

According to the results illustrated in Fig. 3(c), the algorithm that presented the best execution time results was HBOS, being able to detect the outliers in 3.83 s. As presented in Table 1, this algorithm has a linear time complexity $O(n)$, which we believe was the main reason for such a result.

It is worth mentioning that CBLOF and iForest also have linear time complexity, but as stated by Goldstein and Dengel [22], HBOS can detect outliers in large multivariate data sets faster than other algorithms by computing a histogram to each feature of the data set, scoring them individually, and combining them in the end.

On the other hand, ABOD presented the worst execution time due to its cubic time complexity $O(n^3)$, which is caused since several iterations are required for computing the anomaly score of each instance in the data set.

5.4. Power consumption

Regarding power consumption (Fig. 3(d)), Histogram-based Outlier Score was the algorithm with better results, consuming 170.91 mW. Considering the high correlation between resources usage and power consumption presented in Table 3, we can see that the energy-saving achieved by HBOS was influenced by its low CPU (71.04%) and memory (1144.08 MB) demand. On the other hand, Isolation Forest, that presented the worst power consumption result, was the algorithm that most impacted in the device's memory usage (1386.98 MB).

5.5. Temperature

As resources usage affects the device's temperature, we realized that CPU usage impacted the device's heat during the algorithm's execution ($Correlation = 0.82$, as shown in Table 3). Besides, we found a high correlation between execution time and temperature, i.e., the longer the algorithms took to execute, the higher was their impact on the device's heat since the evaluated algorithms involve computing-intensive tasks.

As a consequence of such correlations, the algorithm that presented less impact on device's temperature was CBLOF, that uses the Squeezer algorithm to define the number of clusters without imposing bottleneck with IO operations and requires only one scan over the data set for computing the anomaly score of instances. On the other hand, ABOD was the algorithm that most generated heat to the embedded device due to its higher execution time. All the results regarding temperature are shown in Fig. 3 (e).

5.6. Energy-delay product

One of the well-known strategies to prolong the battery life of embedded systems is through power-saving modes, that focus on minimizing the device's power consumption at the cost of increasing the application's execution time. In this context, there is a concern regarding the balance between performance and power saving [26].

Therefore, in our analysis we also considered the Energy-Delay Product (EDP), which is given by *Performance* $\times$ *Power Consumption*, to evaluate the impact of the chosen algorithms to this trade-off.

The results showed that the execution time was the metric that most impacted in the EDP results, since HBOS, which was the algorithm that demanded less time to process data, achieved the best results, and ABOD, that presented the poorest performance, was the algorithm with the worst EDP index. Besides, as shown in Fig. 3 (f), the EDP results presented a high correlation with the heat

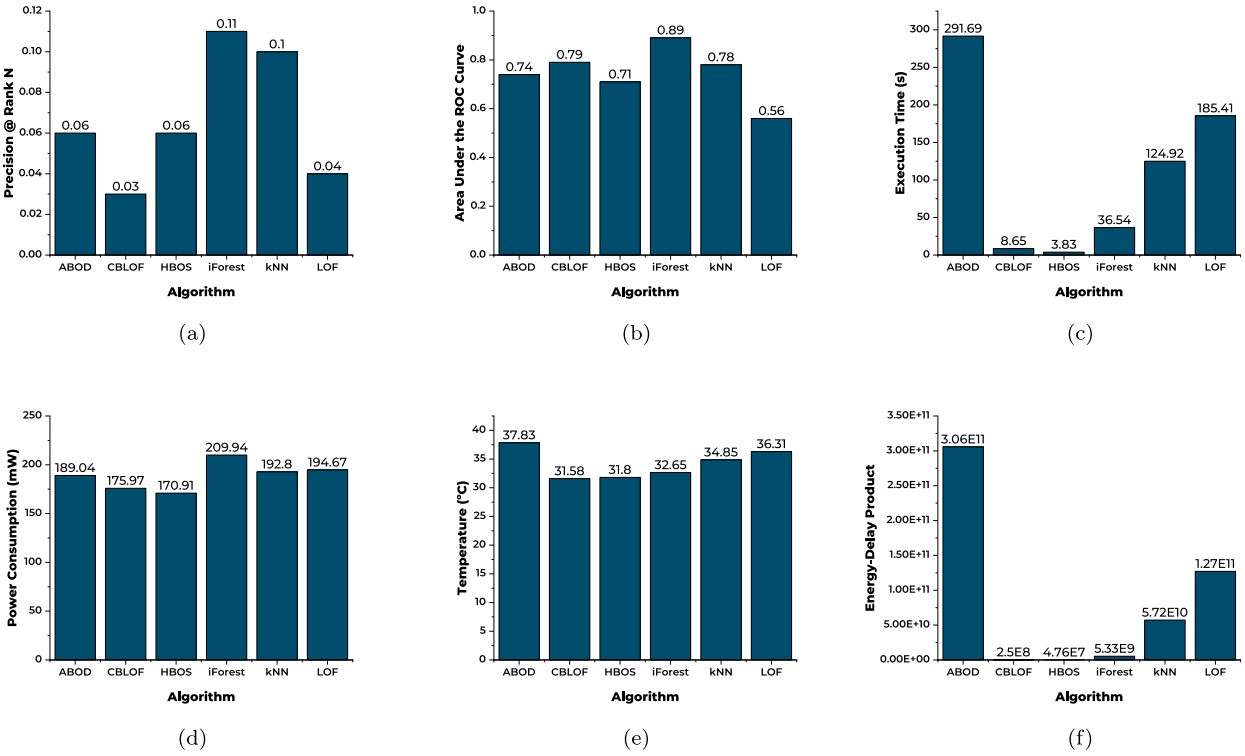

Fig. 3. Evaluation of six anomaly detection algorithms running on an embedded device considering multiple metrics of interest.

Table 3
Correlation among resources usage (CPU and memory) and the evaluated metrics.

Metrics	Pearson Correlation Coefficient
Temperature and Execution Time	0.99
Temperature and EDP	0.92
RAM Usage and Power Consumption	0.87
CPU Usage and Temperature	0.82
CPU Usage and Execution Time	0.77
CPU Usage and Power Consumption	0.75
Temperature and Power Consumption	0.33

imposed to the device, which showed that the more the algorithms manages to balance the trade-off between performance (execution time) and power saving, the better they can achieve thermal-efficiency.

6. Final considerations

A core aspect of IoT is the communication between devices. Although decentralizing processing has many benefits, such as scalability, it also has its drawbacks. Relying solely on external information requires high control of the exchanged data. The data transmitted between devices might end up being anomalous, either due to transmission issues, failed sensors, or even by malicious attempts to affect the receiving end. As a consequence, anomaly detection algorithms are steadily gaining field by allowing the identification of abnormal data.

As changes in the environment drive the behavior of IoT applications, data patterns from IoT sensors might change abruptly, within a short period of time, especially in agricultural scenarios, wherein sensors are exposed to weather conditions which may impose very restricting connectivity constraints. As a consequence, an anomaly detection algorithm, proven as optimal in the morning, may present poor results in the afternoon. Therefore, the decision on which algorithm to employ for IoT data analysis should not be definitive.

Accordingly, in this paper, we present an extensible conceptual architecture that allows the self-adaptive selection and tuning of anomaly detection algorithms based on user-defined goals such as accuracy, power saving, or thermal efficiency. Experiments were conducted, and the results show that timely choosing proper algorithms for anomaly detection in IoT applications improve the analysis accuracy by 72.73% and minimizes device's power consumption and temperature by 18.59% and 15.94% respectively.

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.

References

[1] J. Muangprathub, N. Boonnam, S. Kajornkasirat, N. Lekbangpong, A. Wanichsombat, P. Nillaor, Iot and agriculture data analysis for smart farm, Comput. Electron. Agric. 156 (2019) 467–474.
[2] P. Rekha, K. Sumathi, S. Samyuktha, A. Saranya, G. Tharunya, R. Prabha, Sensor based waste water monitoring for agriculture using iot, in: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2020, pp. 436–439.
[3] M. Xie, S. Han, B. Tian, S. Parvin, Anomaly detection in wireless sensor networks: A survey, J. Netw. Comput. Appl. 34 (2011) 1302–1325.
[4] R. Chamarajnagar, A. Ashok, Integrity threat identification for distributed iot in precision agriculture, in: 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), IEEE, 2019, pp. 1–9.
[5] A. Kamilaris, F.X. Prenafeta-Boldú, Deep learning in agriculture: A survey, Comput. Electron. Agric. 147 (2018) 70–90.
[6] S. Rajasegarar, C. Leckie, M. Palaniswami, Anomaly detection in wireless sensor networks, IEEE Wirel. Commun. 15 (2008) 34–40.
[7] H. Arai, K. Emura, T. Hayashi, A framework of privacy preserving anomaly detection, in: Proc. 2017 Work. Priv. Electron. Soc. – WPES '17, volume 2017-January, ACM Press, New York, New York, USA, 2017, pp. 111–122. doi:10.1145/3139550.3139551.
[8] 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), 2016, pp. 994–997, https://doi.org/10.1109/AINA.2016.158.

[9] S.N. Narayanan, S. Mittal, A. Joshi, Obd_securealert: An anomaly detection system for vehicles, in: 2016 IEEE International Conference on Smart Computing (SMARTCOMP), IEEE, 2016, pp. 1–6.
[10] 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. Dig. Earth 10 (2017) 103–120.
[11] V. Garcia-Font, C. Garrigues, H. Rifà-Pous, A comparative study of anomaly detection techniques for smart city wireless sensor networks, Sensors 16 (2016) 868.
[12] P.S.S. de Souza, W. dos Santos Marques, F.D. Rossi, G. da Cunha Rodrigues, R.N. Calheiros, Performance and accuracy trade-off analysis of techniques for anomaly detection in iot sensors, in: 2017 International Conference on Information Networking (ICOIN), IEEE, 2017, pp. 486–491.
[13] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. 54 (2010) 2787–2805.
[14] 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 (2018) 6900–6919.
[15] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (2009).
[16] F.P. Rubin, P.S.S. de Souza, W. dos Santos Marques, R.R. de Oliveira, F.D. Rossi, T. Ferreto, Evaluating energy and thermal efficiency of anomaly detection algorithms in edge devices, in: 2020 International Conference on Information Networking (ICOIN), IEEE, 2020, pp. 208–213.
[17] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, ICDM '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 413–422. doi:10.1109/ICDM.2008.17.
[18] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00, ACM, New York, NY, USA, 2000, pp. 427–438. doi:10.1145/342009.335437.
[19] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, Lof: Identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00, ACM, New York, NY, USA, 2000, pp. 93–104. doi:10.1145/342009.335388.
[20] Z. He, X. Xu, S. Deng, Discovering cluster-based local outliers, Pattern Recogn. Lett. 24 (2003) 1641–1650.
[21] Z. He, X. Xu, S. Deng, Squeezer: an efficient algorithm for clustering categorical data, J. Comput. Sci. Technol. 17 (2002) 611–624.
[22] M. Goldstein, A. Dengel, Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm, KI-2012: Poster Demo Track (2012) 59–63.
[23] H.-P. Kriegel, M. Schubert, A. Zimek, Angle-based outlier detection in high-dimensional data, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, ACM, New York, NY, USA, 2008, pp. 444–452. doi:10.1145/1401890.1401946.
[24] S. Rayana, ODDS library, 2016. http://odds.cs.stonybrook.edu.
[25] S. Mittal, A survey on optimized implementation of deep learning models on the nvidia jetson platform, J. Syst. Architect. (2019).
[26] T.S. Chis, P.G. Harrison, Performance-energy trade-offs in smartphones, in: Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, ACM, 2016, pp. 127–135.