Sustainable Computing: Informatics and Systems 37 (2023) 100842

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

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

SUSAN: A Deep Learning based anomaly detection framework for sustainable industry

Ángel Luis Perales Gómez [a,*], Lorenzo Fernández Maimó [a], Alberto Huertas Celdrán [b], Félix J. García Clemente [a]

[a] Departamento de Ingeniería y Tecnología de Computadores, University of Murcia, 30100 Murcia, Spain
[b] Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH, CH 8050, Zürich, Switzerland

ARTICLE INFO

Keywords:
Anomaly detection
Deep learning
Industrial control systems
Machine learning
Sustainability

ABSTRACT

Nowadays, sustainability is the core of green technologies, being a critical aspect in many industries concerned with reducing carbon emissions and energy consumption optimization. While this concern increases, the number of cyberattacks causing sustainability issues in industries also grows. These cyberattacks impact industrial systems that control and monitor the right functioning of processes and systems. Furthermore, they are very specialized, requiring knowledge about the target industrial processes, and being undetectable for traditional cybersecurity solutions. To overcome this challenge, we present SUSAN, a Deep Learning-based framework, to build anomaly detectors that expose cyberattacks affecting the sustainability of industrial systems. SUSAN follows a modular and flexible design that allows the ensembling of several detectors to achieve more precise detections. To demonstrate the feasibility of SUSAN, we implemented the framework in a water treatment plant using the SWaT testbed. The experiments performed achieved the best recall rate (0.910) and acceptable precision (0.633), resulting in an F1-score of 0.747. Regarding individual cyberattacks that impact the system's sustainability, our implementation detected all of them, and, concerning the related work, it achieved the most balanced results, with 0.64 as the worst recall rate. Finally, a false-positive rate of 0.000388 makes our solution feasible in real scenarios.

1. Introduction

Green technologies are playing an increasingly significant role in changing the course of socio-economic world growth. In this revolution, sustainability is crucial to allow present and future generations to live in a prosperous and healthy environment. Different industrial sectors, such as energy, agriculture, or water treatment, are progressively changing the production processes of their industrial plants to produce goods and services in a sustainable fashion. In this context, the reduction of carbon emissions, the optimization of energy consumption, the purification of water resources, or recycling processes are just a few examples of the impact that green technologies have on our society.

To cope with the challenge of optimizing resources and producing goods sustainably, the evolution of technology has reached the industrial area and influenced the rising of the fourth stage of industrialization, also known as Industry 4.0 [1]. In this setting, the next generation of mobile networks (5G), the big data, or the Internet of Things (IoT), bring immense opportunities for the realization of sustainable manufacturing [2]. 5G, for example, enables millions of devices connected simultaneously with minimum latency and high bandwidth, the big data analyze large amounts of information in just a few seconds to make intelligent decisions, and IoT connects heterogeneous and resource-constrained devices to ease the daily life of citizens.

However, the integration of the previous technologies and paradigms in the industry also opens the door to new cyberattacks concerning the sustainability of industrial processes [3]. In this respect, despite most of the reported cyberattacks affecting the industry and its industrial control systems (ICS) have been focused on disrupting supplied services or stopping the production of goods, they can also affect the sustainability of industrial processes. Over the years, several examples of these cyberattacks have been documented in the literature, especially those that affect the water [4] and energy [5] sectors. One of the latest cyberattacks affecting sustainability in water supply happened on May 29th, 2019. In this cyberattack, a small city of 35,000 inhabitants located in Florida was hit by a ransomware after an employee of the police department opened an infected email. This ransomware spread quickly in the local government network affecting

* Corresponding author.
E-mail addresses: angelluis.perales@um.es (A.L. Perales Gómez), lfmaimo@um.es (L.F. Maimó), huertas@ifi.uzh.ch (A.H. Celdrán), fgarcia@um.es (F.J.G. Clemente).

https://doi.org/10.1016/j.suscom.2022.100842
Received 6 July 2021; Received in revised form 10 November 2022; Accepted 31 December 2022
Available online 5 January 2023

the local water utility, comprising computer systems controlling pumping stations and water quality testing. In the field of energy supply, the cyberattacks that took place in Ukraine in 2015–2016 were aimed at affecting the sustainability of power-grids processes. In particular, these cyberattacks, performed by a highly skilled group of hackers, were capable of causing a power outage in three electricity distribution companies, cutting out the power for more than 200,000 customers.

Traditional cybersecurity systems used to detect intrusions (IDS) are no longer efficient because they rely on signatures stored in datasets. Cyberattackers are aware of this and modify the cyberattack vectors to ensure that their signatures are different from those stored in the databases. Besides, traditional solutions are not capable of detecting new families or types of unseen cyberattacks. Nowadays, the current trend to avoid this problem is the utilization of Anomaly Detection (AD) systems based on machine learning (ML) and deep learning (DL) techniques [6–8]. However, these techniques are not without problems. For example, ML and DL models are vulnerable to evasion attacks [9]. These attacks consist in modifying an abnormal sample that will be misclassified by the IDS, reaching the industrial device and having an effect on the physical world.

In this context, we identified several open challenges in the ICS field because existing anomaly detectors focus on identifying service disruptions or malfunctions [10,11] and they miss the relevance of sustainability in the industrial processes. The main reason for this limitation is the set of features considered in traditional anomaly detectors based on ML and DL techniques. To be precise, most of the current solutions consider the original features in the dataset without using a feature extraction process to generate high-order features that discriminate between different behaviors in the ICS. Furthermore, the design of existing solutions is focused on general cyberattacks and does not minimize the impact of cyberattacks hitting system's sustainability. The main reason for treating cyberattacks that affect sustainability independently from others is that the AD-based IDS can focus on the specific sustainability properties. Therefore, this approach allows the detection of a larger number of such cyberattacks than a general AD-based IDS. This aspect is crucial since sustainability will become a key aspect in the upcoming years [12]. In fact, some authors point out a lack of sustainability in Industry 4.0 projects [13]. Therefore, it is necessary to devote more effort to developing solutions that improve the sustainability of factories.

In order to meet the previous challenges, the contributions of the current paper can be summarized in the following points:

- A framework, called SUSAN, for building detectors of anomalies produced by cyberattacks that affect the sustainability of industrial processes. SUSAN-based anomaly detectors use DL techniques and are flexible enough to be integrated with other detectors specialized in other types of anomalies.
- A set of sustainability-based features that gather the resource consumption routines for modeling the behavior of industrial sensors and actuators related to sustainability. Cyberattacks impacting the resource consumption could affect those features and, thus, the sustainability of some industrial processes.
- Validation of the proposal using the well-known Secure Water Treatment testbed, SWaT, which was impacted by several cyberattacks affecting the sustainability of different water treatment processes.

The remainder of this manuscript is structured as follows. Section 2 reviews the state of the art in anomaly detection techniques that detect cyberattacks affecting the sustainability of cloud infrastructures, smart buildings, and ICS environments. The motivating example of Section 3 analyzes the cyberattacks affecting the sustainability of SWaT and justifies the need for an anomaly detector capable of identifying sustainability anomalies. In Section 4, we present the design and components of SUSAN as well as how to integrate a SUSAN-based anomaly detector with others. The set of sustainability-based features proposed to detect the cyberattacks are detailed in Section 5. In Section 6, we analyze the performance of a SUSAN-based anomaly detector in the SWaT testbed. In Section 7, we discuss the framework presented among its advantages and limitations. Finally, the conclusions and future work are presented in Section 8.

2. Related work

This section reviews the state-of-the-art solutions aimed at detecting anomalies generated by cyberattacks affecting the correct functioning and resource consumption of different computational systems ranging from cloud general computing infrastructures to particular ICS environments.

2.1. Anomaly detection affecting the energy sustainability of smart buildings

In the field of smart buildings, there is a significant number of solutions that detect anomalies affecting energy consumption and, therefore, building sustainability. Miller et al. [14] reviewed solutions dealing with sustainability in non-residential buildings. They focused on unsupervised statistical learning techniques, and the most important reviewed families were: clustering, novelty detection (where anomaly detection is included), motif and discord detection, rule extraction, and visual analytics. The authors also analyzed the technologies used in each of the previous families, emphasizing smart meters, portfolio analysis, operations, controls optimization, and anomaly detection.

In novelty detection, Araya et al. [15] proposed a contextual anomaly detection framework that used sliding time windows to aggregate data related to energy consumption and contextual features. Some of the most relevant features considered in this work were the mean and median of data values monitored by sensors in each time window, the standard deviation of data in each window, and differences between the elements of one and several windows, among others. Moreover, autoencoder (AE) was used to detect typical consumption patterns and anomalies. Fan et al. [16] proposed another solution that uses several AE architectures and training schemes to detect anomalies in the energy consumption of buildings. The outputs of this work demonstrated that AE are an alternative when extracting high-level features to detect energy consumption anomalies. AE provided good performance while alleviating the data preprocessing workload.

Tasfi et al. [17] proposed a deep semi-supervised Convolutional Neural Network (CNN) with confidence sampling to detect anomalies in the electrical consumption of Heating, Ventilating, and Air Conditioning (HVAC), lighting, and heat pumps. The authors used two networks, the first one aimed at the reconstruction and usage of unlabeled data, while the second one used the labeled data to classify anomalies. To quantify anomaly detection confidence, a valuable metric in anomaly detection, the network used a dropout sampling method. The accuracy of the proposed approach was evaluated and demonstrated with real-world electrical data from systems such as HVAC, lighting, and heat pumps.

In conclusion, as in the previous section, the solutions presented in this section focused on the detection of anomalies that affect sustainability in the smart building context. Thus, the features computed in the previous approaches do not exploit the repetitive nature of ICS. Therefore, they do not help to detect cyberattacks affecting sustainability in an industrial context.

2.2. Detection of anomalies caused by cyberattacks in ICS

The AD paradigm is gaining prominence in the detection of cyberattacks in industrial scenarios [8]. In general, AD in ICS is frequently treated as a time-series classification problem due to its repetitive nature. In the literature, many existing solutions rely on techniques

dealing with the time dimension, such as CNN and LSTM Neural Network.

By way of example, Kravchik and Shabtai [18] considered LSTM and 1-Dimensional CNN (1D-CNN) to detect anomalies in ICS. New features dealing with the relationships of current and past values were proposed by the authors as well. Some experiments concluded that 1D-CNN obtains a better performance and its training is faster than LSTM. Following the same direction, Shalyga et al. [19] proposed an automatic optimization architecture based on Genetic Algorithms (GA) that tested Dense Neural Network (DNN), 1D-CNN, and LSTM networks. Another solution was proposed by Zizzo et al. [20], where an IDS based on LSTM neural network was presented. The Cumulative Sum (CUSUM) method, computed from the residual error calculated from the ground truth and model predictions, was considered to detect anomalies. Li et al. [21] proposed a Generative Adversarial Network-based Anomaly Detection (GAN-AD) for ICS scenarios. The authors implemented an LSTM network in the core of the GAN-AD to capture the distribution of the multivariate time series of sensors and actuators under normal conditions in an ICS. Kim et al. [22] also used LSTM networks, presenting a Sequence-to-Sequence LSTM neural network for anomaly detection. The LSTM network was trained using data under normal conditions and used attention mechanisms.

Apart from the previous solutions, which are based on LSTM and CNN, other DL and ML techniques have been proposed to detect anomalies in ICS. For example, Grammatikis et al. [23] proposed an AD system for IEC 60870-5-104 protocol which is commonly used in industrial settings. Before implementing the AD system, the authors studied the protocol and its vulnerabilities. The solution monitored the network traffic and applied filters based on a whitelist in which all legitimate Medium Access Control (MAC) and Internet Protocol (IP) addresses were stored. Then, the solution extracted flows features that were evaluated by three different ML models. In particular, the models were One-Class Support Vector Machine (OC-SVM), Local Outlier Factor (LOF), and Isolation Forest (IF), reaching the latter the best result in terms of the F1-score. Khraisat et al. [24] presented a novel ensemble, named Hybrid Intrusion Detection System (HIDS), to protect sensors and actuators. The approach combined C5 Decision Tree (C5 DT) classifier and OC-SVM. The proposed ensemble two different paradigms. In the first stage, the solution employed the Signature Intrusion Detection System paradigm by means of the C5 DT classifier. The rules of the classifier were applied to test if a sample was normal or abnormal. If a particular sample matched an attack rule, the HIDS raises an alarm. Otherwise, the sample went to the Anomaly-based Intrusion Detection System stage, where the OC-SVM was used to determine if the sample was normal or abnormal. Another work is presented by Caselli et al. [25], where they discussed the possibility of a sequence of events attack. The authors state that samples in industrial environments must be evaluated in a sequence fashion since a particular valid sequence of events samples can elude the detection system and damage the industrial infrastructure. Therefore, the authors proposed a sequence-aware intrusion detection system based on Discrete-Time Markov Chains (DTMC) that was validated using data from a realistic scenario. Inoue et al. [26] designed and evaluated an application that used unsupervised ML to detect anomalies in Cyber–Physical Systems (CPS). They compared Deep Neural Network (DNN) and Support Vector Machine (SVM), both adapted to work with time-series data. Gómez et al. [27] proposed a methodology to generate reliable anomaly detection datasets by selecting and deploying cyberattacks, capturing their traffic and computing features. The Electra dataset, which contains cyberattacks affecting an electric traction substation used in the railway industry, demonstrated the usefulness of the proposed methodology. Finally, several experiments with Random Forest (RF), SVM, OCSVM, IF, and DNN measured the Electra relevance. Kravchik and Shabtai [28] proposed a DL solution based on AE and 1D-CNN. The authors filtered features to select those more suitable for the anomaly detection task. Finally, the authors proposed a feature extraction procedure that computes features on the frequency domain using the Discrete Fourier Transform (DFT). Elnour et al. [29] presented a novel semi-supervised architecture based on Dual Isolation Forest (DIF), which uses two IF models. One of these IF is trained using the normalized raw data, while the other is trained using a preprocessed version of the data with PCA. Both models were trained using the normal samples from the training and test dataset, while abnormal samples were used to create the test dataset.

It is worth mentioning that, although it is less common due to its drawbacks, some authors also proposed to detect cyberattacks in an industrial environment employing a signature-based approach. For example, Ghaeini and Tippenhauer [30] proposed a hierarchical monitoring intrusion detection system (HAMIDS) for industrial environments. The approach aimed at detecting anomalies at the lower network levels in an industrial plant. To do so, the data gathered from the sensors and actuators were aggregated in one point for analysis in further stages. The proposal was implemented as an extension of the Bro tool, which is a powerful tool for security monitoring and network traffic analysis. In this context, the intrusion detection proposed by the authors makes use of the Bro tool and raises an alarm when a pattern stored in a database is matched.

Table 1 details some of the most representative proposals in the literature and compares them with the proposed framework. As can be seen, there is no anomaly detector in the ICS field focused on detecting anomalies that affect the sustainability of industrial processes and plants.

In conclusion, this section has revealed the lack of solutions for detecting anomalies caused by cyberattacks that affect energy consumption and sustainability of ICS scenarios. Specifically, we have seen that existing solutions focused on detecting anomalies in ICS scenarios do not consider the sustainability of ICS. They based on metrics and characteristics related to the proper functioning of the industrial plant.

3. Case study: Impact of cyberattacks on sustainability in industrial control systems

This section motivates the added value and importance of our proposal. For this purpose, it uses the well-known Secure Water Treatment testbed (SwaT) [31], which is a fully operational scaled-down water treatment plant with a small carbon footprint, producing five gallons/minute of doubly filtered water. This testbed replicates large modern plants for water treatment, such as those found in cities.

The SWaT testbed is illustrated in Fig. 1 and consists of six processes that treat the water for its distribution. Process 1 (P1) is responsible for keeping the raw water tank with enough water to supply the other processes. Process 2 (P2) pre-treats the water to ensure acceptable levels of quality. If the raw water tank in P1 is not of a satisfactory quality (measured by sensors AIT201, AIT202, and AIT203), the pumps (P201, P203, and P205) are opened, and chemical dosing is applied to correct water deficiencies. Once the water is of an appropriate quality, it flows to Process 3 (P3). At this stage, the undesired materials remaining in the water are removed through an Ultrafiltration (UF) system by using a filtration membrane. During Process 4 (P4), any remaining chlorine is removed by a dechlorination process that uses Ultraviolet (UV) lamps. The next stage consists in reducing the inorganic impurities present in the water. To accomplish this task, the dechlorinated water is pumped into the Reverse Osmosis (RO), which represents Process 5 (P5). Finally, Process 6 (P6) leads to the water being made available for its distribution.

From the sustainability point of view, SWaT has an optimal water treatment process. This means that, under normal conditions, resources (chemical and power consumption) are used in an optimal fashion. If any external factor, such as a cyberattack, alters resource consumption, there may be two consequences. On the one hand, lower resource consumption can result in poor water quality, making it useless for distribution and, therefore, demanding a new water treatment. On the

Table 1
Comparison of solutions that detect anomalies affecting sustainability.

Sol.	Scenario	Sustainability	Cyberattack	Techniques	Features
Araya et al. [15] and Fan et al. [16]	Smart buildings	✓	×	AE	Statistical data from building sensors
Tasfi et al. [17]	Smart buildings	✓	×	CNN	Statistical data from HVAC, lightning and heat pumps
Kravchik and Shabtai [18]	ICS	×	✓	LSTM and CNN	Current and historical data from ICS sensors
Shalyga et al. [19]	ICS	×	✓	GA, DNN, 1D-CNN and LSTM	ICS sensors time series
Li et al. [21]	ICS	×	✓	GAN-AD and LSTM	Time series of sensor and actuators
Grammatikis et al. [23]	ICS	×	✓	OC-SVM, LOF, and IF	Flow features
Khraisat et al. [24]	ICS	×	✓	C5 DT and OC-SVM	Flow features
Caselli et al. [25]	ICS	×	✓	DTMC	Network packet, log files, and process variables
Inoue et al. [26]	ICS	×	✓	DNN and SVM	Time-series data from ICS sensors
Gómez et al. [27]	ICS	×	✓	RF, SVM, OCSVM, IF and DNN	Time-series data from ICS sensors
Elnour et al. [29]	ICS	×	✓	DIF	Raw and PCA processed data from ICS sensors
Current proposal	ICS	✓	✓	LSTM	Statistical and time-series data from ICS sensors

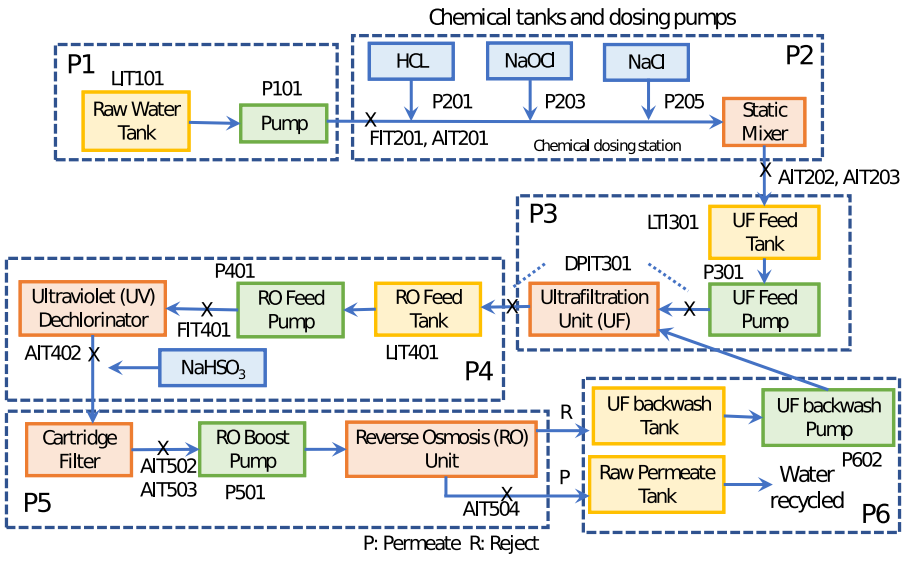

Fig. 1. SWaT testbed and its processes.

other hand, high resource consumption can give rise to two issues: (1) resource wastage, causing monetary losses; (2) over-processed water, affecting its quality for distribution and human consumption.

In the SWaT testbed, a total of 41 cyberattacks was deployed. Most of these cyberattacks were focused on compromising the system cybersecurity, trying to interrupt the service, or producing water at a lower rate than required. However, among the cyberattacks launched, we identified 9 cyberattacks (numbers 6, 11, 19, 20, 22, 24, 28, 38 and 40 in [31]) compromising the sustainability of the system. More specially, these cyberattacks targeted the sensors or actuators managing chemical dosing or UV lamps. These cyberattacks are detailed below.

- Cyberattack 6. This cyberattack targets AIT202, a sensor that measures the HCl level in P2. The cyberattack consists in setting the value of this sensor to a fixed value of 6 for 4 min. The goal of this cyberattack is to decrease water quality.
- Cyberattack 11. It affects the FIT401 sensor and sets its value to a fixed value of 0 for 9 min. The cyberattack goal is to shut down the UV actuator, causing a reduction in energy consumption and resulting in water not suitable for distribution.
- Cyberattack 19. Its objective is the AIT504 chemical sensor, in charge of measuring the NaCl level of the permeate tank in P6. The cyberattack sets the value of the AIT504 sensor to a fixed

value of 16 for 4 min. The goal of this cyberattack is to shut down the RO process and drain water.
- Cyberattack 20 is similar to cyberattack 19. It also affects the AIT504 sensor and has the same goal, but this time the value of the AIT504 sensor is set to a fixed value of 255 for 6 min.
- Cyberattack 22. It affects multiple sensors and actuators. In particular, the UV lamps actuator (UV401), a chemical sensor in charge of measuring NaOCl level (AIT502), and the pump sending dechlorinated water to RO (P501). This cyberattack was executed for 8 min and consisted of stopping the UV lamps, setting the value of the AIT502 sensor to a fixed value of 150, and forcing P501 to remain. The goal of this cyberattack was to damage the RO process.
- Cyberattack 24. It targets chemical actuators in charge of dosing HCl (P203) and NaOCl (P205). The cyberattack was launched for 5 min and consisted of retaining those actuators off to change the water quality.
- Cyberattack 28. It affects the second pump in P3 (P302). The cyberattack consists in keeping the pump closed during 9:30 h to stop the input flow in the first tank of P4. Ultimately, the cyberattack caused a sustainability issue by reducing resource usage.
- Cyberattack 38. It affects chemical sensors measuring the HCl and NaOCl levels in P4 (AIT402), and NaOCl in P5 (AIT502). The cyberattack was launched for 5 min and set both sensors to 260 to drain the water because of overdosing.
- Cyberattack 40. It targets the sensor that measures the water flow in the tank of P4 (FIT401). The cyberattack was launched for 5 min and consisted in setting the FIT401 sensor value to 0. It has the same goal as cyberattack 39, that is, to shutdown the UV process, and send the water to the RO process.

All the cyberattacks that affect sustainability cause an increase in water treatment resource consumption and, consequently, significantly impact the carbon footprint resulting from the process. To face this problem, systems that can detect cyberattacks causing sustainability issues must be adopted in industrial environments. However, the most modern anomaly detection systems capable of detecting specialized cyberattacks are focused on modeling the system's overall behavior (as shown in Section 2), which could miss sustainability cyberattacks or discover them too late. Moreover, we believe that both alerting on most severe anomalies and minimizing false positives are critical in anomaly detection focused on sustainability. Imagine a cyberattack on a water treatment plant that causes an increase in chemicals that are highly harmful to humans. In this case, the cyberattack needs to be detected as soon as possible to prevent human poisoning. However, the false-positive rate needs to be low so that it will not overwhelm the operator and prevent him/her from responding inappropriately.

To overcome these issues, an anomaly detection system is required that considers sustainable features such as power consumption and chemical consumption, among others. Furthermore, due to the physical nature of industrial systems, sensors and actuators experience a high correlation between them. This means that an anomaly detection system focused on sustainability must also detect cyberattacks that indirectly target sustainability. An example of this effect is produced by cyberattack 28, which does not directly affect the sensors/actuators related to sustainability. However, it stops the input water flow and, therefore, decreases resource consumption.

4. SUSAN: Sustainability-aware anomaly detection framework for ICS

This section details the design characteristics of the proposed DL-enabled framework, called SUSAN, for building detection systems of anomalies caused by cyberattacks that affect the sustainability of industrial processes. The left-hand side of Fig. 2 shows the modules and components making up SUSAN. In contrast, the right-hand side depicts the integration of a SUSAN-based anomaly detector with others oriented to different anomalies.

SUSAN is composed of four different modules that work together to build sustainability-aware anomaly detectors. These four modules are: *Data Preprocessing*, *Sustainability-based Features Generation*, *Model Generation* and *Sustainability Anomaly Detector*. The Data Preprocessing module is in charge of cleaning the data, encoding features, generating training, validation, and test datasets, and scaling the features. The Sustainability-based Features Generation module filters those features not suitable to be modeled with DL algorithms and extracts new higher-order features related to sustainability. The Model Generation module allows generating the model that will predict anomalies. Finally, the Sustainability Anomaly Detector module is responsible for making predictions on new unseen data handling the previously generated model. The different modules are executed automatically by SUSAN. However, in certain situations, e.g., to decide the features to remove, the operator/administrator intervention is necessary. Table 2 shows the symbols and their definitions used in this section.

4.1. Data preprocessing

The Data Preprocessing module prepares the data for DL techniques. It consists of the following components: *Data Cleaning*, *Features Encoding*, *Dataset Generation*, and *Features Normalization*.

The Data Cleaning component receives raw data from existing databases and cleans them. These databases store the historical values monitored by sensors and actuators deployed in the industrial scenario. Each value is labeled to indicate if it comes from a standard system behavior or if a cyberattack produces it. The type of sensor and actuator can vary depending on the industrial process. In general, these databases will store values of electrical consumption, chemical usage, and other resources consumed, as well as operational information from pumps, motorized valves, and other electro-mechanical devices used in the industrial process. To clean the databases, this component explores the data and removes spurious or corrupted values. After that, it checks whether the data contain missing values and marks them as non-valid data. Other techniques such as imputation or directly removing those columns should be avoided. There are main reasons for this: firstly, the usage of imputation techniques can lead to inconsistent values in the industrial domain. One of the most frequent operations to perform imputation is the mean. However, using the mean over categorical features produces meaningless values; and secondly, because an explicit missing data value can provide relevant information about the existence of anomalies. Using imputation or removing those features can lead to missing some important information.

The Features Encoding component is in charge of transforming features to make them suitable for use with DL models. At this point, the component encodes categorical values of data by applying One-Hot Encoding (OHE), yielding a binary feature for each different categorical value. For example, consider a feature collected from an industrial pump with two possible states: open and closed. After applying OHE, this feature is replaced by two binary features representing the open and closed pump states. However, original features are maintained to process them in further modules.

The Datasets Generation component is in charge of creating the three datasets that will be used to train and validate the model, i.e., training, validation, and test datasets. In particular, this component considers that most features of an industrial control system are time-dependent, and therefore, the samples cannot be selected randomly to generate the datasets. This means that the samples included the dataset will be selected based on their temporal order. In other words, in cases where authors do not provide a partitioned dataset, the training dataset will include the first samples, the validation dataset will comprise the following samples, and the last samples will be selected as the test dataset.

Table 2
Math symbols used along Section 4 (in order of appearance in the text).

Symbol	Definition	Symbol	Definition
X	Original dataset	σ_{feat2}	Standard deviation of feature $feat2$
X'	Transformed dataset	μ_{feat1}	Mean of feature $feat1$
X_t	Original training dataset	μ_{feat2}	Mean of feature $feat2$
$feat1_i$	A specific value of feature $feat1$	σ^2	Variance
$feat2_i$	A specific value of feature $feat2$	n	Number of samples
x	A specific value smaller than the number of samples, n	μ_t	Mean of each feature of the training dataset
e_{unseen}	Error of unseen samples	σ_t	Standard deviation of each feature of the training dataset
y_{unseen}	Predicted unseen values	$min()$	Minimum function
$\hat{y}_{unseen}$	Ground-truth unseen values	$max()$	Maximum function
e_n	Error normalized	$P_{feat1,feat2}$	Pearson correlation between $feat1$ and $feat2$
μ_{et}	Mean of the error in the training dataset	σ_{feat1}	Standard deviation of feature feat1
σ_{et}	Standard deviation of the error in training dataset	W	Set of all time windows ($\{w^i \mid i = 1 \dots len(x) - l + 1\}$)
F	Set of selected statistical functions ($mean$, $quartile_1$, $median$, $quartile_3$, $interquartile\ range$, $standard\ deviation$, min, max, sum and $range$)	w_j	jth element in the time window
k	Lag for autocorrelation and DFT	μ_w	Mean of the time window
l	Length of the time window	$cov(feat1, feat2)$	Covariance between $feat1$ and $feat2$
False Positive (FP)	Number of normal samples incorrectly classified	False Negative (FN)	Number of anomalous samples incorrectly classified as normal samples
True Positive (TP)	Number of anomalies correctly detected	True Negative (TN)	Number of normal samples correctly classified as normal

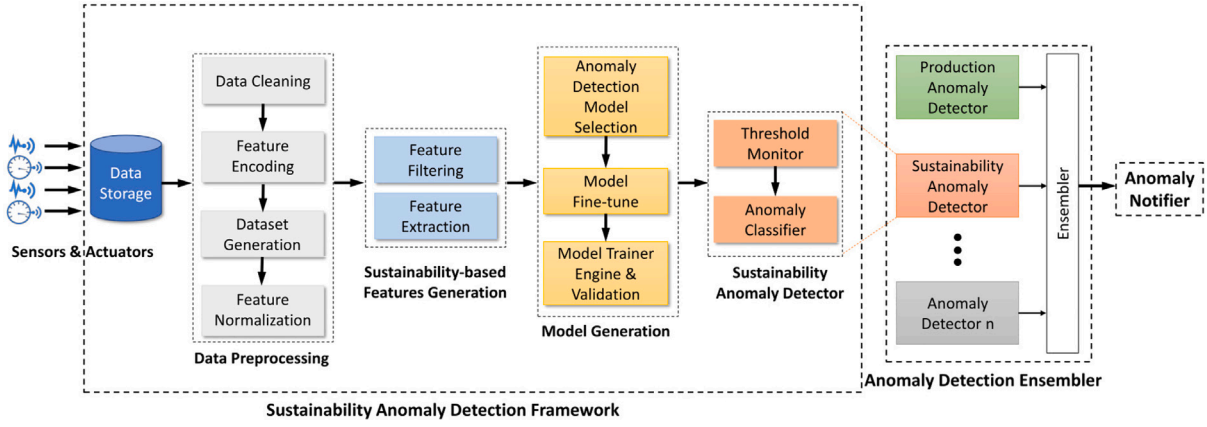

Fig. 2. Modules and components making up the SUSAN framework.

Finally, the Feature Normalization component studies the distributions of the features to perform the normalization process. If a feature follows a normal distribution, the component applies a standard normalization taking the mean and standard deviation from the training

dataset. The formal definition can be observed in Eq. (1). However, if data follow other distributions than normal, the component applies a min–max normalization. The formal definition can be seen in Eq. (2). Note that this operation needs to be performed three times, one per dataset, i.e., training, validation, and test. The goal of the previous techniques is to convert the values of the dataset that have different range values to a common scale.

$$X' = \frac{X - \mu_t}{\sigma_t} \tag{1}$$

$$X' = \frac{X - min(X_t)}{max(X_t) - min(X_t)} \tag{2}$$

4.2. Sustainability-based features generation

The Sustainability-based Features Generation module contains two components: *Feature Filtering* and *Feature Extraction*.

Feature Filtering studies the features contained in the databases and selects those most discriminative to detect sustainability anomalies. This component removes those features highly correlated with the label and constant features whose value does not change in the whole database. It is normal and even desirable in industrial systems that features have a high correlation with other features. For example, when the water quality output from a raw tank is low, the pumps solving this deficiency are open. However, the correlation between a feature and the label (indicating the presence or absence of a cyberattack) probably points to a data leakage. In order to find the correlation between each feature, the component uses the Pearson correlation [32] that is computed using the covariance and the standard deviation of the features. The formal definition of the Pearson correlation is defined in Eq. (3).

$$p_{X,Y} = \frac{cov(feat1, feat2)}{\sigma_{feat1}\sigma_{feat2}} \tag{3}$$

The covariance measures the joint variability of $feat1$ and $feat2$. If greater values for $feat1$ correspond to greater values for $feat2$ or lesser values for $feat1$ correspond to lesser values for $feat2$, the covariance is positive. Otherwise, the covariance is negative. The formal definition of covariance is defined in Eq. (4).

$$cov(feat1, feat2) = \frac{\sum_{i=0}^{n-1}(feat1_i - \mu_{feat1})(feat2_i - \mu_{feat2})}{n} \tag{4}$$

Furthermore, this component also carries out a variance study to remove those features whose values do not change in the whole dataset. The formal definition to compute the variance can be seen in Eq. (5).

$$\sigma^2 = \frac{\sum_{i=0}^{n-1}(feat1_i - \mu_{feat1})}{n} \tag{5}$$

This component also performs a study of datasets distribution to remove those features whose statistical distribution is not preserved across datasets. In particular, it uses the Kolmogorov–Smirnov (K-S) test [33] to check if two sets of continuous features follow the same statistical distribution. The K-S statistic quantifies a distance between the empirical cumulative distribution function of the sample and the cumulative distribution function of the reference distribution. The cumulative distribution function describes the probability that a feature $feat1$ with a given probability distribution will be found at a value less than or equal to $feat1_i$. The empirical distribution function is defined as shown in Eq. (6).

$$F(x) = \frac{1}{n}\sum_{i=1}^{n} 1_{(-\infty,x]}(feat1_i) \tag{6}$$

$1_{(-\infty,x]}(feat1_i)$ is determined in Eq. (7).

$$1_{[-\infty,x]}(feat1_i) = \begin{cases} 1 & \text{if } feat1_i \leq x \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

To test whether two cumulative distribution functions $F_1(feat1)$ and $F_2(feat1)$ are probably the same, the K-S statistic is defined as shown in Eq. (8).

$$D = \max_y(F_1(x) - F_2(x)) \tag{8}$$

If the features under study come from the same distribution, D is close to zero. Otherwise, the result from D will be larger depending on the dissimilarity between the feature distributions.

Finally, the Feature Extraction component extracts higher-order features from the original ones, trying to add more information to discriminate between normal and cyberattack behavior. Section 5 provides more details of the higher-order features proposed for sustainability.

4.3. Model generation

The Model Generation module receives the previous preprocessed data and creates a model capable of detecting anomalies caused by cyberattacks affecting the sustainability of industrial scenarios. The following three components make up this module: *Anomaly Detection Model Selection*, *Model Fine-tune*, and *Model Trainer Engine and Validation*.

The Anomaly Detection Model Selection component aims to choose the most suitable model to detect sustainability anomalies. SUSAN proposes a range of DL regressor models applied to a feature window because it can model more complex behavior than ML techniques. Moreover, these models are suitable to model time-series data. More specifically, they model the problem as time-series data regression instead of a predictive model based on a statistical summary window for two main reasons: (1) signals from ICS sensors and actuators follow a temporal pattern that can be modeled with a regressor; and (2) the adoption of both a regression model and a proper threshold provides information about the sensor/actuator causing the anomaly, which contributes to the desirable interpretability requirement. Besides, the interpretability requirement is usually mandatory in most anomaly detection systems in industrial environments, since it allows to know the exact point where the anomaly is produced. This knowledge allows both administrators and operators to define different mitigation schema depending on the point where the anomaly is detected. Among the options proposed by SUSAN, the administrator selects the most suitable model according to the needs. The model selected will receive time-series data from sensors and actuators $(x_0, x_1 ... x_{n-1})$ to predict the sequence of data $(x_{n+h}, x_{n+h+1}, x_{n+h+m})$ where n is the length of the sequence, m is the number of predictions made in the future, h is the prediction horizon, and $x_{0...n-1}$ as well as $y_{n+h...n+h+m}$ are the network input and output, respectively.

Once the most suitable model is chosen, the Model Fine-tune component selects a set of hyper-parameters and their respective ranges of values to be fine-tuned. The goal of this fine-tuning process is to improve the result achieved by the AD model. The hyper-parameters can be divided into two groups. The first group contains hyper-parameters related to the input data, while the second group includes those exclusively associated with the model. The latter group of hyper-parameters depends on the model and is defined once the model is set, while the first group is fixed. This component considers the following three hyper-parameters of the first group: the window length, n, the timestep in the future from which the model starts to predict, h, and the number of predicted timesteps m. It is worth mentioning that these parameters impact the time required to detect anomalies. In the worst case, SUSAN needs $n + h$ samples to detect an anomaly. Therefore, it is necessary to find a trade-off between detection and time performance. If n is increased, a more complex temporal pattern can be examined by the DL model, but the time needed to detect the anomaly will also increase.

The fine-tuned process trains the model using the training dataset with a range of hyper-parameters values to determine the optimal values. SUSAN uses *grid search* to perform an exhaustive search trying all the possible values for the hyper-parameters defined. Besides, if the

number of hyper-parameters and the range of their values is highly vast, this component can reduce the fine-tune process using the *random search* strategy that tries random values among a list of predefined values for the hyper-parameters defined. However, this approach could not find the optimal solution. Eventually, the Model Fine-tune component will select those hyper-parameters values that achieve the lowest reconstruction error over the validation dataset.

Finally, the Model Trainer Engine and Validation component trains the selected model and validates it. The training is composed of two phases. The first one finds the proper threshold to detect anomalies. For this purpose, the model is trained with the training dataset, and the threshold is found with the validation one. The threshold is the maximum of the normalized error between the ground truth and the values predicted by the model. The second phase trains the model using a new dataset composed of the training and validation dataset union. Once the model is trained, the validation phase evaluates the results achieved by the model using the test dataset. To properly validate the model, SUSAN considers the precision, recall, and F1-score metrics. These metrics are detailed in Section 6. The use of other metrics such as accuracy or the Receiver Operating Characteristic (ROC) curve is discouraged because they present critical drawbacks. On the one hand, the ROC curve needs a continuous score assigned to the samples by the classifier, whereas in our case, the model trained is a DL-based regressor that outputs not a probability but the feature values in the next timestep. Therefore, the use of ROC curve is impractical in our system. On the other hand, AD problems are usually highly imbalanced, and accuracy does not objectively show the performance of the AD system.

4.4. Sustainability anomaly detector

The Sustainability Anomaly Detector module monitors the unseen sensor data and determines if they come from a normal system behavior or a cyberattack. Once the anomaly is detected, this component classifies it depending on the cyberattack point and the affected resource. This module has two components to accomplish the previous tasks: *Threshold Monitor* and *Anomaly Classifier*.

The Threshold Monitor component receives the model generated in the previous module and uses it to test unseen samples. In particular, this module computes the error between the predicted values of the model and the new unseen values as shown by Eq. (9). Once the error is computed, it is normalized using the mean and standard deviation computed from the error prediction from the training dataset in the Model Generation module, as can be seen in Eq. (10). If the normalized error exceeds the threshold defined by the Model Trainer Engine and Validation, an anomaly is detected and passed on to the next component. One of the limitations of this module is the number of false positives that it can report. This phenomenon is common in any semi-supervised AD solution. Instead of raising the alarm each time an anomaly is detected, we suggest to overcome this limitation by raising the alarm when the number of consecutive anomalies detected by this module is higher than a specific number determined by the experts.

$$e_{unseen} = |y_{unseen} - \hat{y}_{unseen}| \tag{9}$$

$$e_n = \frac{e_{unseen} - \mu_{et}}{\sigma_{et}} \tag{10}$$

The Anomaly Classifier component classifies the anomaly depending on the sensor/actuator where it impacts. Besides, this component also classifies the anomaly, depending on the resource that is affected by the cyberattack. This type of classification helps operators and the administrator prioritize certain anomalies over others. For example, in a water treatment scenario, anomalies affecting the excess or defect of chemicals product must be prioritized over other types of sustainability anomalies (e.g., energy consumption) because it can cause poor quality water that ultimately can be distributed in the market and, therefore, affect the customers' health.

4.5. Anomaly detection ensembler

The Anomaly Detection Ensembler component combines the outputs of different anomaly detectors to compute a global value. This value determines with better precision a particular anomaly, or several of them, affecting different aspects of the industrial plant. In this work, we focus on sustainability; however, this module can consider different anomaly detectors. We highlight three of them: Production, Intrusion, and Fault. The production anomaly detector is focused on detecting cyberattacks that decrease the quality of the goods produced. The intrusion anomaly detection is centered on detecting general cyberattacks that impact the system behavior. Finally, the fault anomaly detector is focused on detecting cyberattacks that break specific devices in the industrial system. The only requirement of the Anomaly Detection Ensembler is that the output of the anomaly detectors must a fixed pattern.

In general, the Anomaly Detection Ensembler must receive an input vector AD of the form of $[ts, result, source]$ from each anomaly detector, where ts is the timestamp when the anomaly is detected, $result$ is the computed output of a specific anomaly detector, and $source$ is the system component where the anomaly is produced. Then, the Anomaly Detection Ensembler can gather all available vectors ($[AD_1, AD_2, \ldots, AD_{na}]$), where AD_i is the value generated by the ith anomaly detector deployed in the scenario and outputs a vector ($[ts, anomaly, source]$) where ts indicates when the anomaly was produced, $anomaly$ indicates the type of anomaly (e.g., sustainability) and $source$ is the element (e.g., a sensor or actuator) where the anomaly was produced. In addition, the Anomaly Detection Ensembler can be configured to weigh each input vector in a particular way. The administrator defines the weights, and he/she can give more or less importance to some specific detectors.

In addition, the Anomaly Detection Ensembler can be configured to weigh each input vector in a particular way. The administrator defines the weights, and he or she can give greater or lesser emphasis to some specific detectors. Such importances will be defined depending on the impact of each anomaly in the industrial scenario. For example, an anomaly produced in water treatment plant that increases or decreases the chemical consumption can negatively affect the health of humans and can even poison them.

Besides, since the output of the Anomaly Detection Ensembler contains the source sensor or actuator where the anomaly is produced, the administrators unequivocally identify the affected point and they can focus their efforts on that point to mitigate the impact. This property leads to cost reduction and the design of more effective and customized mitigation strategies. These mitigation strategies will depend on the specific anomaly detected, and they need to be configured previously by the administrators. These strategies can be automatic, or semi-automatic carried out by an operator. The automatic mitigation strategies comprise all of the cases where the system can handle all the situations without human intervention. However, the semi-automatic mitigation strategy is adopted when a particular decision needs to be taken. In automatic and semi-automatic strategies, the anomaly detected is shown in the Human–Machine Interface, together with the mitigation strategies configured for this kind of anomaly. Furthermore, in semi-automatic strategies, all the required steps are shown in the Human–Machine Interface.

5. Sustainability-based features for industrial environments

As previously mentioned, SUSAN uses basic features extracted from sensors and actuators related to sustainability. Moreover, the framework generates higher-order features from the basic ones. On the one hand, sensors return a resource consumption measure, which can take continuous (i.e., watts consumed or pH level) or discrete values (i.e., the lack of particular resources). On the other hand, actuators indicate the status of the device that controls resource consumption. In this case, the actuator mainly returns a discrete value, for example,

if a pump is open or closed. However, they can also return a continuous value in some situations, such as the opening percentage of a valve.

Firstly, specific features will be selected from sensors and actuators that directly impact industrial sustainability, and they will depend on the scenario where the industrial system is deployed. For example, in a water treatment plants, the features related to resource consumption measure or dose the consumption of products (i.e., energy and chemicals) used to treat water. Secondly, given the repetitive nature of industrial systems, two features are computed to encode the time. To provide information about the time, the timestamp feature is encoded in the form of a unit circle extracting two new features representing the sine and cosine. This time representation eases the learning of repetitive patterns. Next, SUSAN considers three techniques to extract a set of statistical and time-series features. All the techniques presented in this section are applied over a time window, w^i, defined as all the samples between x_{i-l+1} and x_i where l is the length of the window. The value of l depends on the specific scenario, and different values of l must be tested to select the optimal one. As in the Model Generation module, l must be chosen carefully due to its impact on time detection. The larger the l selected, the longer it is required to detect an anomaly.

The first technique consists in extracting features from the resource consumption habits. These new features allow us to discriminate between normal and abnormal behavior in resource consumption. To summarize these habits, we use several statistical functions over a time window. The formal definition can be seen in Eq. (11). These new features will be computed from the features without OHE applied, and they will be added to a new database called DB' together with the OHE features. Each set $Habits(i)$ adds ten new features to each sample vector x_i.

$$Habits(i) = \{f(w^i) \mid w^i \in W, f \in F\} \tag{11}$$

The second technique is the autocorrelation that is applied over DB'. The autocorrelation function [34] is useful to find repetitive patterns in time-series data. Autocorrelation is defined as the correlation of a signal with a delayed copy of itself as a function of the delay k (lag). Eq. (12) shows the formal definition of the autocorrelation function applied to a window . Once autocorrelation is applied to each window, we propose to compute statistical measures over the autocorrelation values for every k to summarize the result. In particular, from each feature in the window, the new features computed will be: mean, standard deviation, minimum, maximum, sum, and range.

$$autocorr_{w,k} = \frac{\sum_{j=k+1}^{l}(w_j - \mu_w)(w_{j-k} - \mu_w)}{\sum_{j=1}^{l}(w_j - \mu_w)^2} \tag{12}$$

Finally, the last technique is DFT [35] and is also applied over DB'. DFT is useful to convert each signal from the time domain to the frequency domain allowing us to incorporate frequency information into our model. DFT is a mathematical tool that decomposes a discrete signal into its frequencies. Eq. (13) shows the formal definition of DFT. Like autocorrelation, once DFT is computed for every feature in the window, we summarize the result using statistical functions over the entire DFT vector: mean, standard deviation, minimum, maximum, sum, and range.

$$\overline{DFT}_k = \sum_{j=0}^{l-1} w_j e^{-\frac{2\pi i}{l} kj} \tag{13}$$

We propose these two techniques to capture the repetitive patterns created by the repetitive nature of ICS. Table 3 summarizes the number of new features engineered from one basic feature of the original dataset.

Table 3
New engineered features from each single basic feature.

Technique	New features
Resource consumption habits	10
Autocorrelation	6
DFT	6

6. Experiments

In contrast to Section 4 where we detailed the high-level design of our framework, this section describes how SUSAN modules work in the SWaT scenario. In particular, we focus on the Data Cleaning, Sustainability-based Features Generation, and Model Generation modules. After explaining how these modules process the SWaT dataset, we detail the performance obtained by SUSAN and compare the results with the approaches presented in Section 2.

6.1. Dataset

The SWaT database contains data from 11 days of operation. Seven days were under normal operation of the industrial plant, while four days were under attack conditions. The data stored in the database contains a snap snapshot of 26 sensors and 25 actuators of the industrial plant every second. The SWaT database is provided as two Comma-Separated Values (CSV) files. The first file contains the data for the normal behavior of the first seven days, while the second file contains normal and abnormal behavior corresponding with the four days under attacks. The attacks presented in this file can be classified as follow:

- Single-stage single-point: cyberattacks that target only one sensor or actuator in exactly one process.
- Single-stage multi-point: cyberattacks that target multiple sensors and/or actuators in one process.
- Multi-stage single-point: cyberattacks that target one sensor or actuator in more than one process.
- Multi-stage multi-point: cyberattacks that target multiple sensors and/or actuators in two or more processes.

6.2. Data preprocessing

The Data Cleaning component cleaned the SWaT database and selected proper features to be used in the subsequent SUSAN modules. The features selected and their descriptions are listed in Table 4.

We are interested only in those processes containing sensors or actuators related to resource consumption; therefore, we discarded P1, P3, and P6 and focused, instead, on P2, P4, and P5. Specifically, the P2 features selected were those related to the sensors and actuators in charge of maintaining the water quality. The quality is guaranteed by measuring and dosing the following chemical resources, listed with the features associated with each one: HCL (AIT201, P201, and P202), NaOCl (AIT202, P203, and P204), and NaCl (AIT203, P205, and P206). The features named AITXXX are related to sensors, whereas the PXXX features refer to actuators that open/close pumps to adjust the water quality. Regarding P4, the features selected were related to the quality of the Reverse Osmosis (RO) process. We chose the following features from different sensors and actuators: AIT402 (measures HCl and NaOCl level), AIT401 (water hardness), P403, and P404 (pumps actuators that control the sodium bi-sulfate), and UV-401 (monitors the dechlorination process). Similarly, we selected from P5 the features related to the measurement of water quality, namely, AIT501 (HCl level in RO), AIT502 (NaOCl level in RO), AIT503 (NaCl level in RO), and AIT504 (NaCl level in permeate tank).

Regarding the selection of the appropriate samples from the dataset, some of the cyberattacks included in the dataset do not compromise the system's sustainability because they do not affect the selected features,

Table 4
Features selected concerning resource consumption.

Feature	Objective	Feature	Objective
AIT201	Measure NaCl level	AIT401	Measure water hardness in RO
AIT202	Measure HCl level	AIT402	Measure HCl and NaOCl levels
AIT203	Measure NaOCl level	UV401	Remove chlorine from water
P201	NaCl dosing pump	P403	Sodium bi-sulfate pump
P202	NaCl dosing pump (Backup)	P404	Sodium bi-sulfate pump (Backup)
P203	HCl dosing pump	AIT501	Measure HCl level in RO
P204	HCl dosing pump (Backup)	AIT502	Measure NaOCl level in RO
P205	NaOCl dosing pump	AIT503	Measure NaCl in RO
P206	NaOCl dosing pump (Backup)	AIT504	Measure NaCl in permeate tank

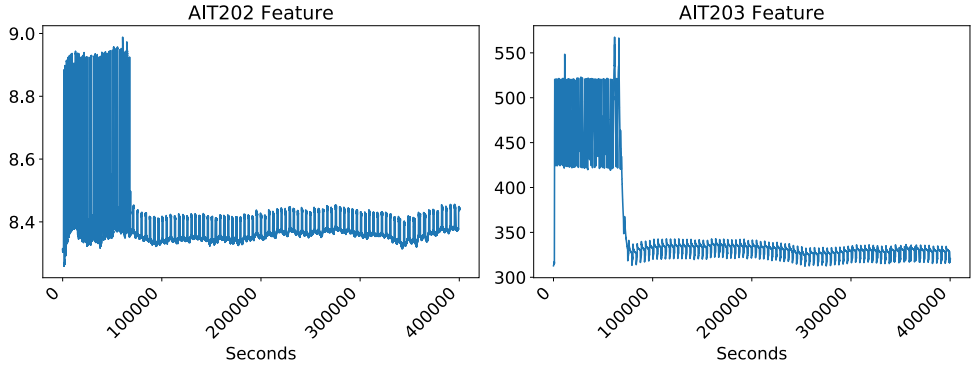

Fig. 3. The impact of the warm-up process on the values of two different features.

Table 5
Cyberattacks that impact the resource consumption.

Cyberattack #	Features affected	Effect
6	AIT202	Change in water quality
11	FIT401	UV process shutdown
19	AIT504	Set AIT504 value to 16
20	AIT504	Set AIT504 value to 255
22	UV401, AIT502, P501	Damage RO process
24	P203, P205	Change in water quality
28	P302	Stop inflow in the first tank of P4 and decrease the resource consumption
38	AIT402, AIT502	Water goes to the drain because of overdosing
40	FIT401	Water goes to RO and UV shutdown

thus producing abnormal behaviors in the system (in which we are not interested). Therefore, the samples belonging to these unwanted cyberattacks were discarded. The remaining cyberattacks were presented in Section 3 and are briefly described in Table 5.

Furthermore, the samples collected after a cyberattack were originally labeled as normal in the SWaT dataset. However, only after the cyberattack, the system is in abnormal status, and it takes some time to reach normal behavior. These incorrectly labeled samples produce spurious false positives, so we decided to remove ten minutes (600 samples) after each cyberattack. In addition, we also observed abnormal values in samples belonging to the first seconds of the dataset. This is caused by the warm-up process of the industrial system, and we observed that the values of some sensors and actuators were not representative. As illustrated in Fig. 3, it took around 80 000 s to stabilize the values for different features. For this reason, the first 100 000 s of the data were removed.

After labeling the samples as described above, the Features Encoding component encoded the categorical features P201, P202, P203, P204, P205, P206, UV401, P403, and P404 using the OHE schema, yielding a total of 14 new features. This process was carried out using the Pandas 1.1.5 library [36] Next, the Dataset Generation component utilized the last four operational days of the SWaT database as a test dataset to preserve its temporal coherence. Besides, the first six operational days after removing the warm-up process were divided as follows: the first 80% of the samples as the training dataset and the remaining 20% as the validation dataset.

Finally, the Feature Normalization component normalized the datasets using a Standard scaler by subtracting the mean for each continuous feature in the training set, and the resulting values were divided by the standard deviation. The same mean and standard deviation computed on the training set were used to standardize both the validation and the test dataset. This process was implemented using the Scikit-Learn 0.23.2 library [37]

6.3. Sustainability features generation

The Feature Filtering component studies the correlation between features using the Pandas library. Fig. 5 depicts the correlation between features, including the label. As can be seen, the most correlated features are P205 and P203, with 99.94%. This means that when SWaT corrects the HCl level in P2, the NaOCl levels will need to be corrected with a high probability. Concerning the correlation between the features and the label, we observed that UV-401 is the feature most correlated with the vector label, with a correlation of 75.10%. Therefore, we did not find any feature highly correlated with the label. In conclusion, no feature was removed due to suspicion of data leakage in the SWaT dataset. Additionally, the Feature Filtering component executed a variance study to remove those features with the lowest variance. This study was carried out by using the Scipy 1.5.4 library [38] Fig. 4 shows the result of the study and, in particular, the features P202 and P404, whose variance is equal to 0, were removed.

Furthermore, the Feature Filtering component determined if the value distribution of all features is preserved between all datasets using the K-S test available in the Scipy library. We concluded that all the continuous features selected in the Data Cleaning component came from the same statistical distribution. However, we could observe that the discrete P201 feature values did not preserve the value distribution, as shown in Fig. 6. Therefore, we removed this feature and all the features computed previously from it.

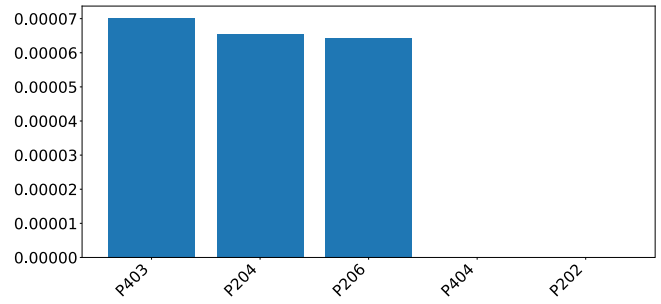

Fig. 4. Features with the lowest variance.

Table 6
Number of new features extracted depending on the technique used.

	# of features
Basic features	21
Time-related	2
Resource consumption habits	150
Autocorrelation	1026
DFT	1026
Total	2234

The Feature Extraction component utilized the techniques detailed in Section 5 to extract new higher-order features. First, the timestamp was replaced by two new features, the sine and cosine of the time seen as an angle in degrees. This time representation eases the learning of regular patterns; and second, we defined a time window length, l, of 120 s to be used in the remaining techniques. To set the value of l, different values were tested (80, 100, 120, 140), with 120 as the optimal one. Next, the Feature Extraction component used Eq. (11) to extract new features related to resource consumption habits. In particular, ten features were generated per each basic feature, resulting in a total of 150 new features. These new features were added to a new database, DB', together with the basic features. Then, the Feature Extraction component applied Eq. (13) and Eq. (12) over DB' to extract new features related to DFT and autocorrelation, respectively (6 features for DFT and 6 features for autocorrelation) for each feature in DB' except for the time-related ones, resulting in a total of 2 052 new features. Both DFT and autocorrelation were implemented by means of Numpy 1.19.5 library [39]. After the feature engineering process, the total number of features was 2 234. Table 6 summarizes the new features extracted at this stage.

Finally, a new variance study was conducted to determine if any of the values of the higher-order features did not change. The study reported a total of 610 features to be remove because they remained unchanged in the dataset. These features were the min, max, and range of certain features computed from autocorrelation and DFT. Finally, the dataset is composed of a total of 1624 features.

6.4. Model generation

To accurately model the SWaT normal behavior and its temporal pattern, an LSTM model was selected in the Anomaly Detection Model Selection component. An LSTM network allows learning the temporal pattern presented in the dataset and making predictions in the future. The input of the LSTM network was a batch of sequences. Each sequence of length l represents the evolution of the feature values during the last l timesteps. The LSTM network output is the value predicted for each feature, h timesteps after the end of the input sequence.

Next, the Model Fine-tune component was in charge of selecting the optimal hyper-parameters values, and therefore, to improve the result of the AD model. In terms of input data hyper-parameters, the window length (l) was set to 120 seconds, and it indicates the number of samples in the window. This value was selected from a list of different values ([80, 100, 120, and 140]). The horizon, h, was set to 10 to prevent the model from copying the last input value, and it indicates the timestep in the future from which the model will start to predict. Finally, the number of timesteps predicted by the model, m, was set to 1 because we are only interested in predicting the next value. In terms of hyper-parameters that depend on the model, a random-search strategy tried hyper-parameters values randomly and selected the optimal combination. The fine-tuning process was achieved by using the training dataset. We monitored the reconstruction error over the validation dataset to decide the optimal values of the hyper-parameters. Table 7 shows the hyper-parameters values tested and the values finally selected to train the DL model.

Table 7
Set of hyper-parameters tested. The optimal ones used in the DL model are shown in bold.

Hyper-parameter	Values
Window length (l)	[80, 100, **120**, 140]
LSTM layers	[2,**3**]
Neurons per LSTM layer	[**[512, 256, 128]** ,[512, 128], [256, 128]]
Dense Layers	[1,**2**]
Neurons per Dense layer	**[[100,1624]** ,[1624]]
Activation function	[**Relu**, Sigmoid]

Finally, the Model Trainer Engine and Validation performed two training processes. The first training was in charge of finding the proper threshold. In this case, the LSTM was trained using the training dataset, and the threshold was computed using the validation dataset. Once the threshold was computed, the second training was intended to train the final LSTM model using a new dataset composed of the training and validation datasets. In particular, this component was ran on a workstation with 94 GB of RAM, a six-core Intel i7-5930K at 3.5 GHz with hyper-threading running Linux, and one NVIDIA GeForce GTX 1080 with 8 GB of RAM. The software used to train the model was TensorFlow 1.12.0 [40] and the Keras library 2.2.4 [41].

6.5. Model validation

The validation process was executed by the Model Trainer Engine and Validation component using the test dataset to provide new unseen samples to the model. The validation aims to determine the precision, recall, and F1-score achieved by our implementation. These metrics are defined as a function of the values TP, TN, FP, and FN.

In particular, precision, recall and F1-score are defined as follows:

- Precision: Indicates what fraction of the detected anomalies is real anomalies.

$$Precision = \frac{TP}{TP + FP}$$

- Recall: Indicates what fraction of the real anomalies is detected.

$$Recall = \frac{TP}{TP + FN}$$

- F1-score: Is the harmonic mean between recall and precision and is a trade-off between precision and recall.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Table 8 summarizes the recall rates achieved by the different approaches focused on the SWaT dataset reviewed in Section 2. We restricted our study to those solutions that show the detection rate per cyberattack. As our implementation is focused on the sustainability cyberattacks described in Section 6.2 above, we only depict the recall achieved for those cyberattacks. In general terms, our solution achieved the most balanced results for all cyberattacks that produce sustainability issues. In particular, the best recall achieved was 1 for cyberattack 20, and the worst recall was 0.64 for cyberattack 19.

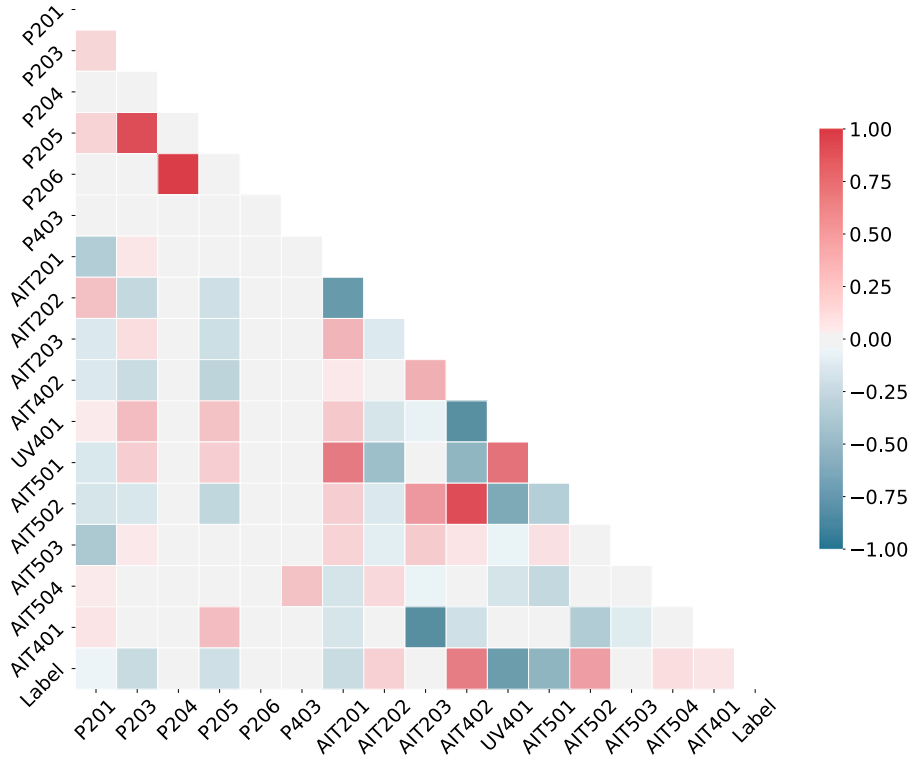

Fig. 5. Correlation between each pair of features.

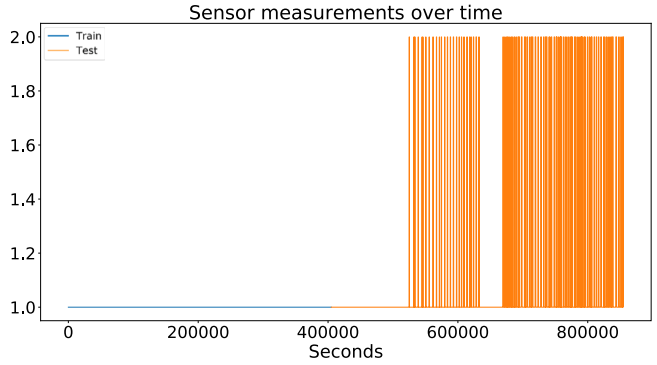

Fig. 6. Values of P201 feature measures over the time after removing the first 100 000 samples because of the warm-up process.

Table 8
Recall comparison between different approaches on SWaT dataset. DNN and OCSVM are proposed by Inoue et al. [26], RNN is proposed by Shalyga et al. [19], 1D-CNN is proposed by Kravchik and Shabtai [18], and DIF is proposed by Elnour et al. [29].

Cyberattack #	DNN	RNN	OCSVM	1D-CNN	DIF	Ours
6	0.95	0.72	0.72	0.90	**1**	0.92
11	0.99	0.98	**1**	**1**	**1**	0.96
19	**0.97**	0.12	0.13	0	0.45	0.64
20	0	0.85	0.85	**1**	0.45	**1**
22	0.98	0.99	**1**	**1**	**1**	0.97
24	**0.92**	0	0	0.17	0.34	0.78
28	0.03	0.94	0.94	**1**	**1**	0.93
38	0.77	0.92	0.93	0.86	**1**	0.69
40	0.78	0.93	0.93	**1**	**1**	0.74

However, regarding cyberattack 19, our implementation achieved the second-best results. By way of comparison, other approaches failed to detect at least one of the cyberattacks. For example, RNN [19], OCSVM [26], 1D-CNN [18], and DIF [29] detected cyberattack 24 with a recall below 0.34. However, our work detected this cyberattack with a recall of 0.78. Regarding DNN [26], it detected cyberattack 28 with a recall of 0.03. In contrast, our solution detected this cyberattack with a recall of 0.93. In addition to sustainability cyberattacks, our anomaly detector also was able to detect cyberattack 28 with a recall of 0.93. This cyberattack targeted P302, trying to close it to stop the inflow of tank T401. Although we focused on sustainability anomalies, all sensors and actuators are interlinked in an industrial system. In particular, when the P302 is closed, a decrease in the amount of the chemical resources is produced, detecting it as an anomaly.

Table 9 shows the precision, recall, and F1-score results achieved by our solution. Besides, they are compared with some of the existing solutions. It is worth mentioning that, because of the nonexistence of sustainability solutions focused on the SWaT testbed, the results of other approaches show the score achieved for all the anomalies presented in the SWaT testbed. In our case, to compute the precision, recall, and F1-score, we only considered the normal system behavior and the cyberattacks shown in Table 8. Furthermore, we only considered the results achieved by the Sustainability Anomaly Detector module. These results will be better if we use the Anomaly Detection Ensembler, which aggregates different anomaly detectors. In terms of recall, our work achieved the best result, with a score of 0.910. Regarding the precision (0.633) and F1-score (0.747), the results showed that our solution is suitable to detect cyberattacks that impact the sustainability of industrial systems.

The scores described above mean that our proposal will alert the anomalies as soon as possible to reduce the impact, as we established in Section 5. However, we observed that the false positive rate (FPR) was particularly low if we considered the fact that anomaly detection occurs in bursts due to the time dependence present in industrial systems. To calculate the FPR, we extracted all normal traffic from the test dataset and found that false positives also happened in bursts. Each of these bursts can be considered as a single alarm since the behavior of the system from one instant to the next does not vary significantly.

Table 9
Result comparison of different solutions using SWaT Dataset.

Solution	Precision	Recall	F1-score
1D CNN [18]	0.968	0.791	0.871
MLP [19]	0.967	0.696	0.812
CNN [19]	0.952	0.702	0.808
RNN [19]	0.936	0.692	0.796
LSTM [20]	–	–	0.817
DNN [26]	**0.982**	0.678	0.802
OCSVM [26]	0.925	0.699	0.796
AE Frequency [28]	0.924	0.827	0.873
DIF [29]	0.935	0.835	**0.882**
Ours	0.633	**0.910**	0.747

Therefore, if we consider the false-positive bursts as a single alarm, SUSAN detected 144 false-positive bursts during the 103 h of normal traffic in the test dataset. This resulted in an FPR of 0.000388, which makes SUSAN suitable to be deployed in real scenarios.

7. Discussion

The SUSAN framework is aimed at building detectors of anomalies caused by cyberattacks targeting the sustainability of industrial processes. To deal with such anomalies, SUSAN is composed of different modules and components to help preprocess the data and generate the proper model used later to make predictions about new unseen data. We evaluated the framework utility using a dataset collected from the SWaT testbed, a fully operational scaled-down water treatment plant.

Section 6 showed how the different modules and components of SUSAN were implemented to work together to detect anomalies related to sustainability. In general, our implementation achieved the most balanced results, detecting all the cyberattacks targeting sustainability with a high recall rate. The solutions in the literature achieved better recall than our solution, but they always failed to detect almost one cyberattack. For example, RNN [19], OCSVM [26], 1D-CNN [18] and DIF [29] failed to detect cyberattack 24. While our solution achieved a recall of 0.78. Regarding DNN [26], it failed to detect cyberattack 28 while our solution achieved a recall of 0.93. The results obtained allow our SUSAN implementation to be used in real industrial environments to detect cyberattacks that target the sustainability of the industrial systems.

In Section 2, we reviewed the most relevant contributions focused on anomaly detection, which has mainly focused on general behavior anomalies caused by cyberattacks. However, SUSAN was designed with a specific goal: to build sustainability anomaly detectors with the highest recall. The goal of the SUSAN implementation is to detect as many cyberattacks as possible despite sacrificing a low degree of precision.

One of the most exciting characteristics of SUSAN is its generic, modular, and extensible design. In this work, we integrated our SUSAN implementation into a water treatment testbed. However, SUSAN is generic enough to be implemented in any industrial system. Furthermore, SUSAN can be expanded with new modules without affecting the existing ones. This is especially useful when working with databases requiring extra preprocessing steps (i.e., a new module that performs conversion between different units previous to the feature extraction). Besides, our solution can be used in the Anomaly Detection Ensembler in conjunction with other types of anomaly detectors to increase detection rates.

Another important characteristic of SUSAN is that, due to the relationship between all the sensors and actuators in an industrial system, it can detect anomalies that slightly affect the sustainability of the industrial system. For example, when the water treated in a water treatment plant decreases by an anomaly, not only is the industrial production process strongly impacted, but also the number of resources (chemical and electricity) decreases. This relationship allows SUSAN implementations to detect anomalies that greatly impact other aspects such as production. For example, cyberattack 28 was intended to cause an underflow in Tank 101 and an overflow in Tank 301. However, our implementation was able to detect this cyberattack with a recall of 0.93 due to the resource consumption reduction. However, this connection is not frequent, and certain anomalies generated in production may not impact the sustainability of the system. To overcome this issue, the Anomaly Detection Ensembler can aggregate additional anomaly detectors to perform a more precise detection.

8. Conclusions and future work

This paper presents SUSAN, an extensible framework to build sustainability anomaly detectors in industrial processes. SUSAN is composed of three different modules that work together to detect cyberattacks affecting sustainability. These modules are Data Preprocessing, Sustainability-based Features Generation, Model Generation, and Sustainability Anomaly Detector. Furthermore, the SUSAN framework proposes a set of techniques to extract new features related to resource-consumption habits to help the model discriminate between normal and abnormal system behaviors. In addition, some experiments were carried out to demonstrate the feasibility of the SUSAN framework. In particular, we showed the implementation of the different modules and components to detect sustainability anomalies caused by cyberattacks in a water treatment testbed. Our implementation achieved the most balanced results and detected all the considered cyberattacks causing sustainability anomalies. Regarding the recall rate, our implementation achieved a score of 1 as the best recall for the cyberattack 20 and a score of 0.64 as the worst recall for the cyberattack 19. Taking into account the general results, our implementation achieved the best recall with a score of 0.910, an acceptable precision rate of 0.633, and an F1-score of 0.747. Finally, we figured out that false positives were grouped on bursts and computed the FPR to check the feasibility of SUSAN in real scenarios. In particular, SUSAN obtained an FPR of 0.000388 which makes our solution feasible.

In future work, we plan to dedicate efforts in order to improve the precision rate. Although the precision rate achieved by our implementation is acceptable, a higher precision would help to reduce the false positives. In addition, to overcome the lack of industrial datasets focused on sustainability cyberattacks, we plan to generate such datasets. Finally, another line of work is to extend the variety of anomaly detectors that can be aggregated by the Anomaly Detection Ensembler, e.g., the Production Anomaly Detector.

Declaration of competing interest

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

Data availability

No data was used for the research described in the article.

Funding

This work has been funded under Grant TED2021-129300B-I00, by MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, UE, Grant PID2021-122466OB-I00 and Grant RTI2018-095855-B-I00, by MCIN/AEI/10.13039/501100011033/FEDER, UE, and the Swiss Federal Office for Defense Procurement (armasuisse) with the CyberSpec (CYD-C-2020003).

References

[1] Ercan Oztemel, Samet Gursev, Literature review of industry 4.0 and related technologies, J. Intell. Manuf. 31 (1) (2020) 127–182.
[2] T. Stock, G. Seliger, Opportunities of sustainable manufacturing in industry 4.0, Procedia CIRP 40 (2016) 536–541.

[3] T.H. Szymanski, Security and privacy for a green internet of things, IT Prof. 19 (5) (2017) 34–41.
[4] Amin Hassanzadeh, Amin Rasekh, Stefano Galelli, Mohsen Aghashahi, Riccardo Taormina, Avi Ostfeld, M. Katherine Banks, A review of cybersecurity incidents in the water sector, J. Environ. Eng. 146 (5) (2020) 03120003.
[5] Tomas Plėta, Manuela Tvaronavičienė, Silvia Della Casa, Konstantin Agafonov, Cyber-attacks to critical energy infrastructure and management issues: Overview of selected cases, 2020.
[6] Lorenzo Fernández Maimó, Ángel Luis Perales Gómez, Félix J. García Clemente, Manuel Gil Pérez, Gregorio Martínez Pérez, A self-adaptive deep learning-based system for anomaly detection in 5G networks, IEEE Access 6 (2018) 7700–7712.
[7] Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Manuel Gil Pérez, Félix J. García Clemente, Gregorio Martínez Pérez, Dynamic management of a dep learning-based anomaly detection system for 5G networks, J. Ambient Intell. Humaniz. Comput. 10 (8) (2019) 3083–3097.
[8] Abiodun Ayodeji, Yong-kuo Liu, Nan Chao, Li-qun Yang, A new perspective towards the development of robust data-driven intrusion detection for industrial control systems, Nucl. Eng. Technol. 52 (12) (2020) 2687–2698.
[9] Ángel Luis Perales Gómez, Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Félix J. García Clemente, Frances Cleary, Crafting adversarial samples for anomaly detectors in industrial control systems, in: The 4th International Conference on Emerging Data and Industry 4.0 (EDI40), 2021, (In Press).
[10] Ángel Luis Perales Gómez, Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Félix J. García Clemente, MADICS: A methodology for anomaly detection in industrial control systems, Symmetry (Basel) 12 (10) (2020) 1583.
[11] Ángel Luis Perales Gómez, Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Félix J. García Clemente, Manuel Gil Pérez, Gregorio Martínez Pérez, SafeMan: A unified framework to manage cybersecurity and safety in manufacturing industry, Softw. Pract. Exp. 51 (3) (2020) 607–627.
[12] Nathan Pelletier, Maurice Doyon, Bruce Muirhead, Tina Widowski, Jodey Nurse-Gupta, Michelle Hunniford, Sustainability in the Canadian egg industry—Learning from the past, navigating the present, planning for the future, Sustainability 10 (10) (2018) 3524.
[13] Surajit Bag, Arnesh Telukdarie, J.H.C. Pretorius, Shivam Gupta, Industry 4.0 and supply chain sustainability: Framework and future research directions, Benchmarking: Int. J. 28 (5) (2018) 1410–1450.
[14] Clayton Miller, Zoltán Nagy, Arno Schlueter, A review of unsupervised statistical learning and visual analytics techniques applied to performance analysis of non-residential buildings, Renew. Sustain. Energy 81 (2018) 1365–1377.
[15] D.B. Araya, K. Grolinger, H.F. ElYamany, M.A.M. Capretz, G. Bitsuamlak, Collective contextual anomaly detection framework for smart buildings, in: 2016 International Joint Conference on Neural Networks, IJCNN, 2016, pp. 511–518.
[16] Cheng Fan, Fu Xiao, Yang Zhao, Jiayuan Wang, Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data, Appl. Energy 211 (2018) 1123–1135.
[17] N.L. Tasfi, W.A. Higashino, K. Grolinger, M.A.M. Capretz, Deep neural networks with confidence sampling for electrical anomaly detection, in: 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017, pp. 1038–1045.
[18] Moshe Kravchik, Asaf Shabtai, Detecting cyber attacks in industrial control systems using convolutional neural networks, in: Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy, 2018, pp. 72–83.
[19] Dmitry Shalyga, Pavel Filonov, Andrey Lavrentyev, Anomaly detection for water treatment system based on neural network with automatic architecture optimization, 2018, arXiv preprint arXiv:1807.07282.
[20] Giulio Zizzo, Chris Hankin, Sergio Maffeis, Kevin Jones, Intrusion detection for industrial control systems: Evaluation analysis and adversarial attacks, 2019, arXiv preprint arXiv:1911.04278.
[21] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, See-Kiong Ng, MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks, in: Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series, 2019, pp. 703–716.
[22] Jonguk Kim, Jeong-Han Yun, Hyoung Chun Kim, Anomaly detection for industrial control systems using sequence-to-sequence neural networks, 2019, arXiv preprint arXiv:1911.04831.
[23] Panagiotis Radoglou Grammatikis, Panagiotis Sarigiannidis, Antonios Sarigiannidis, Dimitrios Margounakis, Apostolos Tsiakalos, Georgios Efstathopoulos, An anomaly detection mechanism for IEC 60870-5-104, in: 2020 9th International Conference on Modern Circuits and Systems Technologies, MOCAST, IEEE, 2020, pp. 1–4.
[24] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, Joarder Kamruzzaman, Ammar Alazab, A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks, Electronics 8 (11) (2019) 1210.
[25] Marco Caselli, Emmanuele Zambon, Frank Kargl, Sequence-aware intrusion detection in industrial control systems, in: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, 2015, pp. 13–24.
[26] J. Inoue, Y. Yamagata, Y. Chen, C.M. Poskitt, J. Sun, Anomaly detection for a water treatment system using unsupervised machine learning, in: 2017 IEEE International Conference on Data Mining Workshops, ICDMW, 2017, pp. 1058–1065.
[27] Ángel Luis Perales Gómez, Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Félix J. García Clemente, Cristian Cadenas Sarmiento, Carlos Javier Del Canto Masa, Rubén Méndez Nistal, On the generation of anomaly detection datasets in industrial control systems, IEEE Access 7 (2019) 177460–177473.
[28] Moshe Kravchik, Asaf Shabtai, Efficient cyber attacks detection in industrial control systems using lightweight neural networks, 2019, arXiv preprint arXiv:1907.01216.
[29] M. Elnour, N. Meskin, K. Khan, R. Jain, A dual-isolation-forests-based attack detection framework for industrial control systems, IEEE Access 8 (2020) 36639–36651.
[30] Hamid Reza Ghaeini, Nils Ole Tippenhauer, Hamids: Hierarchical monitoring intrusion detection system for industrial control systems, in: Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy, 2016, pp. 103–111.
[31] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, Aditya Mathur, A dataset to support research in the design of secure water treatment systems, in: International Conference on Critical Information Infrastructures Security, Springer, 2016, pp. 88–99.
[32] Jacob Benesty, Jingdong Chen, Yiteng Huang, Israel Cohen, Pearson correlation coefficient, in: Noise Reduction in Speech Processing, Springer, 2009, pp. 1–4.
[33] Vance W. Berger, YanYan Zhou, Kolmogorov–smirnov test: Overview, in: Wiley Statsref: Statistics Reference Online, 2014.
[34] John A. Gubner, Probability and Random Processes for Electrical and Computer Engineers, Cambridge University Press, 2006.
[35] Kamisetty Ramamohan Rao, Do Nyeon Kim, Jae Jeong Hwang, Fast Fourier Transform: Algorithms and Applications, Vol. 32, Springer, 2010.
[36] The pandas development team, Pandas-dev/pandas: Pandas, 2020, http://dx.doi.org/10.5281/zenodo.3509134.
[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al., Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[38] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al., SciPy 1.0: Fundamental algorithms for scientific computing in Python, Nature Methods 17 (2020) 261–272.
[39] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al., Array programming with NumPy, Nature 585 (7825) (2020) 357–362, http://dx.doi.org/10.1038/s41586-020-2649-2.
[40] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016, arXiv preprint arXiv:1603.04467.
[41] François Chollet, et al., Keras, 2015, https://keras.io.