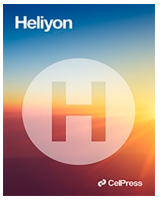
Heliyon 9 (2023) e21475

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Towards a sustainable edge computing framework for condition monitoring in decentralized photovoltaic systems

Ibtihal Ait Abdelmoula [a,b,*], Samir Idrissi Kaitouni [b], Nassim Lamrini [b], Mourad Jbene [a], Abdellatif Ghennioui [b], Adil Mehdary [a], Mohamed El Aroussi [a]

[a] SIRC/LAGeS laboratory-EHTP Hassania School of Public Works, Casablanca, Morocco
[b] Green Energy Park (UM6P and IRESEN), Benguerir, Morocco

ARTICLE INFO

Keywords:
Digitalization
Smart grids
Anomaly detection
Edge-computing
Embedded
Online monitoring

ABSTRACT

In recent times, the rapid advancements in technology have led to a digital revolution in urban areas, and new computing frameworks are emerging to address the current issues in monitoring and fault detection, particularly in the context of the growing renewable decentralized energy systems. This research proposes a novel framework for monitoring the condition of decentralized photovoltaic systems within a smart city infrastructure. The approach uses edge computing to overcome the challenges associated with costly processing through remote cloud servers. By processing data at the edge of the network, this concept allows for significant gains in speed and bandwidth consumption, making it suitable for a sustainable city environment. In the proposed edge-learning scheme, several machine learning models are compared to find the best suitable model achieving both high accuracy and low latency in detecting photovoltaic faults. Four light and rapid machine learning models, namely, CBLOF, LOF, KNN, ANN, are selected as best performers and trained locally in decentralized edge nodes. The overall approach is deployed in a smart solar campus with multiple distributed PV units located in the R&D platform Green & Smart Building Park. Several experiments were conducted on different anomaly scenarios, and the models were evaluated based on their supervision method, f1-score, inference time, RAM usage, and model size. The paper also investigates the impact of the type of supervision and the class of the model on the anomaly detection performance. The findings indicated that the supervised artificial neural network (ANN) had superior performance compared to other models, obtaining an f1-score of 80 % even in the most unfavorable conditions. The findings also showed that KNN was the most suitable unsupervised model for the investigated experiments achieving good f1-scores (100 %, 95 % and 92 %) in 3 out of 4 scenarios making it a good candidate for similar anomaly detection tasks.

1. Introduction

Smart houses, smart grids and smart cities are all concepts referring to the integration of novel information and communication technologies (ICT) into the existing traditional infrastructures. The global crisis, climate change, huge population growth and resource depletion around the world are all drivers that led to the development of such perceptions [1]. Moreover, the smart city framework

* Corresponding author. SIRC/LAGeS laboratory-EHTP Hassania School of Public Works, Casablanca, Morocco.
E-mail address: aitabdelmoula@greenenergypark.ma (I. Ait Abdelmoula).

https://doi.org/10.1016/j.heliyon.2023.e21475
Received 22 July 2023; Received in revised form 9 October 2023; Accepted 22 October 2023
Available online 28 October 2023

includes a wide range of components covering buildings, transportation, electrical grids, healthcare and security. All these components aim to deliver modern and high-quality services to people which are considered the main valuable asset. In order to achieve this ultimate goal, major changes should be made to cover the increasing and complicated needs of a fast-growing community. Smarter management cannot be achieved without three main drivers known as the three 'D's namely Digitalization, Decarbonization and Decentralization. As a matter of fact, the most recent industrial revolutions were triggered by digitalization and the introduction of modern information technologies, cloud computing, big data and artificial intelligence phenomena [2,3]. With the increasing use of sensor networks, there has been a significant surge in the generation and real-time processing of vast quantities of data. Sensor nodes are ubiquitously present in various settings, ranging from architectural structures to automotive systems and electrical networks. Despite variations in their objectives and importance, sensors generally adhere to a common operational paradigm, which involves the timely capture of signals. However, sensor data may exhibit abnormalities due to sensor malfunctions or genuine anomalies in the environment being detected. Therefore, it is crucial to consider the timely detection of anomalies to effectively mitigate the emergence of catastrophic circumstances. In that regard, anomaly detection has been employed across many fields for various applications. Scholars have conducted investigations on many strategies pertaining to water consumption data [4,5], energy data in buildings [6], security in smart houses [7], and healthcare applications [8,9], among others. Some authors conducted reviews on the time-series anomaly detection techniques applied specifically on IoT data streams. For example, in Ref. [4], the authors conducted an evaluation of the most relevant approaches for anomaly detection in time-series especially in IoT field while giving different examples from literature about industry 4.0, smart buildings and energy applications. They also highlighted major challenges that should be taken into consideration while implementing anomaly detection techniques on IoT timeseries data. The particularities, high dimensionality, and dynamics of IoT sensors data make the process even harder. That is why the authors underlined how crucial it is to have a continuous awareness of the contextual environment while analyzing this type of data. Similarly, authors in Ref. [5], investigated over 60 published papers dealing with the anomaly detection techniques in IoT time-series highlighting the methods used, the challenges that need to be addressed and also giving some recommendations about the requirements needed for the implementation of such techniques.

Nonetheless, due to the growing digitalization, huge volumes of data are getting transferred to centralized servers which constitutes a big pressure on the network. The variety, velocity, veracity and variability of data volumes makes the process more challenging [6]. On another hand, the volumes of data circulating within the network still need lot of processing steps in order to be used properly. They should be cleaned, analyzed and carefully processed to render efficient information and useful knowledge. Moreover, data from tens of thousands of connected devices must be processed quickly, otherwise the network would experience significant latency and break down. For this reason, the concept of edge computing appeared as a novel technique that may improve the processing possibilities in Big Data platforms. This is especially relevant in a smart city context, as solar-based distributed energy generation is one of its fundamental characteristics. These distributed units require constant and real-time monitoring to ensure not only their performance but also the stability of the entire grid. Similar to any physical system, anomalies can occur in solar plants. Detecting anomalies can be sometimes possible just by examining data visually especially if it is an outlier. However, a human intervention is not feasible especially with all the large volume of datasets and their distributed behavior. For this reason, machine learning (ML) techniques are widely used in this area. In fact, using the recent improvements in ML technology is essential to precisely expose the irregularities within photovoltaic systems, especially that preventing anomalies is the only way to advance the growth of the solar field and enable its massive deployment and friendly integration. Being aware of the anomalies as soon as they appear enables the operators to intervene quickly and conduct the necessary adjustments or repair; therefore, the performance losses are considerably reduced. Yet, deploying machine learning algorithms onto resource-constrained edge devices is not a straight-forward process. In fact, these devices are typically used for tasks such as data collection and pre-processing and are not well-suited for the computationally intensive tasks required for machine learning. As a result, various optimizations must be conducted to ensure that the algorithms can run effectively on these devices.

In this context, a number of works in the literature studied edge computation techniques in the scope of smart grids linking it, for the majority, with the Digital Twin concept. In Ref. [7], the authors developed a Digital Twin platform using a cloud-edge integration framework to diagnose faults in a microgrid. The authors chose a data-driven method to detect anomalies instead of a model or knowledge-based technique stating that the former is more adequate to a microgrid case. However, two problems faced this method; the first was the need of high quantity of data and the second was the high computational resources. According to the authors, these challenges were mitigated using a cloud-edge framework. In an older work [8], the authors studied anomaly detection in the form of cyber-attacks against PV systems in a distributed framework. Simulation scenarios showed how the aggregation of sources and the fusion of models are beneficial in detecting a large spectrum of anomalies. However, the study focused only on the simulation part using GridLAB-D tool. Other studies approached edge computing in different fields. In Ref. [9], the authors developed a workflow to detect anomalies in machinery log files leveraging both cloud and edge computing capabilities. In Ref. [10], the authors evaluated five different algorithms (LR, SVM, DT, RF, ANN) for anomaly detection in IoT traffic nodes and implemented them on a Raspberry to compare results with a laptop performance and execution time. In the greenhouse environment, the authors in Ref. [11] propose the implementation of ML algorithms on two types of Edge devices (Arduino and Raspberry). The overall performance is measured using accuracy, consumption and execution time. A metric has been offered to take into account both the speed and electrical consumption and thus guide the choice of the best compromise between software and hardware integration. These related works reveal the importance of combining anomaly detection with edge computing capabilities in Smart Grid applications which motivates their adoption in solar PV condition monitoring.

Furthermore, several studies have discussed photovoltaic fault detection in smart grids using a wide range of machine learning techniques. The study in Ref. [12] was conducted on Modelica software and dealt with both prediction and anomaly detection of a

Table 1
Sub-keywords used in the systematic review.

Category I	Category II	Category III
• edge • edge computing • edge artificial intelligence • edge AI	• Anomaly detection • Fault detection	• PV • Photovoltaic • Solar

Table 2
Overview of the relevant works related to anomaly detection and edge computing in photovoltaic systems.

Ref	ML algorithms used	Anomalies detected	Edge devices used	Number of PV systems used
[18]	Random under-sampling boosting (RUSBoost)	line-to-line (LL) fault, open circuit (OC) fault, partial shading (PS) fault, and degradation (DF) fault	No	One PV system of nominal power 4.8 kWp
[19]	Stacked gated recurrent unit (GRU) neural network.	The open circuit, short circuit, degradation, Partial shading, Soiling, PID and the average reduction in power output.	No	23 configurations of the PV system
[20]	Graph neural network	Open and short circuit fault, degradation, Partial shading, Soiling, PID	No	6 PV systems
[21]	Artificial neural network (ANN), Recurrent Neural Networks (RNN) and Bidirectional LSTM	Line to line, Line to ground fault, Connectivity fault and Bypass Diode fault	No	3 PV systems each one has a nominal power of 4 kWp
[22]	AlexNet CNN with 2-D scalograms	Partial shading	Adafruit PyBadge MCU	One PV system of 1.7 kWp
[23]	Ensemble learning of Semi-supervised Self-Training (ELSST): K-nearest neighbors (KNN), Decision tree (DT) and Support vector machine (SVM) classifier	Line-to-Line and Arc fault, partial shading, open-circuit and power tracker unit fault.	No	One PV system
	Artificial neural network (ANN) for fault detection and Multi-stacking ensemble learning of Random forest (RF), AdaBoost, CatBoost, and XGBoost for fault classification.	Dust accumulation, partial shading, open circuit diode with dust accumulation and shunted diode with shading.	No	One PV system
[24]	Random forest (RF) and Artificial neural network (ANN)	Connector fault, PID, Partial shading and building shading condition, Failing bypass diode/short circuit (SC) soiling accumulation and Glass breakage.	Health-Helio (HH) sensors by SmartHelio	One PV system of nominal power 2.94 kWp
[25]	Extreme gradient boosting (XGBoost) classifier for fault classification	Line to line fault, partial shading	Edge node implementing eXplainable Fault Detection Systems (XFDS)	Two photovoltaic panels: monocrystalline (STM5-40/36) and polycrystalline (Solartech SPM-020P–R)
[26]	Siamese-twin neural networks for anomaly detection kNN, SVM, XGBoost, Random Forest and Neural Network for Fault classification	Shading	Raspberry PI, Nvidia Nano and Google Coral	IV tracing prototype composed of two panels
[27]	Neural network processing for anomaly detection, combined with the convolutional neural network for anomaly classification	Silicon wafer defect	rk3399 pro version	_
[28]	Image convolution by Gaussian Filter Coefficient	Defective cells	_	_
[29]	deep conventional neural networks (DCNNs) for fault detection and diagnosis and fault classifications	partial shading effect, dust deposit on PV modules surface, short-circuited PV module and bypass diode failure	Raspberry Pi 4	PV modules
[30]	Artificial neural network is developed to detect faults and an effective stacking ensemble learning algorithm is developed to classify the nature of the fault	dust deposit, partial shading, open circuited diode and dust accumulation, partial shading and dust accumulation, and shunted diode in a shaded PV module	Raspberry Pi 4	PV string of 3 modules

photovoltaic system. The solution coupled the physical and digital models in different ways and aimed at finding the models that achieved the best accuracy. In the same scope, the authors in Ref. [13] presented a platform developed as a Digital Twin for PV systems aiming at predicting and detecting anomalies. A broad spectrum of artificial intelligence (AI) algorithms are available on the Cloud platform for the PV plant monitoring and diagnosis. The work in Ref. [14] also studied ML algorithms for the anomaly detection task in PV plants. In the article, a comparison is made between unsupervised anomaly detection algorithms on several datasets of PV data over a period of 4 months. The authors looked for uncommon anomalies that are difficult to identify with little data. Therefore, an aggregation of 70 PV systems was investigated and the performance of algorithms was evaluated on each distinct anomaly. Anomalies in

Table 3
Technical characteristics of the Data Acquisition system.

Corresponding layer	Hardware device	Description	Technical characteristics
Perception	Photovoltaic DC datalogger *TR16-RS485*	Measurement device for PV strings currents and voltage. Can measure up to 16 current signals and 1 DC voltage	Voltage measurement margin: 30–1000 V Voltage error: 1 % FS Current error: ±0.5 % FS Communication protocol: Modbus RTU
	Current sensor *M/TR*-25 Acc *x 4*	Hall effect transformer to connect with the PV datalogger for DC currents measurements	Maximum current: 25A Number of internal current circuits: 4
	Irradiance sensor *Kipp&Zonen CMP21*	Solar irradiance measurement pyranometer (GHI and DHI)	Irradiance range: 0–4000 W/m^2 Accuracy: ±2 %
	Irradiance sensor *Kipp&Zonen CHP1*	Solar irradiance measurement pyrheliometer (DNI)	Irradiance range: 0–4000 W/m^2 Accuracy: ±1 %
	Master gateway *EDS Energy Manager*	Connects to multiple slave devices through serial RS-485 communication bus and store data in a web server	Communication protocol: HTTP/ Modbus RTU
	Meteorological datalogger *Campbell scientific CR1000*	Multi-purpose datalogger used in monitoring and control applications	Analog inputs: 16 Communication protocol: Modbus TCP
Communication	Ethernet switch devices *FL SWITCH SFN 4TX/ FX*	Ethernet switch with 4 RJ45 10/100 Mbps	Supply voltage: 24 V DC
Application	Monitoring server	Linux server	

PV systems can also rise as a result of malware intrusions. In Ref. [15], a hardware experiment is performed to evaluate the performance of two intrusion detection approaches for PV inverters. Both approaches are reported in the literature, and each has its own characteristics, however the article considers that real experimentation is lacking to really evaluate the performance of the methods in a real test environment. Regarding anomaly detection in time series in general, the work in Ref. [16] proposes a framework to redefine anomalies in time series and benchmark real and synthetic datasets on different algorithms to understand the capabilities of each algorithm on the types of anomalies.

To conduct a comprehensive state of the art that captures the most important literature in our research field, we adopted a searching methodology and applied it in this work. The methodology is called Sub-keyword Synonym Searching (SSS) [17] and its purpose is to identify relevant papers by multiple searches with synonym sub-keywords. In this paper, Scopus is the main search engine of the methodology, and the full list of searching keywords in Scopus is the full combination of each category of Table 1. The result of this review of literature is summarized in Table 2.

These previous works on anomaly detection in photovoltaic systems have mostly focused on using detection, classification, or their hybrid combination to create real-time monitoring frameworks. However, there is a lack of extensive discussion on an edge computing framework for anomaly detection in PV systems as only 7 papers discussed the concept. Most of the papers that discussed it applied it to small-scale systems and did not provide an architectural scheme for large-scale deployment in smart cities or grids. Nowadays, many applications require both real-time processing and critical analysis. Therefore, edge-based processing is highly recommended. This raises the following research question: how can we implement highly critical analytics on the edge while rationalizing computation costs and energy needs? In this study, we aim to provide insights to answer this question by using a photovoltaic use case to investigate anomaly detection techniques and their deployment on an edge infrastructure. To the best of our knowledge, this is the first study that provides an edge computing architecture dedicated to smart cities deployment on the subject of PV fault detection. Unlike existing studies that implement anomaly detection and classification on local edge devices in an isolated PV system or a small prototype, here we focus on providing a detailed workflow for decentralized fault detection in PV systems using a campus of more than 30 houses with solar rooftops.

The novelty of this work resides in the edge computing framework dedicated to distributed solar PV plants in an urban area. The combined software and hardware methodology described will help increase the penetration of solar into our traditional grids. In the following, we highlight our contributions.

1) We create four synthetic datasets representing four different anomaly scenarios by varying the percentage of power magnitude. This technique is useful when dealing with the lack of labelled datasets. After that, we compare several unsupervised and supervised anomaly detection approaches to detect power drops in photovoltaic datasets and investigate the learning type's impact on the model performance.
2) We propose a framework to detect anomalies in decentralized PV plants by embedding the most performant models into edge nodes taking into account their prediction time, accuracy, Random Access Memory (RAM) usage and model complexity. This approach takes advantage of the hybridization between edge, fog and cloud layers.

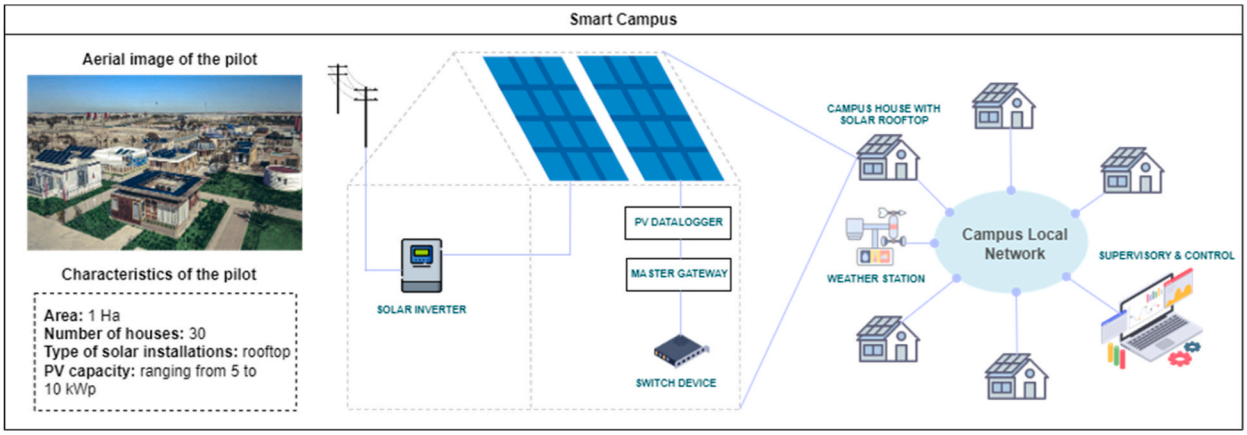


Fig. 1. Overview of the campus communication architecture.

Tilt angle (°)	15,7	27	31	27,7	31
Azimuth angle (°)	175	220	180	135	205
PV module capacity (W)	275	275	335	350	275
Technology	Monocrystalline	Dual Glass PERC Monocrystalline	Monocrystalline	Polycrystalline	Dual Glass PERC Monocrystalline
Measured Module efficiency (%)	14,77	15	19,24	16,5	15,06

Fig. 2. Solar systems located in the solar village as part of the case study.

The paper is organized in the following way: in Section 2, we present the experimental setup and the methodology of the work. Section 3 summarizes the results of the paper and provides concluding remarks.

2. Experimental setup and methodology

2.1. Experimental setup

The present study is conducted in a solar campus spread over 1 ha, comprising many residential houses equipped with solar rooftops utilized for both R&D and accommodation activities. The campus is located in the Green and Smart Building Park in Benguerir [31], Morocco. Photovoltaic arrays are interconnected with on-grid inverters within each individual house, facilitating the conversion of direct current (DC) power to alternating current (AC) power. Furthermore, the monitoring of DC power is carried out through the use of a photovoltaic datalogger. The frequency of data gathering is 15 min. The dataloggers employ a pre-sampling technique to collect data, and the supervisory system retrieves the average value during a 15-min interval. Table 3 presents the data obtained from each source independently. Additionally, Fig. 1 provides a comprehensive overview of the architectural layout of the campus, with a closer examination on one of the houses.

A systematic methodology is employed to gather data from the campus houses. Despite the absence of a standardized communication and data architecture for smart grids, a recent study [5] has proposed a comprehensive layered framework consisting of four stages (perception, connection, analytics, and security). This framework aims to support the development of an Internet of Things (IoT)-enabled smart grid. In this particular scenario, the perception layer consists of measurement nodes that are strategically positioned in the field. These nodes are responsible for capturing information related to the photovoltaic system, including solar irradiance, ambient temperature, string current, and string voltage. The connectivity layer assumes the role of facilitating the transmission of data between the perception layer and the monitoring system. This layer is composed of the wired Ethernet network and the switch devices, which serve as the means of establishing and maintaining the connection. The application layer encompasses various analytical and

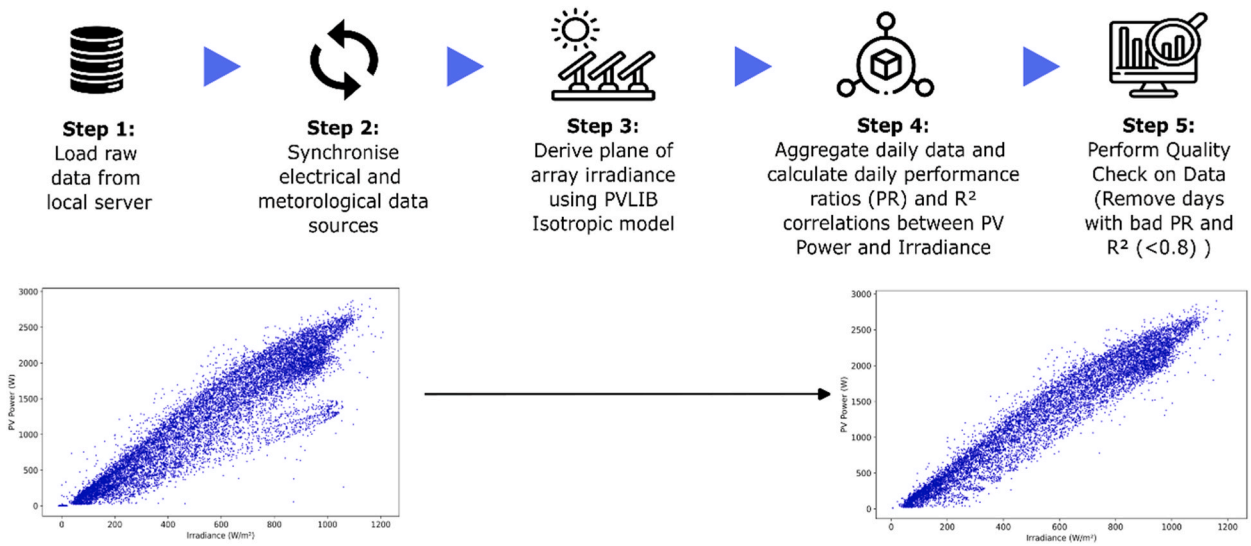


Fig. 3. The processing methodology applied to raw data.

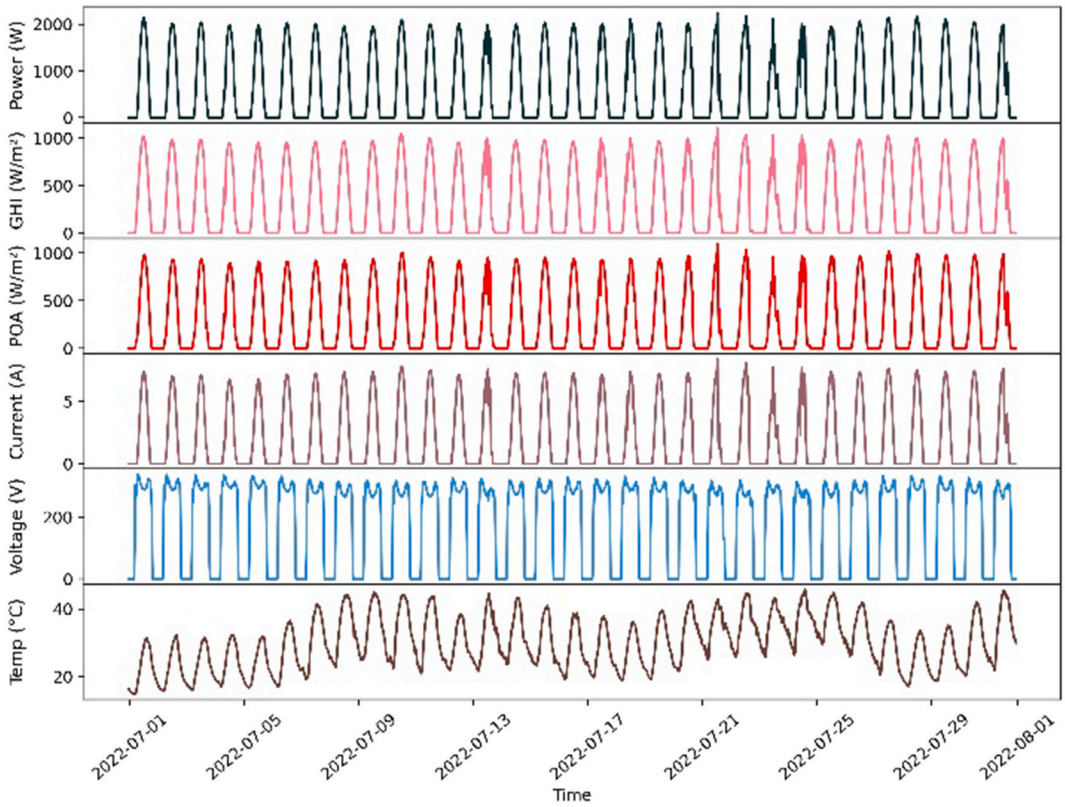


Fig. 4. A sample of the dataset retrieved for the case study

processing tasks that need to be executed on the gathered data. These jobs include but are not limited to forecasting, anomaly detection, performance analysis, and condition monitoring. The security layer, while not addressed in this study, is a subject of ongoing research and an essential consideration in the design of smart grids. In the framework of this paper, the case study is part of a living lab "solar village of solar decathlon Africa", in which more than 30 zero energy buildings are equipped with different solar PV technologies. The latter have different characteristics in terms of installed capacities, orientations, inclinations, energy yield and performance ratio providing a unique groundwork to identify the key factors that impact the performance of solar systems with respect to the local

Table 4
The input features used in the study.

Variable name	Type of variable	Description	Unit
DC Current (Idc)	Measured	DC current of the monitored string	[A]
DC Voltage (Vdc)	Measured	DC voltage of the monitored string	[V]
DC Power (Pdc)	Calculated	DC power of the monitored string	[W]
Global Horizontal Irradiance (GHI)	Measured	Global horizontal irradiance retrieved from meteorological station	[W/m^2]
Plane of Array Irradiance (POA)	Calculated	Plane of array irradiance calculated using isotropic model	[W/m^2]
Air temperature	Measured	Air temperature retrieved from meteorological station	[°C]

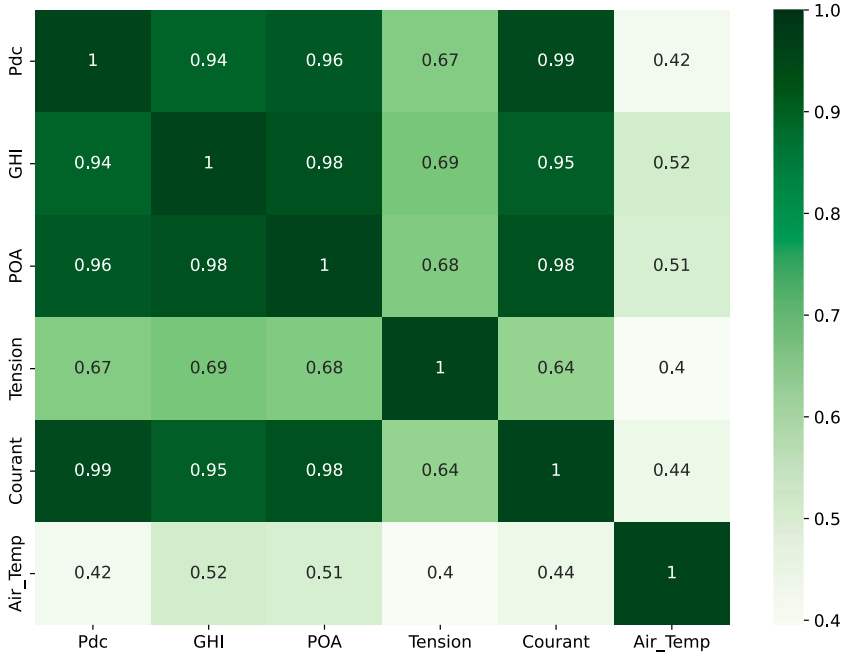


Fig. 5. Correlation matrix of the features.

semi-arid climate. Fig. 2 presents the attributes of 5 solar systems located in the solar village.

Data collected from the testbed was cleaned using a structured framework inspired by international standards and recommendations for photovoltaic systems monitoring. The cleaning process involves a set of filters to remove erroneous and invalid records as well as performance metrics calculation such as the daily performance ratio of the PV system under study. First, we resampled the data and made sure the data sources were synchronized. We then performed several processing techniques such as: dropping repetitive and duplicate values, filtering out the night values, checking that the recorded values are among certain threshold (Current >0 and $<$ 1.5*Isc, Voltage>0 and $<$ 1.2*Ns*Voc such as Isc is the short circuit current, Voc the open circuit voltage found in the datasheet and Ns is the number of modules per string). We then applied the R^2 correlation filter to remove days with R^2 inferior to 0.8 since irradiance should be highly correlated with the output power. After that, we applied the performance ratio (PR) filter to remove days with PR $<$ 0.8. Fig. 3 shows the processing methodology applied to the raw dataset illustrated in Fig. 4 while the input features used in the study are described in Table 4 and their corresponding correlation matrix in Fig. 5. The DJI Mavic 2 Enterprise drone is utilized at regular intervals to perform inspection missions within the platform premises. Its primary objective is to identify any faults or issues and afterwards notify operators of the necessary maintenance measures, including cleaning.

2.2. Methodology

We will start by conducting a benchmark of the methods used for unsupervised anomaly detection and use the PyOd library [32] to rapidly scan 11 algorithms using their default parameters and evaluate their accuracy on the same anomalous datasets. The algorithms used in this work are classified into statistical, distance-based, machine learning, ensemble and artificial neural network methods. After that, the best selected algorithms will be further investigated based on the complexity, rapidity and prediction time. Finally, an edge computing framework and its constituting layers is proposed and described in detail.

Since our purpose is to evaluate anomaly detection in an edge computing framework, we will conduct experiments using different scenarios to understand the impact of model selection and the anomaly type on the accuracy of the solution [33]. Anomaly detection

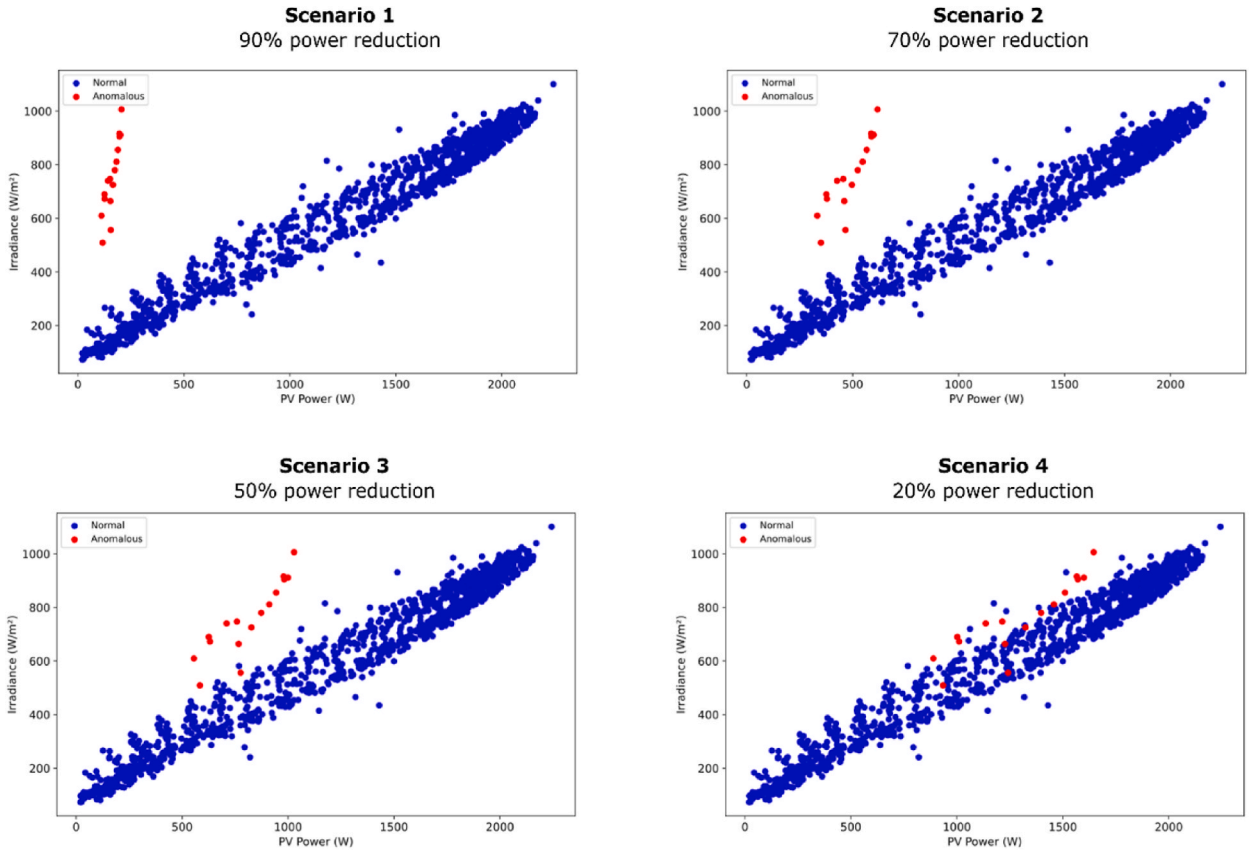


Fig. 6. Overview of the anomaly scenarios.

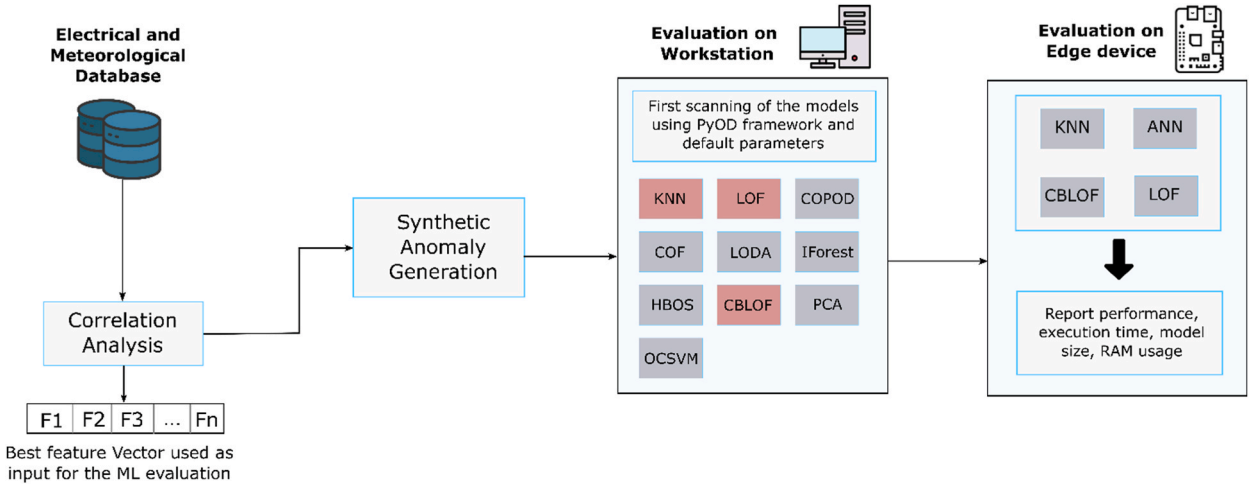


Fig. 7. Work methodology.

can be a tedious task especially when data is not labelled and when it is hard to separate anomalous events from normal ones. For this purpose, synthetic datasets can be used to better understand the anomaly behavior and map each case with its adapted model. Another parameter to observe for our case study is the lightness of the model since our goal is to transfer the model to the edge device for a real-time monitoring of the PV system. The work in Ref. [34] followed the same approach of synthetic datasets and simulated anomalies to develop a Digital Twin on the cloud, however, they used only daily averaged data and didn't evaluate the models on an edge setting. In our case, there is a need to go further and study the model-anomaly duality under a more constrained edge

environment necessitating a real-time cost-effective deployment [35].

Furthermore, we describe the scenarios that were tested on the real dataset. In the first scenario, we injected a reduced magnitude of the power output by different percentages ranging from 20 % to 90 % as illustrated in Fig. 6. In the second scenario, we varied the supervision type along with the ML models used to detect the anomalies. Two techniques were experimented: supervised learning represented by MLP and unsupervised learning represented by CBLOF, LOF and KNN. In the unsupervised techniques, it was necessary to conduct experiments on the contamination rate to choose the best value. Finally, the methodology of the work is summarized in Fig. 7.

2.3. Algorithms

2.3.1●. Local outlier factor (LOF)

Local outlier factor as defined by Ref. [36] is an unsupervised technique that relies on neighborhood density to detect anomalies. It is computed by dividing the density of the point by the densities of its k closest neighbors. Because the density of the neighborhood constitutes the basis of this method, LOF works well on datasets that are imbalanced. For each point, the algorithm calculates a parameter called k-distance that defines the distance between the point of interest and its farthest neighbor, then, the local outlier factor is computed. The points presenting high outlier factor and therefore low density are considered anomalies.

2.3.2●. K-nearest neighbors (KNN)

kNN is initially a supervised ML algorithm proposed by Ramaswamy [37]. However, the model can also be transformed into an unsupervised learning technique in anomaly detection use cases since anomalies are regarded as rare events that are hard to label. kNN belongs to the proximity-based clustering techniques meaning that it relies on the distance of a data point to its kth neighbor to determine whether it is an anomaly or not. The farthest a data point is from its closest cluster, the higher its probability of being anomalous. In practice, it is necessary to find the optimal value for the hyperparameter k representing the number of neighbors in the model. The k-NN algorithm has 2 main parameters: the distance measure and the number k of nearest neighbors to use in the calculations.

2.3.3●. Cluster-Based Local Outlier Factor (CBLOF)

Cluster-Based Local Outlier Factor (CBLOF) is an unsupervised method that belongs to the family of clustering-based algorithms. It has been introduced in Ref. [38] by Zengyou and was aiming to address some drawbacks of the existing clustering methods at the time. The method was effectively proven to be competitive compared to other clustering techniques and to the individual Local Outlier Factor (LOF) method. In order to spot outliers, CBLOF use both the notions of size and distances. In fact, a data point can belong to a cluster and still be considered an outlier because of the size of this cluster.

2.3.4●. Multi-layer perceptron (MLP)

The multilayer perceptron (MLP) is a feed-forward neural network inspired from the human brain functioning. It is constituted from an input layer, a hidden layer, and an output layer. The input layer comprises the feature vector. The hidden layers are made of multiple neurons and are positioned between the input layer and the output layer. The output layer is responsible for delivering the result of the classification. A multilayer perceptron (MLP) operates in a manner that is analogous to that of a feed forward network. Data travels from input to output layer in the forward direction. MLPs are used to resolve non-linear problems due to their design enabling them to approximate any continuous function [39].

3. Results and discussion

3.1. The proposed edge framework

The photovoltaic systems installed on campus are interconnected together in a traditional setup. The dataloggers send their raw data to a centralized supervisory server, which is responsible for processing and analyzing the data. However, in a smart decentralized configuration, the hazards of network congestion and delayed alarm generation make it infeasible to send raw data to a central Cloud server for decision making purposes [40]. Therefore, we propose a novel edge framework aiming to solve this issue in a more sustainable and intelligent way. The concept of edge computing has arisen as a solution to address the challenges associated with network congestion caused by the transfer of extensive datasets to distant servers, particularly in the context of big data and smart cities [41]. Utilizing the capabilities of the edge yields the subsequent benefits: **(1) Reduced latency:** The latency in data processing is minimized as a result of conducting analytics in close proximity to the data source. The reduced distance that data needs to traverse leads to enhanced response times. The significance of this is particularly pronounced in the context of applications that necessitate the processing of data in real-time, such as IoT sensors and distributed energy systems, **(2) Reduced bandwidth:** The process of developing a communication architecture requires a comprehensive comprehension of the bandwidth needs associated with each utilized communication protocol. The transmission of data from one location (point A) to another (point B) necessitates the utilization of network connectivity, resulting in the consumption of a portion of the available capacity or bandwidth. Currently, there is an anticipation that smart grids would impose significant demands on forthcoming communication networks [41]. The reduction of bandwidth is therefore an imperative research direction in this domain, which can be effectively accomplished through the implementation of edge computing. **(3) Reduced costs:** The expenses related to communication in smart networks are directly associated

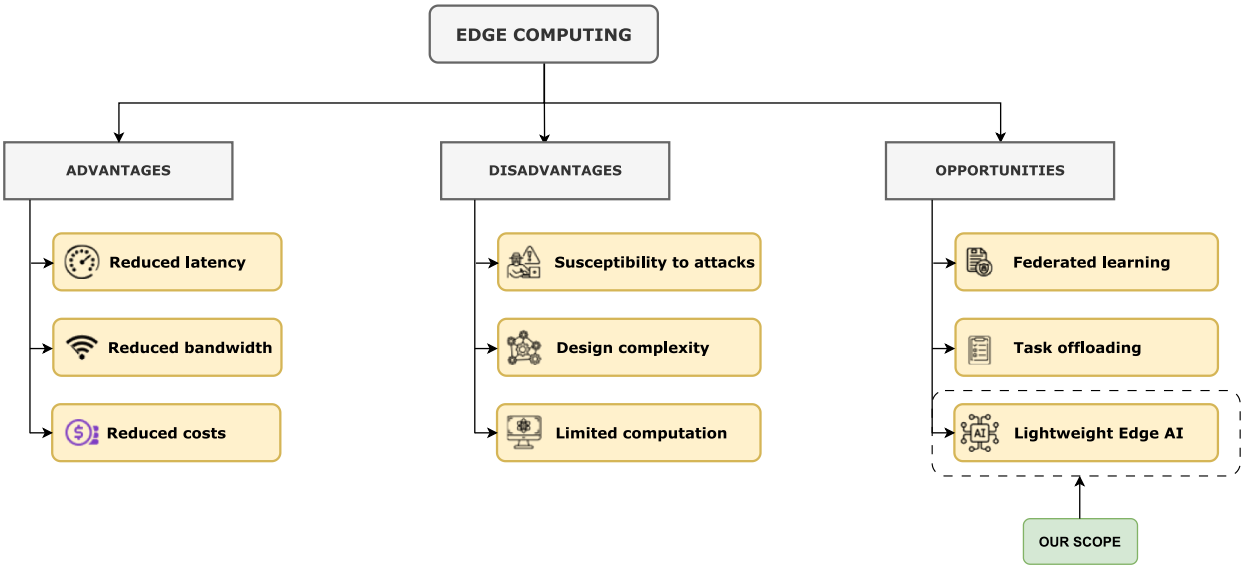


Fig. 8. Advantages, disadvantages and opportunities in edge computing paradigm.

with the provision of data storage and analytics services on cloud platforms. These services impose substantial fees that could be mitigated by transmitting only relevant information to the central cloud rather than the entire raw datasets. Despite the numerous advantages and opportunities it presents, edge computing, like any other technological concept, is not without limitations that pose challenges to its widespread implementation. The limitations are associated with the inherent characteristics of edge computing and can be succinctly described as follows: (1) The susceptibility to cyber-attacks: due to the distributed nature of edge nodes, which often involves deployment in remote and isolated areas, cyber-attacks can be heightened and easier to perform. The vulnerability of these nodes is due to its constrained resources forcing them to implement only constricted defense measures and security protocols. (2) The implementation of edge computing necessitates a multitude of adaptations in order to enhance the reliability and consistency of its operation. In addition to the hardware configuration, the deployment of a trustworthy edge node entails software improvements such as task offloading, and in-memory computing [41]. (3) Limited computation: The limited processing capabilities available in edge nodes create considerable constraints, hence creating issues when it comes to building intricate machine learning or deep learning models. Hence, it is imperative to conduct a thorough investigation for lightweight models [42] and find a balance between computational efficiency and precision. Fig. 8 summarizes all the above aspects related to edge computing.

The framework is illustrated in Fig. 9. It consists of 4 main layers.

- **The thing layer** represents the devices that generate raw data at the source. In our case study, it represents the photovoltaic dataloggers that measure dc voltage and current within PV systems, in addition to the weather sensors that measure solar irradiance and temperature, the PV inverters that convert DC current to alternating current (AC) and the drones that perform thermal inspection of the PV modules periodically.
- **The edge layer** describes the edge nodes that will handle the processing and analysis of incoming data flows before sending the results to the cloud. It consists of a cluster of Raspberry-PIs deployed within each of the decentralized PV systems. The edge layer is positioned as a middle-layer between the thing and fog layers. It is used to enhance the efficiency of the monitoring system by reducing the processing time and bringing AI capabilities close to the devices.
- **The fog layer** depicts the intermediate layer between the edge and the cloud. It consists of servers that centralize the information from a group of edge nodes placed in the houses and performs additional computation and data aggregation from the same location (eg neighborhood).
- **The cloud layer** represents the ultimate layer where additional analysis and decision-support systems can be deployed. Many of the tasks that used to be conducted at this level are being transferred to the edge nodes thanks to this architecture. The cloud layer will only be responsible for collecting the decisions, alarms, and status reports received from the local nodes, rather than getting the entire raw datasets.

To deploy the framework within the campus, two procedures are executed: **(1) Offline mode:** This mode is comprised of historical data acquisition from the thing devices, data processing, and the benchmark of the available supervised and unsupervised models. The training of the best models is then conducted and evaluated. **(2) Online mode:** The intended approach involves conducting real-time monitoring to promptly identify errors, utilizing the most effective model chosen based on the benchmark outcomes. Upon the detection of a malfunction, a signal will be transmitted to the classification module with the purpose of ascertaining the precise category of the anomaly. The field validation is then conducted with drone imagery. Fig. 10 depicts these two methods in an

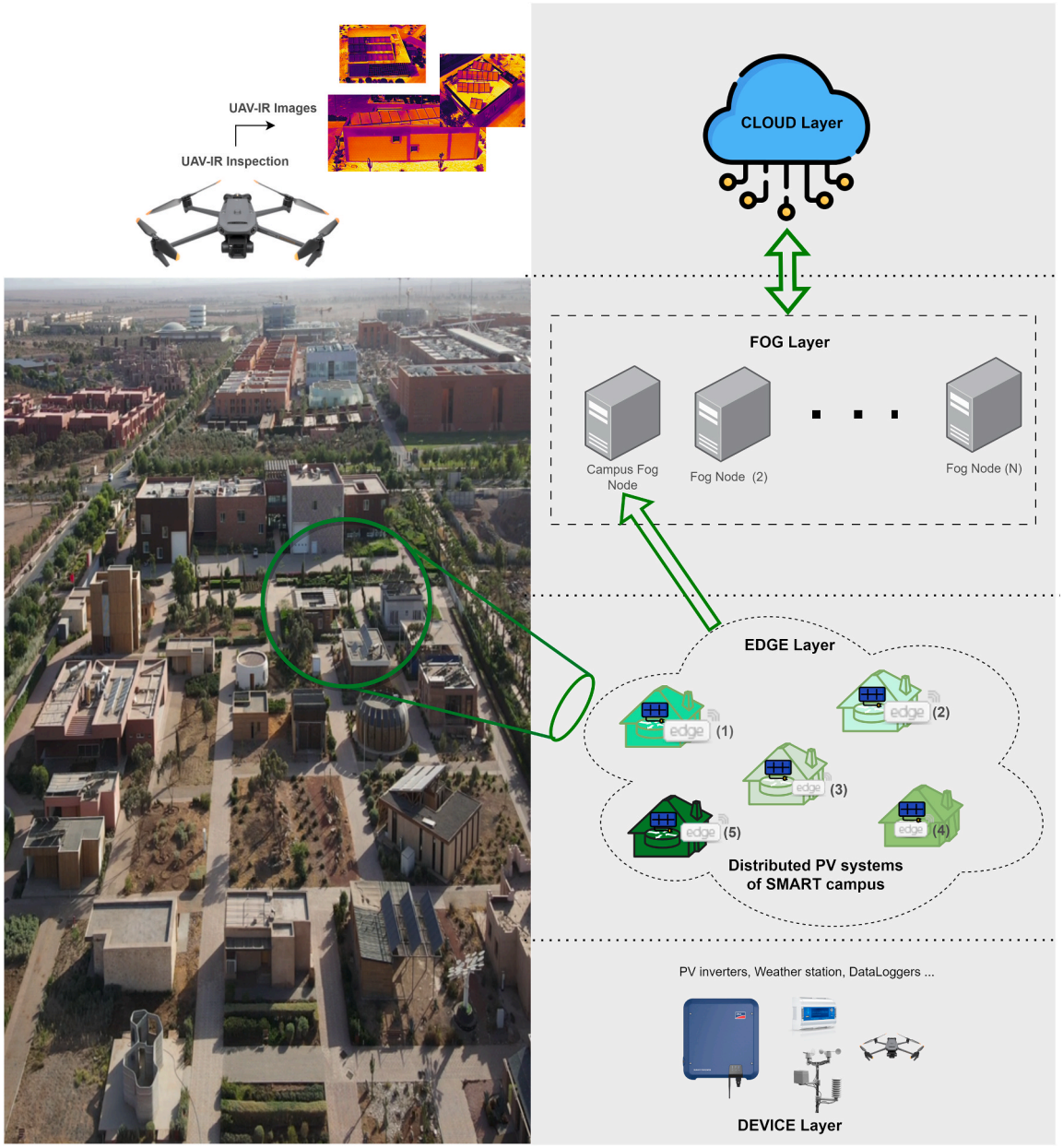


Fig. 9. The proposed edge framework.

organizational diagram.

In edge computing, the term System on chip (SoC) is used to describe the embedded technology used to perform computations at the edge of a network. It is generally composed of a central processor unit (CPU), memory (RAM), and input/output (I/O) interfaces [43]. The main goals of edge computing research is to boost the computing and processing efficiency on both hardware and software technologies. Several efforts have been deployed to increase computational capabilities of embedded processors while optimizing their size and energy consumption [44]. In this context, many giant providers of SoCs (such as NVIDIA and Google) have put in place a considerable amount of solutions and development boards dedicated to real-time edge processing. Table 5 describes some of the common entry-level SoCs used in research and prototyping. Since our goal is to embed machine learning models for monitoring a time-series signal, a low-cost board is sufficient as we don't require high graphical processing requirements. The board chosen to conduct the evaluation in this paper is the Raspberry Pi4 due to its affordability and availability in the market.

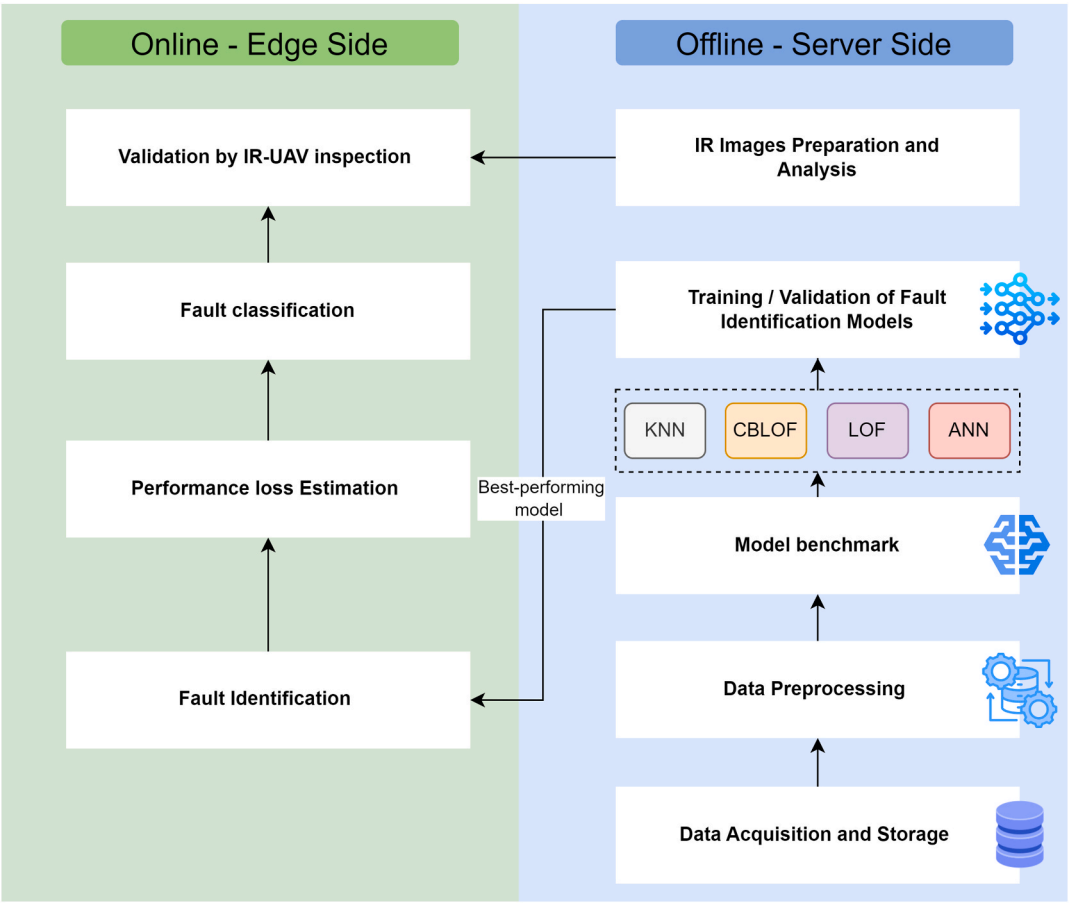


Fig. 10. Implementation procedure of the edge framework.

Table 5
Benchmark of the common SoCs used in research.

Features	Raspberry PI 4	Jetson Orin Nano 8 GB	Google Coral
Architecture	ARM	ARM	ARM
CPU	Broadcom BCM2711 quad-core Cortex-A72 64-bit	Arm Cortex-A78AE 64-bit CPU	Quad Cortex-A53, Cortex-M4F)
Memory	8 GB LPDDR4	8 GB 128-bit LPDDR5	4 GB LPDDR4
GPU	Broadcom VideoCore	1024-core NVIDIA GPU with 32 tensor cores	Integrated GC7000 Lite Graphics
Supported frameworks for ML	TensorFlow Lite	TensorFlow Lite	TensorFlow Lite
Cost (USD)	~100	~375	~170

3.2. Performance metrics

Performance evaluation is the most important part of the machine learning experiment. Depending on the problem to be solved, different metrics can be used in order to assess the goodness of the model. In a classification problem, like in our case, the most popular metrics are precision, recall and f1-score. Precision is a metric that computes the model's accuracy in classifying the positive class. The precision is highest when we have more correct positive classifications than incorrect ones. The recall, on the other hand, is more focused on the sensitivity of the classification. This means that the best recall is achieved when all positive instances are correctly labelled as positive. The f1-score is an average metric between precision and recall that gives a better conclusion on the performance of the model. An ideal classification task will achieve 100 % score in all three metrics. However, this is far from reality. A score is therefore judged by its closeness to the 100 % ideal situation. In a binary classification task, meaning that we have 2 classes: normal class labelled as 0 and anomaly class labelled as 1, we can encounter 4 different situations. The first two situations called "True Positive (TP)" and "True Negative (TN)" mean that both the predictions and ground truth labels are equal. The distinction is related to the class

Table 6
Definition of the metrics used in the evaluation.

	Precision	Recall	F1-Score	Support
Negative class (0)	$TN/(TN + FN)$	$TN/(TN + FP)$	$2 * Recall_0 * Precision_0/(Recall_0 + Precision_0)$	Count of 0 occurences
Positive class (1)	$TP/(TP + FP)$	$TP/(TP + FN)$	$2 * Recall_1 * Precision_1/(Recall_1 + Precision_1)$	Count of 1 occurences
Accuracy	$(TP + TN)/(TP + TN + FP + FN)$			Count of all occurences
Macro average	$(Metric_0 + Metric_1)/2$			
Weighted average	$(Metric_0 * Support_0 + Metric_1 * Support_1)/(Support_0 + Support_1)$			

Table 7
Performance results of ML models on a computer desktop.

	Model	Best parameters	F1_Score	Training/Fitting time	Prediction Time
Scenario 1	**CBLOF**	n_clusters = 10	1.00	1.54	0.04
	LOF	n_neighbors = 10	1.00	0.015	0.045
	KNN	n_neighbors = 10	1.00	0.01	0.06
	ANN	hidden_layer_sizes': (10, 30)	1.00	0.09	0.01
		max_iter = 500, alpha = 0.001, learning_rate = 'adaptive'			
Scenario 2	**CBLOF**	n_clusters = 10	0.95	2.61	0.07
	LOF	n_neighbors = 40	0.95	0.008	0.006
	KNN	n_neighbors = 10	0.95	0.01	0.09
	ANN	hidden_layer_sizes': (50, 100)	0.97	0.13	0.01
		max_iter = 500, alpha = 0.001, learning_rate = 'adaptive'			
Scenario 3	**CBLOF**	n_clusters = 10	0.90	2.27	0.07
	LOF	n_neighbors = 40	0.84	0.014	0.004
	KNN	n_neighbors = 10	0.92	0.02	0.10
	ANN	hidden_layer_sizes': (100,30)	0.94	0.14	0.008
Scenario 4	**CBLOF**	n_clusters = 10	0.49	2.17 s	0.06 s
	LOF	n_neighbors = 40	0.54	0.01 s	0.01 s
	KNN	n_neighbors = 10	0.59	0.01 s	0.05 s
	ANN	hidden_layer_sizes': (100, 30)	0.79	0.34 s	0.01 s
		max_iter = 500, alpha = 0.001, learning_rate = 'adaptive'			

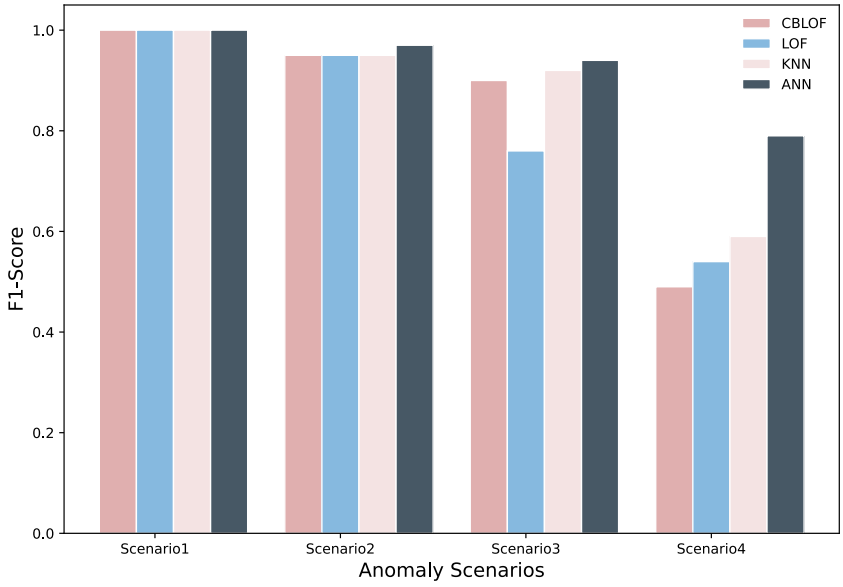


Fig. 11. F1-score for each ML model across the different scenarios evaluated.

type. In 'TP' case, the target is the positive class whereas the 'TN' case represents the negative class. Generally, in an anomaly classification task, the positive class (or the class of interest) is the one with anomalous instances and the negative class represents the one with normal data points. The other situations are called "False Positive (FP)" and "False Negative (FN)". They refer to the incorrect classification of normal instances as anomalous ones (FP) and anomalous instances as normal ones (FN). It is therefore important to specify the target class when calculating each metric [45]. In order to compute these metrics, we can either use the classification report to have a neat visual representation of the results or use the precision_recall_fscore_support function in sklearn. Table 6 below

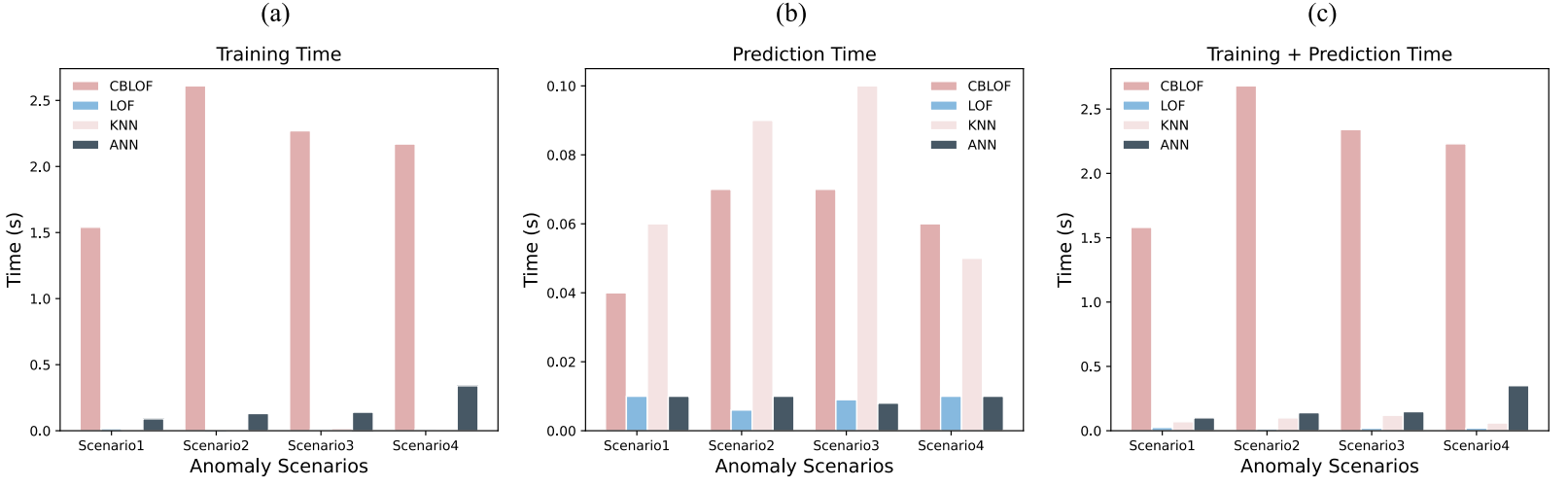


Fig. 12. Training and classification time for each ML model across the different scenarios evaluated: (a) Overview of time performance during the training phase (b) Overview of time performance during the prediction time (c) Overview of total time performance considering training and prediction phases.

Table 8
Performance results of ML models on an edge device.

Model	F1_Score	Prediction Time (after dump)	RAM (memory profiler)	Model size (Bytes)
CBLOF	1.00	4.50	25.1 MiB	25329
LOF	1.00	0.021	1.3 MiB	425425
KNN	1.00	0.18	0.4 MiB	116065
ANN	1.00	0.0047	0.5 MiB	17192
CBLOF	0.95	4.52	25.2 MiB	25329
LOF	**0.95**	0.021	1.3 MiB	425425
KNN	0.95	0.18	0.4 MiB	116065
ANN	**0.97**	0.0087	1.6 MiB	96552
CBLOF	0.90	4.55	24.9 MiB	25297
LOF	0.84	0.021	1.3 MiB	425425
KNN	0.92	0.18	0.4 MiB	116065
ANN	0.94	0.008	1.4 MiB	97016
CBLOF	0.49	4.55	25.2 MiB	25297
LOF	0.54	0.021	1.3 MiB	425425
KNN	0.59	0.18	0.4 MiB	116065
ANN	0.80	0.008	1.6 MiB	97096

represents the structure of the classification report with the equations describing each of the metrics. To have a fair idea about the goodness of the model, we use the macro average technique that takes into account both classes without weighing them. This is highly important when the dataset is imbalanced. In our case, choosing a weighted average will always return a score close to 1 since the normal class outweighs the anomalous one. In addition, the normal class is correctly classified by all the models that we experimented. Therefore, relying on the weighted average will lead to biaised and false interpretations. To have a correct interpretation, we used the macro average technique applied to the f1-score considering that this metric gives a more balanced indication compared to precision and recall [46].

Since our aim is to implement the models on an embedded configuration, it is important to consider additional metrics as well, mainly inference time, ram usage and model size. The inference time refers to the time spent by the model on the test set prediction. It is calculated using the time library in Python. The model size refers to the amount of space used to store the pretrained model. For this task, we use the pickle library, also in Python. RAM usage was calculated using the memory profiler library in Python.

3.3. Performance evaluation on a central server

The algorithms were trained and tested on a central server running an 11th Gen Intel Core (TM) i7-1165G7 CPU, 8 GB DDR4-3200 SDRAM, 512 GB NVMe SSD and NVIDIA GeForce MX450 GPU. The station is running Windows 11 and the machine learning algorithms are implemented using Python3. Table 7 describes the performance results of the selected ML models (CBLOF, LOF, KNN and ANN) when evaluated on the server side. The metrics that we highlight in this scenario are f1-score, training time (for supervised models), fitting time (for unsupervised ones) and the prediction time. The best parameters found for each model are also described in the table. In order to search for the best parameters, we used the grid search library, especially for ANN (MLP) model that has many parameters needing considerable effort in tunning. The grid search process was not computed in the training time column.

For a better visualization of the results, we plot in Fig. 11 the f-score values for each model across the different scenarios. Analyzing the results, we can observe that ANN achieve the highest classification score in all scenarios, followed by KNN. In the first and second scenarios, all models achieve excellent f1-score (100 % in first scenario and above 95 % in second one where we notice a slight distinction for ANN). However, in the two other scenarios, the distinction starts to be increasingly visible in favor of ANN, especially in the last scenario where ANN achieves 79 % in the f1-score metric, whereas all other models do not exceed 59 %.

In order to evaluate the computational footprint of the models, we calculate the training and prediction times required in each scenario. The results, summarized in Fig. 12, show that CBLOF takes the longest time in the training phase followed by ANN. However, the training times of LOF and KNN are hardly noticeable in the chart. In the prediction phase, KNN takes the longest time to predict new samples, followed by CBLOF, ANN and LOF. In conclusion, the model that takes the shortest time in both training and prediction is LOF, followed by KNN, ANN and CBLOF.

3.4. Performance evaluation on an edge device

As an edge device, we use a Raspberry-Pi with the characteristics described in Table 5. Due to the fact that the edge device is a low computational resource environment, additional metrics should be involved in the evaluation. Therefore, in Table 8, we describe the performance of the ML models on the edge device and report prediction time, RAM usage and model size.

On the Raspberry-Pi, CBLOF takes the longest time to predict, followed by KNN, LOF and ANN. Generally, cluster-based methods (CBLOF) are computationally lighter than nearest-neighbor (KNN and LOF), however the distinction is only visible in very large datasets which is not our case [47]. Regarding memory, we observe that CBLOF is the model having high RAM requirement, as opposed to KNN, ANN and LOF that have close memory footprint, as it can be seen in Fig. 13.

The first conclusion to draw from the results is regarding the supervision type. It is essential to emphasize that supervised and

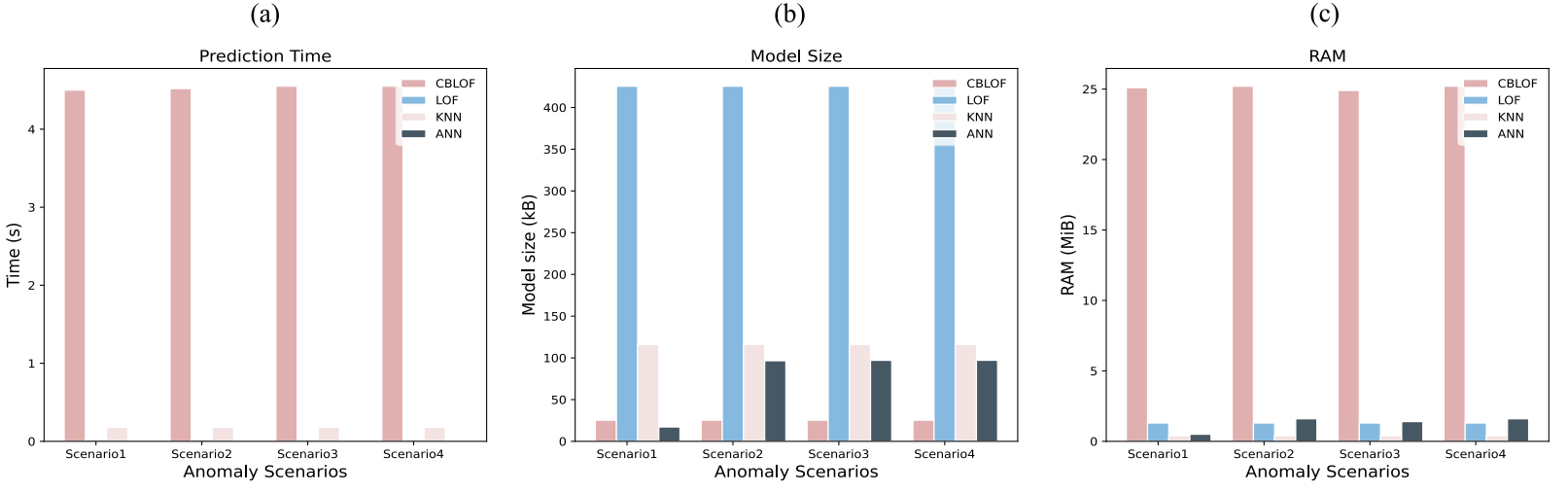


Fig. 13. Classification time, RAM usage and model size performance for each ML model across the different scenarios evaluated (a) Overview of time performance during the prediction phase (b) Overview of model size performance across the evaluated scenarios (c) Overview of RAM usage across the evaluated scenarios.

unsupervised techniques are different and hardly comparable tasks [46]. In fact, unsupervised models, even with no prior training achieved better performances than the supervised ones. Moreover, the supervised models had access to labelled data as opposed to unsupervised ones. According to the results, and despite being considered the hardest task, unsupervised models were the best performers. In fact, in all the scenarios above, the supervised model ANN was superior in only the last scenario, which clearly needed more training to be able to detect this type of anomaly considered the hardest among the four scenarios.

The second conclusion is related to the suitable model to be deployed in the edge architecture. As highlighted in the results, ANN was the best model in terms of accuracy followed by KNN. Considering that KNN is an unsupervised model that scored similar to ANN in 3 out of 4 scenarios without prior training while achieving the least RAM allocation, we can conclude that KNN is the most suitable unsupervised model.

4. Limitations and perspectives of the study

In this study, an architectural framework has been devised for the purpose of monitoring the state of decentralized photovoltaic (PV) installations. The case study focuses specifically on a solar campus. One of the limitations inherent in the study pertained to the unaddressed concerns of security and privacy. The implementation of edge devices throughout the smart city presents additional difficulties with regard to susceptibility to cyber-attacks. Hence, the primary aim of our future work is to increase the reliability of the edge devices by the incorporation of an additional level of privacy utilizing the federated learning methodology. This particular strategy will be further elaborated in our future research.

5. Conclusions

The goal of this paper is to present a framework for condition monitoring in decentralized photovoltaic systems using an edge computing framework and extending it to a smart city environment. An experimental evaluation has been conducted using a dataset collected from a Smart Campus where multiple PV systems are used to generate clean energy for laboratory R&D purposes.

The key results of this study are summarized below.

- A comparative evaluation has been performed on multiple machine learning models from various families and supervision types. The models underwent evaluation using a genuine dataset containing synthetic anomalies. The evaluation of the performance involved the utilization of commonly used measures, along with additional ones that are particularly relevant in the context of edge computing. The K-nearest neighbors (KNN) model was determined to be the most effective in the unsupervised scenario, whereas the artificial neural network (ANN) demonstrated superior performance in the supervised scenario. This finding demonstrates that unsupervised models can be effectively utilized in certain anomaly circumstances without the need for pre-training, resulting therefore in efficient and rapid detection.
- An edge-based framework specifically designed for smart cities was presented and demonstrated within the context of a smart campus. The framework presents a novel approach that integrates edge, fog, and cloud layers, as well as online and offline configurations, to identify anomalies effectively and rapidly. The framework elucidates the benefits of edge computing in the context of condition monitoring in smart cities, specifically emphasizing the reduced latency, decreased bandwidth requirements, and lowered costs.

Finally, future work will be dedicated to the hardware implementation of the models on an embedded system connected in real-time with the PV testbed to physically assess the complexity of each of the candidate models on other configurations and types of anomalies.

Data availability statement

The data that has been used is confidential.

CRediT authorship contribution statement

Ibtihal Ait Abdelmoula: Writing – original draft, Methodology, Data curation, Conceptualization. **Samir Idrissi Kaitouni:** Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Nassim Lamrini:** Writing – original draft, Methodology, Formal analysis, Data curation. **Mourad Jbene:** Investigation, Data curation. **Abdellatif Ghennioui:** Validation, Supervision, Investigation. **Adil Mehdary:** Writing – review & editing, Validation, Supervision, Methodology. **Mohamed El Aroussi:** Writing – review & editing, Validation, Project administration, Methodology, Investigation.

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] M. Farsi, A. Daneshkhah, A. Hosseinian-Far, et H. Jahankhani, « Internet of Things Digital Twin Technologies and Smart Cities ». [En ligne]. Disponible sur: http://www.springer.com/series/11636.

[2] M.L. Di Silvestre, S. Favuzza, E. Riva Sanseverino, G. Zizzo, How Decarbonization, Digitalization and Decentralization are changing key power infrastructures, Renew. Sustain. Energy Rev. 93 (February) (2018) 483–498, https://doi.org/10.1016/j.rser.2018.05.068.

[3] W. Serrano, Digital systems in smart city and infrastructure: digital as a service, Smart Cities 1 (1) (2018) 134–154, https://doi.org/10.3390/smartcities1010008.

[4] A.A. Cook, G. Misirli, Z. Fan, Anomaly detection for IoT time-series data: a survey, IEEE Internet Things J. 7 (7) (2020) 6481–6494, https://doi.org/10.1109/JIOT.2019.2958185.

[5] A. Sgueglia, A. Di Sorbo, C.A. Visaggio, G. Canfora, A systematic literature review of IoT time series anomaly detection solutions, Future Generat. Comput. Syst. 134 (2022) 170–186, https://doi.org/10.1016/j.future.2022.04.005.

[6] L. Erhan, et al., Smart anomaly detection in sensor systems: a multi-perspective review, Inf. Fusion 67 (2021) 64–79, https://doi.org/10.1016/j.inffus.2020.10.001, mars.

[7] Z. Chen, L. Wu, S. Cheng, P. Lin, Y. Wu, W. Lin, Intelligent fault diagnosis of photovoltaic arrays based on optimized kernel extreme learning machine and I-V characteristics, Appl. Energy 204 (2017) 912–931, https://doi.org/10.1016/j.apenergy.2017.05.034.

[8] D.M. Shila, K.G. Lore, T. Wei, T. Lovetty, Y. Cheng, « Catching anomalous distributed photovoltaics: an edge-based multi-modal anomaly detection », CoRR abs/1709 (2017), 08830 [En ligne]. Disponible sur: http://arxiv.org/abs/1709.08830.

[9] Z. Wang, J. Tian, H. Fang, L. Chen, J. Qin, LightLog: a lightweight temporal convolutional network for log anomaly detection on the edge, Comput. Network. 203 (2022), 108616, https://doi.org/10.1016/j.comnet.2021.108616.

[10] A. Huč, J. Šalej, M. Trebar, Analysis of machine learning algorithms for anomaly detection on edge devices, Sensors 21 (14) (2021), https://doi.org/10.3390/s21144946.

[11] M.F. Alati, G. Fortino, J. Morales, J.M. Cecilia, P. Manzoni, Time series analysis for temperature forecasting using TinyML, in: 2022 IEEE 19th Annual Consumer Communications & Networking Conference, CCNC), 2022, pp. 691–694, https://doi.org/10.1109/CCNC49033.2022.9700573.

[12] F. Delussu, D. Manzione, R. Meo, G. Ottino, M. Asare, Experiments and comparison of digital twinning of photovoltaic panels by machine learning models and a cyber-physical model in modelica, IEEE Trans. Ind. Inf. 18 (6) (2022) 4018–4028, https://doi.org/10.1109/TII.2021.3108688.

[13] A. Livera et al., « Intelligent Cloud-Based Monitoring and Control Digital Twin for Photovoltaic Power Plants », p. 9.

[14] S. Hempelmann, et al., Evaluation of unsupervised anomaly detection approaches on photovoltaic monitoring data, in: Conference Record of the IEEE Photovoltaic Specialists Conference, 2020-June, 2020, pp. 2671–2674, https://doi.org/10.1109/PVSC45281.2020.9300481.

[15] C.B. Jones, A.R. Chavez, R. Darbali-Zamora, S. Hossain-McKenzie, Implementation of intrusion detection methods for distributed photovoltaic inverters at the grid-edge, in: 2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference, ISGT), 2020, pp. 1–5, https://doi.org/10.1109/ISGT45199.2020.9087756.

[16] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, X. Hu, Revisiting time series outlier detection: definitions and benchmarks, in: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1), 2021 [En ligne]. Disponible sur: https://openreview.net/forum?id=r8IvOsnHchr.

[17] L. Zhang, et al., A review of machine learning in building load prediction, Appl. Energy 285 (2021), 116452, https://doi.org/10.1016/j.apenergy.2021.116452 mars.

[18] D. Adhya, S. Chatterjee, A.K. Chakraborty, Diagnosis of PV array faults using RUSBoost, J Control Autom Electr Syst 34 (1) (2023) 157–165, https://doi.org/10.1007/s40313-022-00947-6.

[19] J. Van Gompel, D. Spina, C. Develder, Satellite based fault diagnosis of photovoltaic systems using recurrent neural networks, Appl. Energy 305 (2022), 117874, https://doi.org/10.1016/j.apenergy.2021.117874.

[20] J. Van Gompel, D. Spina, C. Develder, Cost-effective fault diagnosis of nearby photovoltaic systems using graph neural networks, Energy 266 (2023), 126444, https://doi.org/10.1016/j.energy.2022.126444.

[21] M. Hajji, Z. Yahyaoui, M. Mansouri, H. Nounou, M. Nounou, Fault detection and diagnosis in grid-connected PV systems under irradiance variations, Energy Rep. 9 (2023) 4005–4017, https://doi.org/10.1016/j.egyr.2023.03.033.

[22] A. Latoui, M.E.H. Daachi, Real-time monitoring of partial shading in large PV plants using Convolutional Neural Network, Sol. Energy 253 (2023) 428–438, https://doi.org/10.1016/j.solener.2023.02.041.

[23] M.M. Badr, et al., Intelligent fault identification strategy of photovoltaic array based on ensemble self-training learning, Sol. Energy 249 (2023) 122–138, https://doi.org/10.1016/j.solener.2022.11.017.

[24] M. Hojabri, S. Kellerhals, G. Upadhyay, B. Bowler, IoT-based PV array fault detection and classification using embedded supervised learning methods, Energies 15 (6) (2022) 2097, https://doi.org/10.3390/en15062097.

[25] S. Sairam, S. Seshadhri, G. Marafioti, S. Srinivasan, G. Mathisen, K. Bekiroglu, Edge-based explainable fault detection systems for photovoltaic panels on edge nodes, Renew. Energy 185 (2022) 1425–1440, https://doi.org/10.1016/j.renene.2021.10.063.

[26] A.R. Sajun, S. Shapsough, I. Zualkernan, R. Dhaouadi, « Edge-Based Individualized Anomaly Detection in Large-Scale Distributed Solar Farms », ICT Express, 2022 https://doi.org/10.1016/j.icte.2021.12.011.

[27] M. Dong, J. Zhao, D. Li, B. Zhu, S. An, Z. Liu, ISEE: industrial Internet of Things perception in solar cell detection based on edge computing, Int. J. Distributed Sens. Netw. 17 (11) (2021), 15501477211050552, https://doi.org/10.1177/15501477211050552.

[28] K.V.G. Raghavendra, N.T.U. Kumar, W. Kazim, An efficient optical inspection of photovoltaic modules deploying edge detectors and ancillary techniques, Int. J. Electr. Comput. Eng. 12 (5) (2022) 4772.

[29] A. Mellit, An embedded solution for fault detection and diagnosis of photovoltaic modules using thermographic images and deep convolutional neural networks, Eng. Appl. Artif. Intell. 116 (2022), 105459, https://doi.org/10.1016/j.engappai.2022.105459.

[30] A. Mellit, M. Benghanem, S. Kalogirou, A. Massi Pavan, An embedded system for remote monitoring and fault diagnosis of photovoltaic arrays using machine learning and the internet of things, Renew. Energy 208 (2023) 399–408, https://doi.org/10.1016/j.renene.2023.03.096.

[31] « Green Energy Park », Consulté le: 8 janvier, https://www.greenenergypark.ma/, 2023.

[32] Y. Zhao, Z. Nasrullah, Z.P. Li, « A python Toolbox for Scalable Outlier Detection, 2019 arXiv 2019 », *arXiv preprint arXiv:1901.01588*.

[33] K.-H. Lai, D. Zha, J. Xu, Y. Zhao, G. Wang, X. Hu, Revisiting time series outlier detection: definitions and benchmarks, in: présenté à Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1), 2022. Consulté le: 11 octobre 2022. [En ligne]. Disponible sur: https://openreview.net/forum?id=r8IvOsnHchr.

[34] A. Livera et al., « Intelligent Cloud-Based Monitoring and Control Digital Twin for Photovoltaic Power Plants », p. 9.

[35] C.U. Carmona, F.-X. Aubet, V. Flunkert, J. Gasthaus, « Neural Contextual Anomaly Detection for Time Series », arXiv, 2021 https://doi.org/10.48550/ARXIV.2107.07702.

[36] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, SIGMOD Rec 29 (2) (2000) 93–104, https://doi.org/10.1145/335191.335388.

[37] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, SIGMOD Rec 29 (2) (2000) 427–438, https://doi.org/10.1145/335191.335437.

[38] Z. He, X. Xu, S. Deng, Discovering cluster-based local outliers, Pattern Recogn. Lett. 24 (9) (2003) 1641–1650, https://doi.org/10.1016/S0167-8655(03)00003-5.

[39] M.W. Gardner, S.R. Dorling, Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences, Atmos. Environ. 32 (14) (1998) 2627–2636, https://doi.org/10.1016/S1352-2310(97)00447-0.

[40] Q.N. Minh, V.-H. Nguyen, V.K. Quy, L.A. Ngoc, A. Chehri, G. Jeon, Edge computing for IoT-Enabled smart grid: the future of energy, Energies 15 (17) (2022), https://doi.org/10.3390/en15176140.
[41] C. Feng, Y. Wang, Q. Chen, Y. Ding, G. Strbac, C. Kang, Smart grid encounters edge computing: opportunities and applications, Advances in Applied Energy 1 (2021), 100006, https://doi.org/10.1016/j.adapen.2020.100006.
[42] C. Feng, Y. Wang, Q. Chen, Y. Ding, G. Strbac, C. Kang, Smart grid encounters edge computing: opportunities and applications, Advances in Applied Energy 1 (2021), 100006, https://doi.org/10.1016/j.adapen.2020.100006.
[43] F. Samie, L. Bauer, J. Henkel, Edge computing for smart grid: an overview on architectures and solutions, in: K. Siozios, D. Anagnostos, D. Soudris, et E. Kosmatopoulos (Eds.), IoT for Smart Grids: Design Challenges and Paradigms, Springer International Publishing, Cham, 2019, pp. 21–42, https://doi.org/10.1007/978-3-030-03640-9_2.
[44] H.A. Imran, U. Mujahid, S. Wazir, U. Latif, K. Mehmood, « Embedded development boards for edge-AI: a comprehensive report », CoRR abs/2009 (2020), 00803 [En ligne]. Disponible sur: https://arxiv.org/abs/2009.00803.
[45] A. Bakumenko, A. Elragal, Detecting anomalies in financial data using machine learning algorithms, Systems 10 (5) (2022) 130, https://doi.org/10.3390/systems10050130.
[46] F. Cavallin, R. Mayer, Anomaly detection from distributed data sources via federated learning, in: L. Barolli, F. Hussain, et T. Enokido (Eds.), Advanced Information Networking and Applications, Springer International Publishing, Cham, 2022, pp. 317–328.
[47] M. Amer et M. Goldstein, « Nearest-Neighbor and Clustering Based Anomaly Detection Algorithms for RapidMiner ».