Internet of Things 29 (2025) 101446

Contents lists available at ScienceDirect

Internet of Things

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

A framework for anomaly classification in Industrial Internet of Things systems

Martha Rodríguez, Diana P. Tobón, Danny Múnera *

Universidad de Antioquia, calle 67 No. 53 – 108, Medellin, Colombia

ARTICLE INFO

Keywords:
Anomaly classification
Industrial Internet of Things
IIoT
Anomaly detection

ABSTRACT

Introducing the Industrial Internet of Things (IIoT) into traditional industrial processes has marked a new era of enhanced connectivity and productivity. By integrating advanced sensors, communication technologies, and data analysis, IIoT enables real-time monitoring, proactive maintenance, and increased operational efficiency. However, this increased complexity and interconnectivity also introduce new challenges in maintaining system dependability and safety. Considering these issues, this work presents an IIoT Anomaly Classification Framework designed to detect and categorize anomalies such as failures and attacks. The research addresses the critical need for robust anomaly detection and classification in IIoT systems by providing a comprehensive and scalable solution adaptable to various industrial contexts. The framework comprises two main components: an anomaly detection model and an anomaly classification model. The anomaly detection model operates unsupervised, continuously monitoring system data to identify deviations from normal behavior patterns. At the same time, the anomaly classification model categorizes these anomalies based on historical data using machine learning algorithms. The proposed framework has been tested in a realistic IIoT environment, demonstrating its effectiveness and practicality. During the cross-validation process, a precision of 0.95, recall of 0.88, and F1-score equal to 0.91 were obtained. This research contributes significantly to IIoT, offering a valuable tool for improving industrial operations and laying the groundwork for future anomaly classification and system resilience advancements.

1. Introduction

The incorporation of the Industrial Internet of Things (IIoT) into conventional industrial operations signifies the dawn of a new age of connectivity, productivity, and intelligence. [1–3]. By embedding advanced sensors, communication technologies, and data analytics into industrial operations, IIoT has enabled real-time monitoring, predictive maintenance, and enhanced operational efficiency [4–6]. However, these systems' increased complexity and interconnectivity have also introduced new challenges, particularly in maintaining system reliability and security [6]. Anomalies such as unexpected equipment failures and cyber-attacks can disrupt operations, leading to significant financial losses and safety risks [6–9].

Anomaly detection in Industrial Internet of Things is a critical research issue due to the vulnerability of IIoT networks to novel and complex cyber threats [1]. Unlike traditional Information Technology IT environments, the IIoT integrates cyber elements into core operational processes, making the consequences of security breaches more severe [10]. The dynamic and heterogeneous nature of IIoT environments, characterized by a vast array of devices and communication protocols, complicates the detection of

* Corresponding author.
E-mail addresses: mlucia.rodriguez@udea.edu.co (M. Rodríguez), diana.tobon@udea.edu.co (D.P. Tobón), danny.munera@udea.edu.co (D. Múnera).

https://doi.org/10.1016/j.iot.2024.101446
Received 24 September 2024; Received in revised form 5 November 2024; Accepted 22 November 2024
Available online 30 November 2024

anomalies [11]. This needs continuous research into sophisticated, adaptable anomaly detection methods that can effectively discern between normal fluctuations in data and genuine security threats [12].

The significance of advancing anomaly detection in IIoT is underscored by recent research efforts that employ diverse methodologies ranging from statistical approaches to machine learning and deep learning techniques. Key works in this area include the exploration of unsupervised techniques by Zhang et al. for multivariate time series analysis in IIoT settings [10] and the approach of Ding et al. in adapting machine learning models to handle concept drift in dynamic environments [13]. Additionally, Truong et al. introduced a lightweight architecture suitable for edge computing devices, optimizing resource usage while maintaining high detection accuracy [12]. These studies highlight the ongoing need for research that enhances detection capabilities and ensures that anomaly detection systems are scalable, efficient, and capable of operating under the constraints typical of IIoT environments.

Additionally, anomalies must be classified to mitigate their effects on the IIoT. Since Industrial Internet of Things systems support industrial operations, they must promptly address malfunctions or cybersecurity threats [14]. This requires the collaboration of various teams, each with specific roles; for instance, maintenance staff handles physical failures, while the IT department manages cyber attacks. However, the detection and classification system must operate automatically with minimal human intervention to address these anomalies quickly.

Considering the above, we propose an Industrial Internet of Things Anomaly Classification Framework to detect and categorize anomalies such as failures or attacks. This research aims to contribute significantly to the field by addressing the critical need for robust anomaly detection and classification in IIoT systems. It provides a comprehensive and scalable solution that can be adapted to various industrial contexts. The proposed framework offers a valuable tool for improving modern industrial operations' reliability, security, and efficiency, paving the way for more resilient and intelligent IIoT systems. This work makes several significant contributions to the field of IIoT:

- Incorporation of Contextual Information: Demonstrating the value of integrating contextual data into anomaly detection and classification, leading to more accurate and reliable results and allowing the classifier to differentiate between average CPS or IIoT events or situations caused by failures or attacks.
- Development of an IIoT anomaly Classification Framework: Introducing a new framework that combines machine learning techniques with traditional statistical methods to improve classification speed and accuracy. The goal was to create a framework that can be easily scaled and adjusted to meet the diverse requirements of various sectors.
- Real-World Testing and Validation: Implementing the proposed frameworks in emulated settings to validate their effectiveness and adaptability and conducting extensive testing and validation of the framework using a realistic testbed environment, ensuring its applicability in real-world industrial settings.
- Discuss Implications for Future Research and Practice. Anomaly classification is an ongoing process that requires continuous monitoring and model updates. Based on the implementation results, exhaustive analysis, and knowledge provided, some lines of future work are described in the field of anomaly classification in IIoT systems.

The remainder of this paper is organized as follows. Section 2 reviews related works on IIoT anomaly classification. Section 3 describes the methodology for developing the IIoT anomaly classification framework and introduces its implementation. It includes a detailed description of its architecture, development environment, and software tools. Section 4 presents the framework's implementation and validation results with performance metrics, such as precision, recall, and F1-score. Finally, Section 6 concludes this work and outlines potential future research directions stemming from it.

2. Related works

The Industrial Internet of Things (IIoT) represents a significant leap in industrial automation and digital transformation. By interconnecting devices and systems through advanced networks and integrating Cyber–Physical Systems (CPS), IIoT enhances industrial operations' efficiency, productivity, and adaptability [6–9]. However, this interconnection also introduces vulnerabilities, making IIoT systems susceptible to anomalies. Effective anomaly classification is critical for maintaining the integrity, safety, and reliability of IIoT systems [6].

Anomaly classification in the Industrial Internet of Things (IIoT) remains an open research issue due to the complexities inherent in distinguishing between anomalies such as cyber-attacks and physical failures [14]. The challenge is amplified by the diversity of IIoT environments where distinct anomalies may manifest similarly, complicating accurate classification [15,16]. For instance, Yang et al. [15] developed a data-driven approach to classify physical faults and cyber-attacks in manufacturing motor drives. Despite its effectiveness, this method relies heavily on specific motor data patterns and may not generalize across IIoT systems, highlighting the need for more adaptable and comprehensive solutions. Similarly, the solution described in Ganjkhani et al. [17] excels in identifying specific electrical faults and attack types in power systems but does not address the broad spectrum of anomalies present in IIoT domains.

Moreover, the dynamic and evolving nature of industrial environments where new devices and technologies are constantly integrated poses another difficulty for anomaly detection systems. Anomalies can vary significantly—from mechanical failures and operational inefficiencies to sophisticated cyber-attacks—requiring diverse strategies to detect and classify them accurately. Research by Younan et al. [18] and Truong et al. [12] underscores the complexity of these anomalies but also points out that many systems are not equipped to evolve with these changes, often requiring manual recalibration or redesign to handle new types of anomalies or to adapt to new operational contexts.

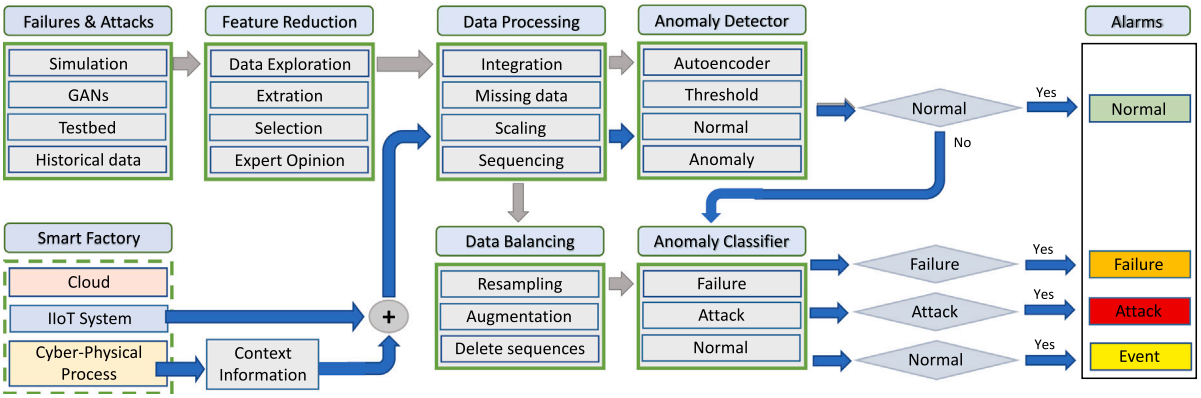

Fig. 1. Framework for anomaly classification in IIoT. During training (gray arrows), data is collected from a testbed or dataset or is artificially generated. Features are then selected, preprocessed, and used to train the anomaly detector, and classes are balanced before training the classifier. In deployment (blue arrows), data from the CPS and the IIoT system is processed before feeding the detector. If a sample is labeled as an anomaly, the classifier is invoked to classify it as a failure or attack. But if there is ambiguity between the detector and the classifier, an "event" alert is generated for a human operator to classify the sample.

Another significant challenge is the real-time requirement for anomaly detection in IIoT environments. Industrial processes often require immediate responses to prevent damage or downtime, meaning anomaly detection systems must be accurate and fast. Nizam et al. [9] and other researchers have worked on edge computing solutions to address latency issues. Still, the balance between speed, accuracy, and computational efficiency remains delicate, with current solutions often sacrificing one aspect for another.

Furthermore, existing solutions often focus on specific subsets of IIoT applications, which limits their applicability in a broader context. For example, the methodology by Zhang et al. [16] effectively distinguishes between sensor replay attacks and sensor bias failures in cyber–physical systems using watermarks tailored to each threat type. However, such a specialized approach may not be suitable for other IIoT applications where different underlying mechanisms might cause anomalies. Liang et al. [19] propose a scheme for multi-agent systems to minimize communication loss and enhance fault diagnosis. While this method shows promise in smart grids, its complexity and the need for extensive customization may hinder its deployment in less controlled or more heterogeneous environments. These examples underscore the persistent gap in developing effective anomaly classification systems that can dynamically adapt to the varied and evolving landscapes of IIoT, which makes continuous research and innovation necessary in this field.

3. Methods and materials

This section presents the framework for anomaly classification in IIoT. It shows its structure and implementation using a testbed. It describes data collection and processing and the deep-learning models to detect and classify anomalies in the dataset.

3.1. Framework description

The rapid evolution of the Industrial Internet of Things (IIoT) and its integration with Cyber–Physical Systems (CPS) have paved the way for the emergence of smart factories [20]. These advanced manufacturing environments leverage digital technologies to enhance operational efficiency, productivity, and flexibility [10,12,21]. However, these systems' complexity and interconnected nature introduce significant challenges in monitoring and securing industrial processes [6–8,18]. In response to these challenges, this comprehensive framework delineates a novel approach to anomaly detection and classification designed for the IIoT landscape. Integrating specific domain knowledge of the CPS with the data-driven insights provided by IIoT technologies. Fig. 1 and described below shows the framework's structure.

- Data Collection. A combination of several strategies for data collection data may be necessary. Using datasets available in public repositories is a quick and inexpensive option. Using software to emulate failures and attacks that could occur in the real IIoT offers the possibility of generating many samples; however, building a simulation with all the details of the real IIoT may be challenging. Using historical data on the process, it is possible that there are not enough samples with anomalies or that not all the types of anomalies that we want to classify are found. Techniques such as generative neural networks can be trained to generate artificial data, but training these models requires a large amount of data. Simulating anomalies directly in the IIoT system is the most direct way to collect the data set. Still, it can be dangerous to simulate failures and attacks in the IIoT while the process is in production. Implementing a testbed with devices similar to the actual system and simulating failures and attacks can be a less complex solution than programming a simulator. If extra devices are available in the plant, it can also be not expensive.

- Smart Factory. In this context, the term smart factory refers to integrating the cyber–physical system (CPS) with the Industrial Internet of Things (IIoT) system. The CPS is an automated industrial process that uses SCADA and programmable logic controllers (PLC). CPS is the core component of the business [10,12,21]. The IIoT system is installed to monitor the CPS through a cloud platform, optimizing its performance. In this framework, the first activity is to know the CPS and IIoT that already operate in the plant. In this stage, it is expected that the designers of the anomaly classification system have a good understanding of the IIoT architecture implemented in the plant. This includes its layers, features, protocols, security, raw data, processed information, and how the IIoT system interacts with the industrial process.
- Failures and attacks. Before collecting data, it is necessary to determine which attacks and failures the IIoT system needs to be protected against. This involves deciding which ones to include in the list and which ones to exclude. Some may be unlikely, while others may be too complex or expensive to address. Sometimes, alternative options, such as acquiring a policy, may be more practical.
- Context information. It is an aspect of any application, particularly in the industrial sector. In a few words, context awareness means using data from various sources to provide services that meet a user's specific requirements and expectations [22]. Here, it is crucial to identify any variables that might change based on the industrial process, which can be helpful in categorizing anomalies in IIoT.
- Data Processing. It transforms unstructured data into a structured format suitable for analysis. It ensures data quality, consistency, and error-free analysis, making it ideal for machine learning-based classification models [2]. Data processing involves the integration of multiple data sources, handling missing values, scaling, removing duplicates, encoding non-numerical features, normalizing the data, and handling outliers [2].
- Feature Reduction. Unnecessary features present in data lead to computational complexity and significantly decrease the precision and effectiveness of classification techniques [2,23]. Several techniques and criteria can be used to select variables. Understanding the problem domain and exploratory data analysis are used to understand the nature of variables. Dimensionality reduction techniques can be split into feature extraction by projecting original features into a new low-dimensional feature space [2,23], and feature selection by choosing a smaller set of features that can predict class labels more effectively. Finally, recursive feature elimination uses decision tree models or regularization-based methods.
- Data Balancing. In multiclass classification, methods to address class unbalance include resampling, classifier adaptation, ensemble strategies, and cost-sensitive techniques [24,25]. Another way to deal with a scarcity of available data is to generate synthetic data using a generative model based on the initial dataset [26–29]. Simulating anomalous conditions through reliable and high-fidelity simulation data can offer low-cost training data and address the problem of insufficient samples [26,27].
- Anomaly detection. It is a process that identifies patterns in time-series data that deviate from the system's normal behavior [9]. An approach suggested in this framework is to employ an unsupervised method for detecting anomalies, in which the model is trained only with normal data. This technique facilitates the identification of previously unknown anomalies. Autoencoders are an example of this approach.
- Anomaly Classification. This framework proposes a two-stage approach to anomaly classification. In the first stage, a detector differentiates between normal data and anomalies. The second stage is triggered if the samples exceed the threshold value to classify the anomaly. The classifier is designed to identify normal data and various failures and attacks. It provides one of three labels as output: normal, fault, or attack. However, when the classifier is used to predict, it only receives data sequences classified as anomalies by the detector. Therefore, if the classifier identifies one of these sequences as 'normal,' the application generates an 'event' alert. This alert will help the human operator manually verify the alert's origin.

3.2. Testbed

Four testbeds have been used to emulate a conveyor belt system; Fig. 2 shows one of them, where each comprises an AC motor, a speed driver, an S7-300 programmable logic controller (PLC), and a Human Machine Interface (HMI). The four testbeds are connected through an Ethernet network implementing Profinet protocol. An IIoT Industrial Internet of Things system has been implemented to monitor the CPS from a cloud application. In Fig. 3, the CPS is represented only with four motors and an HMI. IIoT comprises four Raspberries PI 3B, acting as edge devices. Several sensors communicate with an ESP32 microcontroller, which collects measurements and sends them to their edge devices through the MQTT protocol. Additionally, an edge device communicates via MODBUS with the HMI to obtain information about the motors. On each edge device, data is processed locally before being sent to a monitoring and control application deployed in the AWS cloud. This IIoT system will also collect data to detect and classify anomalies while monitoring the CPS.

3.3. Dataset

In this framework implementation, data was collected for ten days at intervals of four hours. Through the HMI, an edge device collects data from all the motors in the conveyor system. Internal data from each Raspberry PI was collected with the RPi-Monitor app [30] and saved into CSV files using a Python script. Additionally, relative humidity and environment temperature measurements were collected at two different locations in the plant. Table 1 describes the raw data set structure. In this work, we tried to differentiate between failures and attacks. The following describes the failures and attacks executed on the testbed.

- High-temperature failure (F1) was simulated by turning off the fan that cools the edge device. A 24-watt AC fan was placed 20 centimeters above the Raspberry Pi.

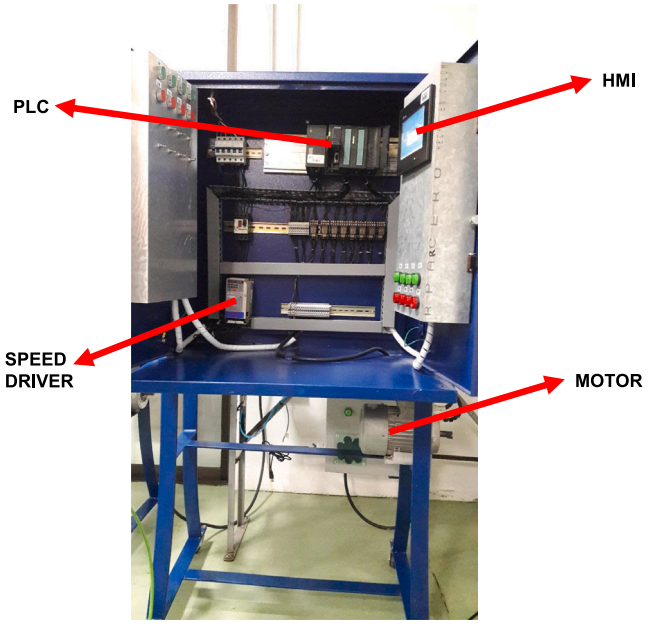

Fig. 2. Testbed used for emulating the CPS. It is composed of a three-phase motor, speed driver, S7-300 PLC, and Delta HMI.

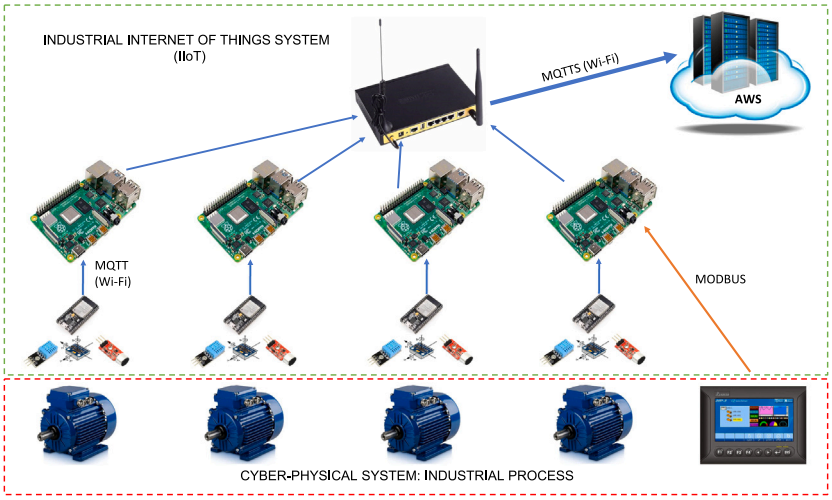

Fig. 3. An IIoT three-layer (Perception, Network, Application) model is used for anomaly classification. CPS is represented here with four three-phase motors and an HMI.

Table 1
Data Collection Details. Raw data were collected from different sources with different sampling rates and stored in separate files.

Location	File	Variables	Rate	Description
CPS	modbus.csv	frequency, voltage, current, power, recipe	0.1 s	Data from all the motors in the conveyor system
CPS	sound.csv	sound	0.001 s	Sound of motor
Edge Device	internal.csv	Temperature, frequency, CPU load for 1, 5, and 15 min, voltage, free and available memory, bytes via WiFi, current	1 s .. 10 s	Internal data from edge device collected with RPI monitor App.
Building	mqtt_temp.csv	humidity and temperature	10 s	Environment data

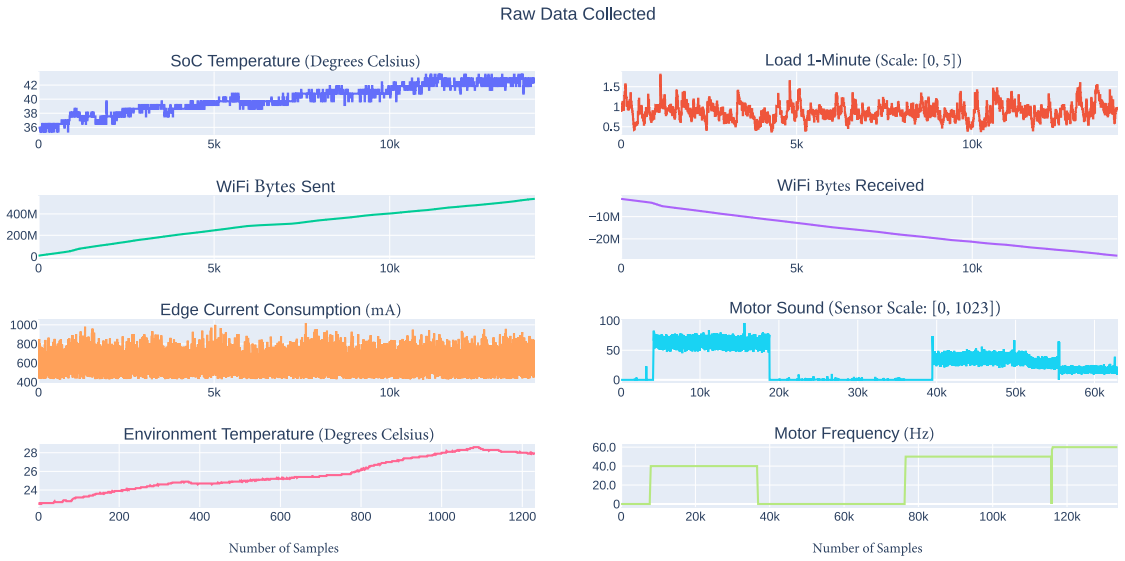

Fig. 4. Raw data collected. The plots display the appearance of some of the signals that are gathered by the IIoT system.

- Miscalibrated Sensor failure (F2). To simulate the miscalibration of the motor sound sensor, the actual measurement value was increased by 20%. This was done in the collected data, as miscalibration was not directly simulated on the sensor.
- Sensor disconnected failure (F3). A sound sensor disconnection fault was added, turning off the sensor and its respective ESP32. The dataset reflects this failure as zero values in the respective feature. This failure is not used to evaluate the model with previously unseen anomalies during cross-validation.
- A data injection attack (A1) was executed by spoofing the ESP32s connected to the motor sound sensors on each edge device. Sending the MQTT broker random data between 0 and the average value of the motor's sound signal.
- Denial of Service (DoS) attack (A2) was launched on the MQTT brokers running on all edge devices. A Python script on a laptop connected to the local network subscribed 1024 clients to the topic "#" It published short messages to the Mosquitto MQTT broker every second, thus saturating the communication channel.

3.4. Framework implementation

This section delves into the practical implementation and evaluation of the IIoT Anomaly Classifier Framework within a testbed environment. Our experimentation involved an emulated Cyber–Physical System (CPS) represented by a four-belt conveyor system and monitored by an Industrial Internet of Things (IIoT) system through a cloud platform. This setup was chosen to emulate real-world industrial conditions closely and rigorously test the framework's capability to detect and classify anomalies.

3.4.1. Data processing

Raw data often contains inconsistencies, errors, or outliers that can negatively affect the model's performance, so data processing prepares it to be used in classification models.

Data exploration. It involves understanding the data's patterns, irregularities, underlying structure, and critical characteristics through visual and quantitative methods. Fig. 4 shows some graphics obtained during visual data exploration. The figure shows, for example, that the features associated with bytes sent and received over WiFi have increasing trends, positive or negative, or that the motor's sound is strongly related to its frequency. Visual exploration of the data also allows us to determine outliers and normal ranges of the data.

Data integration. Another aspect to consider is that not all data sources were available simultaneously. Each data collection session was stored in a different CSV file, and all the variables collected were integrated into a single dataset. This implies that only timestamps where all variables are available are selected. In addition, since the sampling rate is different, resampling was performed so that all variables had five samples per second.

Missing data. When working with anomaly detection, it is vital to handle missing data effectively since it can significantly impact the accuracy and reliability of model predictions. When only a few samples were missing, we used linear interpolation to generate a new value between two existing values by taking their average. Besides, we used hot-deck imputation when delays caused the absence of many samples. This technique involves copying a range of well-conformed data of the same signal into the interval where samples are missing. Fig. 5 shows a section of the motor sound where fewer samples than the average rate were collected (above).

Fig. 5. Handling missing data. Two methods to replace missing values are linear interpolation, which creates a new value between two samples, and hot deck imputation, which copies a range of data from the same series.

This section is zoomed in, and the graphs in the center show what the signal would look like if the data were aggregated using the linear interpolation technique (red) and hot-deck imputation (green). It could be concluded a priori that for long stretches of data, hot-deck imputation better preserves the signal's shape (below).

Feature engineering. It is a process that involves using domain knowledge to create new features or modify existing ones to increase the predictive power of a machine-learning model. In this work, to eliminate the ever-increasing trend of bytes sent and received over WiFi, the delta value of the signals has been decided to be used. This means taking the difference between each value and the value that comes immediately after.

Data scaling. Datasets often contain features that have varying units and scales. These differences can result in a biased model that prioritizes larger-scale features. To avoid this issue, it is essential to normalize or standardize the data so that each feature contributes equally to the model's prediction. We used the min–max technique, as shown below. The formula for min–max scaling is:

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)}$$

Where:

- x' is the rescaled value.
- x is the original value.
- $\min(x)$ and $\max(x)$ are the minimum and maximum values in the dataset, respectively.
- a and b are the new minimum and maximum values after rescaling.

This formula adjusts the values of the data to a specific range $[a, b]$, here to $[0, 1]$, which can help compare measures that have different units or scales.

Sequence creation. The sliding window technique in time series analysis transforms a dataset into a format suitable for machine learning models, especially deep learning [11,31,32]. Transforming time series data from [n-samples, n-features] to [n-samples, timesteps, n-features] is crucial for several reasons, such as improved learning dynamics, efficient data utilization, flexible forecasting, feature utilization, model input requirements, and temporal dependency capture [9,31]. Choosing the right window size for anomaly detection is crucial. It should be large enough to capture patterns but not too large to dilute anomalies. No universal solution exists, so experimentation is necessary to balance relevance and computational efficiency [11,32]. Fig. 6 graphically shows the process of making this transformation.

Each sequence of n_times samples is pooled separately for each feature. The figure shows that the new dataset takes up much more memory space. For further illustration, please note the n_times vertical samples highlighted in the figure on the left and their horizontal location on the right. The window size ('n_times') that slides through the data is essential to creating the sequence. Table 2 shows the F1-score for two classifiers, a Transformer and an LSTM model. According to this result, a window with 300 samples is the best option for the case study and will be used in later models.

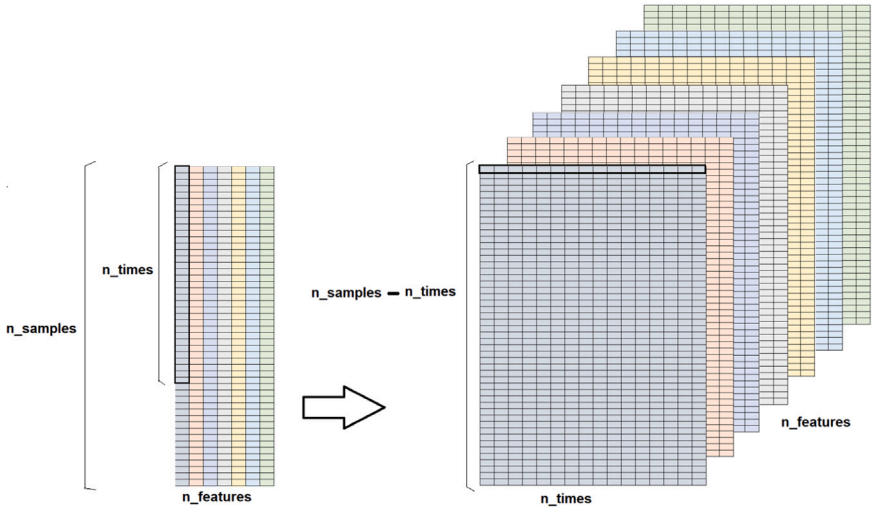

Fig. 6. Graphical representation for transforming time series data from *[n_samples, n_features]* into *[n_samples, n_times, n_features]*.

Table 2
Length sequence data. Two anomaly classification models with different sliding window sizes were evaluated based on their F1-score, and a length of 300 samples was the best for these classifiers.

Window	Transformer model	LSTM model
100	0.799250	0.754377
200	0.812366	0.773365
300	**0.833983**	**0.781802**
400	0.804396	0.684335
500	0.778157	0.617783

3.4.2. Feature selection

Specific machine learning algorithms can be computationally expensive and require a long time to train on datasets with many samples. To speed up the training process, a feature selection approach has been followed to reduce the number of input variables and simplify the data, making it easier to work with during the training phase. A Spearman correlation analysis showed that the CPU load in the last 5 min was strongly correlated with the load of 1 and 15 min. The same thing happened with the CPU's frequency and voltage. While correlation does not imply causation, it indicates that load_5 and cpu_volt could be eliminated from the dataset; this was confirmed when the anomaly detector was trained with and without these features, as shown in Table 3. Bytes sent and received over WiFi were replaced by their delta values. The temperature and relative humidity in the building where the CPS operates are not directly related to what happens in the IIoT and were considered to be discarded. Finally, during the data exploration, it was observed that the CPU's available and free memory did not show significant variations when an anomaly occurred; therefore, they were identified as candidates to be discarded. As mentioned earlier, an anomaly detection model was trained to decide whether a variable should remain in the training set with and without each variable. Table 3 shows the metrics obtained for each case. The best performance was obtained with a training set that included internal temperature, 1—and 15-minute load, the bytes sent and received delta, CPU current consumption, and motor sound. Motor frequency was also included as context information to improve model performance.

3.4.3. Data balancing

Unbalanced datasets in classification models are a crucial issue that significantly affects predictive model performance. This occurs when the distribution of classes is not uniform, causing bias towards the more frequent class. Building classification models for an unbalanced dataset requires special consideration for satisfactory performance [24,33]. Imbalanced datasets can cause machine learning models to favor the majority class, leading to inadequate performance for the minority class. This is particularly problematic for applications like anomaly detection and predicting rare events where the minority class is of greater significance [24,28,33].

In multiclass classification, some approaches combat class imbalance. Resampling adjusts the training data distribution by oversampling the minority class or undersampling the majority class; classifier adaptation modifies the classifier's decision boundary; ensemble strategies combine multiple classifiers; and cost-sensitive techniques penalize misclassification of minority classes more heavily [24,25].

Other methods include data augmentation, which is advantageous for deep learning models as they prevent overfitting [27]. Data augmentation can be significantly enhanced using Generative Adversarial Networks (GANs), particularly in domains where

Table 3
Feature Selection. The anomaly detector model was trained with all IIoT and context variables. Then, variables were eliminated one by one, and the model was trained again until the training set was reduced to 8 features.

Features	Excluded	Reason	Precision	Recall	F1 score
15	wifi_sent, wifi_receiv	It was used delta_sent & delta_receiv	0.518628	0.327521	0.401446
13	indoor_temp, %humidity	Not related to IIoT or CPS	0.912609	0.522313	0.664335
10	cpu_freq, memo_free, memo_avail	Changes not related to anomalies	0.933241	0.529629	0.675710
8	load5, cpu_volt	Strongly correlated to other variables	0.922616	0.630521	0.749053

data is limited, expensive to acquire, or requires more variety than the original dataset. GANs consist of two neural networks, the generator and the discriminator, that are trained together in a competitive process [31,34]. The training method is a zero-sum game in which the generator attempts to increase the probability of the discriminator making a mistake. In contrast, the discriminator seeks to distinguish actual data from fake data better. This game continues until the discriminator can no longer easily differentiate between actual and generated data, indicating that the generator has learned to create highly realistic data [31,34].

This framework implementation involves undersampling of the normal class. After creating the sequence with form *[n_samples, n_times, n_features]*, several samples were randomly removed from normal sequences.

3.4.4. Anomaly detection

We use an autoencoder to detect anomalies. Autoencoders are a type of machine-learning model that is commonly used for anomaly detection. These models work unsupervised and consist of two networks: an encoder and a decoder. The encoder processes the input data and converts it into a latent representation. Then, the decoder takes this representation and generates an output. By comparing the input and the output data, the differences can be used to identify anomalies in the original data [35]. In anomaly detection, an autoencoder learns to compress and decompress normal data efficiently. When an anomalous piece of data is fed into the autoencoder, the reconstruction error (the difference between the input and the reconstructed output) tends to be higher.

Here, a threshold for the reconstruction error is set. The data point is classified as an anomaly if the error exceeds this threshold. A threshold value distinguishes normal and unusual data points. Techniques balancing sensitivity and specificity are used to calculate the best threshold value. When the model was evaluated with normal data, we used the maximum difference between the input and the reconstructed output as a threshold. When calculating the threshold values, it must be considered that although datasets with normal data are used, they may present outliers not reconstructed by the autoencoder. This causes the reconstruction error to be higher, affecting the autoencoder's performance when deployed in production. To solve the above issue, the outliers from the normal data were removed using a Python script that eliminates outliers; all values above mean plus three times standard deviation are replaced by the comparison value (if $value > mean() + 3 * std()$ then $value = mean() + 3 * std()$). Moreover, it should be noted that a threshold value is calculated for each feature.

How the autoencoder works is described graphically in Fig. 7; the first plot (above) shows the delta signal of the bytes received over WiFi, and then it shows the autoencoder's reconstruction of this signal. The plot below shows the subtraction between both signals and the threshold value as a dotted line. Any value over this line is labeled as an anomaly.

In this framework implementation, one-dimensional convolutional (conv1D) alternating with dropout layers have been used for the encoder and transposed convolutional with dropout layers for the decoder. Conv1D is a type of convolutional layer designed explicitly for 1D data. It is commonly used in sequential data models, such as time series. This layer performs a convolution operation involving a sliding filter moving along the input data, effectively allowing the model to learn from local patterns within the sequence. Fig. 8 shows the schematic of the anomaly detector. When the threshold value that monitors the reconstructed signal is exceeded, the anomaly classifier is invoked. Otherwise, the sample is labeled as normal.

3.4.5. Anomaly classification

In anomaly classification, the objective is to predict a discrete label for a given input from a set of predefined classes. This is generally a supervised learning task in which a model is trained on a dataset with known labels and learns to classify new data based on this training. Initially designed for natural language processing tasks, a Transformer neural network can be used as an anomaly classifier for time series. The transformer model used here is similar to the original encoder proposed in [36]. The core component of Transformer models is the self-attention mechanism, which allows the model to weigh the importance of different parts of the input data relative to each other. For anomaly classification, the Transformer can simultaneously consider the entire sequence of inputs, determining which features are most abnormal or indicative of an anomaly. Expanding on self-attention, multi-head attention facilitates joint attention across different representation subspaces. This is particularly useful in anomaly detection as it can help the model capture multiple types of relationships or anomalies in the data. We do not use the input embedding or positional encoder mechanisms; instead, the temporal sequences described in the previous section were delivered directly to the encoder block. The rest of the layers, feed-forward, and normalization were implemented as in the original version in [36]. Fig. 9 shows the schematic of the transformer used in the anomaly classifier.

Fig. 7. Anomaly prediction with Autoencoder. Any sample above the threshold value is labeled as an anomaly.

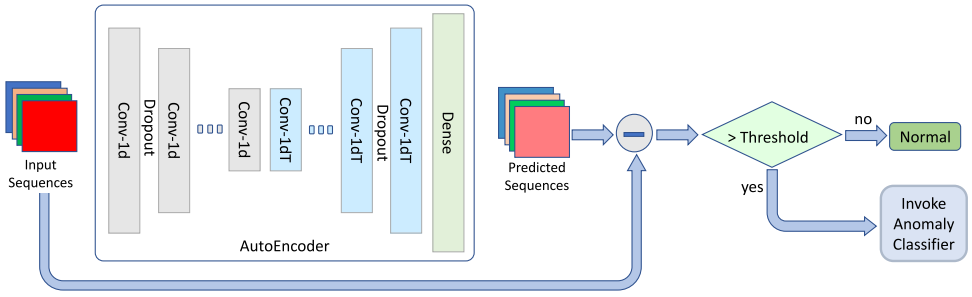

Fig. 8. Autoencoder-based anomaly detector. The 1D-convolutional model reconstructs the input signal. The anomaly classifier is invoked if both signals' differences exceed the threshold value.

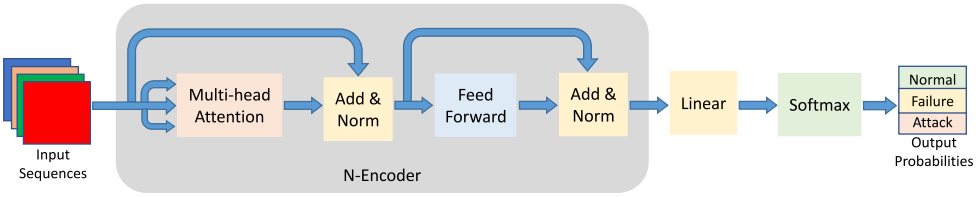

Fig. 9. Transformer-based classifier. Data sequences are delivered directly to the encoder without input embedding or positional encoding layers.

3.4.6. Anomaly detector and classifier together

Fig. 10 shows the schematic of the anomaly detector and classifier operating together. The detector implements an autoencoder model, which alerts of anomalies and invokes the classifier to determine whether it is a failure or an attack. If the classifier generates an ambiguous diagnosis indicating that the detector wrongly labeled the sample and that it is actually *normal*, an *event* label is generated so that the human operator can classify it manually.

3.4.7. Metrics

Several metrics have been proposed in the literature to evaluate the performance of machine learning models. One of the most used is accuracy, which measures the number of samples correctly classified. However, in this study case, there are unbalanced datasets with much higher amounts of normal data than data labeled as anomalies. This generates high accuracy even though the model performance is very poor. Instead, precision, recall, and F1-score metrics that do not involve the number of True Negatives (TN) (normal values) in their equations are recommended. The equations of these metrics are shown in Table 4. On one hand, Recall shows how many anomalies were not detected; many False Negatives (FN) lead to a low recall. On the other hand, precision

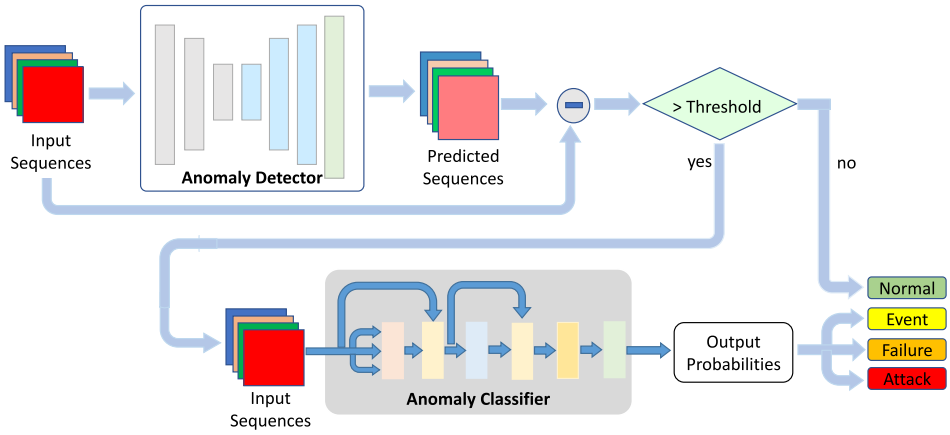

Fig. 10. Anomaly detection and classification. If the difference between the input and the reconstructed signal exceeds the threshold value, the classifier is invoked to classify the samples between failures and attacks; if it is classified as normal, it is labeled as an event so that the operator can classify it manually.

Table 4
Evaluation Metrics for Anomaly Classification. For unbalanced datasets, it is advisable to use metrics that do not involve the majority class, such as precision, recall, and F1 score.

Metric	Formula	What it Evaluates
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Overall correctness
Sensitivity (Recall)	$\frac{TP}{TP+FN}$	Correctly identified positives
Precision	$\frac{TP}{TP+FP}$	Correctness of positive predictions
F1-Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}}$	Balance between precision and recall

shows how much normal data was identified as anomalies; a high False Positive (FP) rate produces a low precision. While F1-score provides an estimate between the False Positive and False Negative rates.

4. Results

The results presented in this section underscore the efficacy and potential of the proposed IIoT anomaly classification framework in identifying and categorizing anomalies within industrial IoT systems. The comprehensive experimental setup, involving an emulated Cyber–Physical System (CPS) and a robust Industrial Internet of Things (IIoT) monitoring system, provided a realistic testbed to evaluate the framework's performance. The framework demonstrated high detection accuracy and low false positive rates across various test scenarios, highlighting its reliability in real-world applications.

4.1. Dataset

Generating enough data is key for detecting and classifying anomalies, including failures, cyberattacks, and perturbations [37]. Some authors develop datasets using simulators [35,37–39], testbeds [21,40–43], or online data [2,9–11,21,32,44–48] . Utilizing accessible online datasets can help cut down on collection expenses. We did not find a dataset that separately labeled enough failures and attacks in IIoT, so we used a testbed to build a set with data that met the established requirements. This dataset is available online in a public repository.[1] We collected data for ten days, storing information for approximately 4 h daily. The data has been stored in 48 files, of which 31 contain normal data, 13 contain failures due to high temperature (F1), miscalibration (F2), and disconnection (F3), and four files contain attacks type DoS (A2) and data injection (A1). Each file can contain one or more anomalies of the same or different types. There are 2105134 normal records, 60463 records labeled as failures, and 38755 as attacks. Table 5 summarizes these data. Files with F3 failure were not used to train the model but to evaluate its ability to detect and classify anomalies not seen during training. This dataset will help address the shortage of ensembles for training models that classify anomalies into failures and attack categories. Likewise, it could classify specific types of failures or attacks.

4.2. Context information

The IIoT system influences the cyber–physical dataset, which affects certain variables. For instance, the sound sensor is impacted by the operating frequency of the nearest motor. The frequency of the motors has been used as "context information" for the dataset

[1] The repository link will be provided in the final manuscript but is omitted here to maintain author anonymity.

Table 5
Dataset. The data was stored in 48 files, 31 with normal data, 13 with failures, and 4 with attacks. The anomaly types are, F1: high temperature, F2: miscalibration, F3: disconnection, A2: DoS attack, A1: data injection attack.

Files	Normal	Failures	Attacks	F1	F2	F3	A1	A2
31	1 254 183	0	0	0	0	0	0	0
13	345 547	60 463	0	11	8	2	0	0
4	505 404	0	38 755	0	0	0	8	5
48	**2 105 134**	**60 463**	**38 755**	**11**	**8**	**2**	**8**	**5**

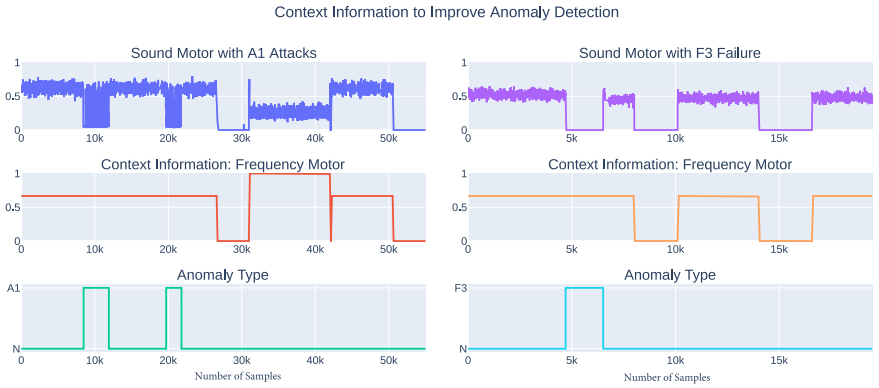

Fig. 11. Context information. By incorporating motor data into the Industrial Internet of Things (IIoT) dataset, it becomes possible to distinguish between normal and abnormal behavior.

Table 6
Effect of context information on anomaly detection and classification models. Context information helps classify all anomalies.

Model	Context	F1	F2	F3	A1	A2	F1-Score
Detector	Yes	✓	✓	✓	✓	✓	0.963900
	No	✓	✓	✗	✓	✓	0.926210
Classifier	Yes	✓	✓	✓	✓	✓	0.900422
	No	✓	✓	✗	✗	✓	0.833197

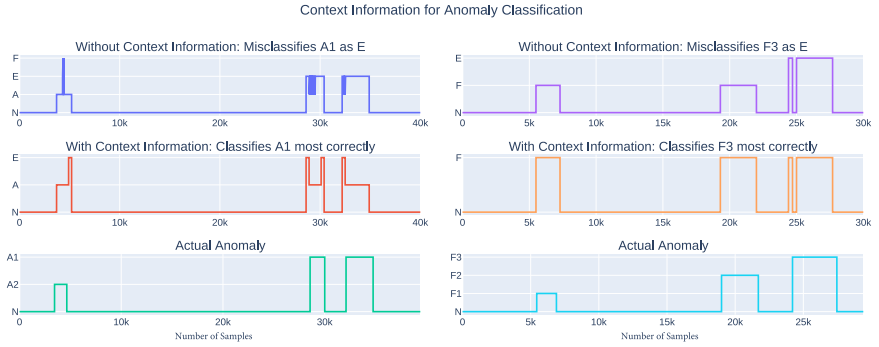

Fig. 12. Context information helps classify attacks correctly (A1). Without context information, A1 is classified as Event.

extracted from the IIoT system. Fig. 11 demonstrates that the frequency of the motor provides insight into the abnormal behavior of the sound signal. This is particularly noticeable during a data injection attack (on the left) or a sensor disconnection failure (on the right). If the sound level fluctuates while the motor frequency remains unchanged, this indicates the presence of an anomaly.

Table 6 summarizes the effect of context information on anomaly detection and classification models. When they do not use the CPS feature, none of the models recognize the sound sensor disconnection failure, and the classifier does not recognize the data injection attack performed on the same variable. The above reduces F1-score in both models.

The negative effect of the lack of context information on the anomaly detector and classifier is graphically observed in Fig. 12. The plot above shows that A1 attacks are classified as events, meaning that the classifier does not know the nature of the anomaly. Meanwhile, models that incorporate context information mostly correctly detect and classify these attacks (in the middle). Below, you can see the actual classification of the anomaly.

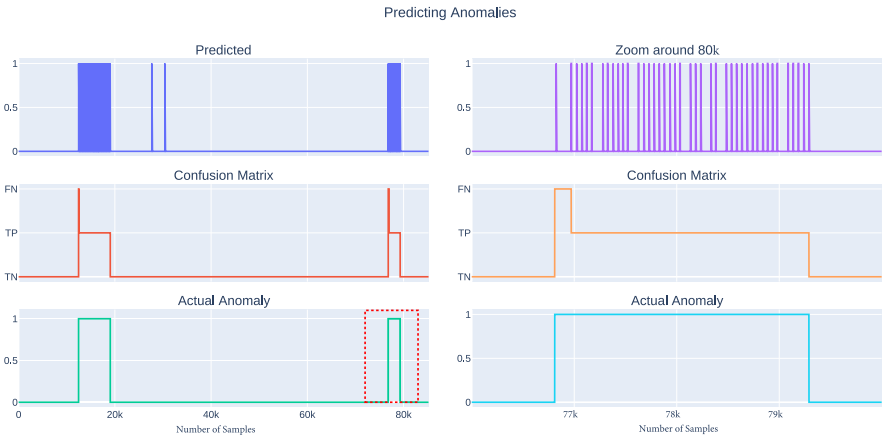

Fig. 13. Some anomalies are detected intermittently, alternating with normal samples. Here is a zoom of around 80k on the right. If at least fifteen samples exceed the threshold value in the output sequence, it is labeled as an anomaly. Here, '1' means anomaly, and '0' is normal.

Table 7
Anomalies detected intermittently affect the metrics related to the confusion matrix (TP, TN, FP, FN) and, therefore, precision, recall, and F1-score. The sequence of the last 300 samples is used to validate if at least 15 samples have been labeled as anomalies to label the new sample.

Prediction	TN	TP	FN	FP	Accuracy	Precision	Recall	F1-Score
300-Sample Sequence	76 119	8839	231	0	0.997288	1	0.974531	0.987101
Counting anomaly points	76 113	518	8552	6	0.899541	0.988549	0.057113	0.107984

4.3. Sliding window to detect anomalies

Some anomalies are not detected sustainably. Instead, samples labeled as anomalies alternate with more samples labeled as normal. Fig. 13 shows this situation, from which we could affirm a priori that the anomaly is correctly detected. However, if we calculate the confusion matrix and precision, recall, and F1-score, a poor performance of the detector is obtained. The strategy used in the detector and the classifier is to take advantage of the sequence of 300 samples with which the model works. If, in the output sequence, there are at least fifteen samples that exceed the threshold value, the sequence is labeled as an anomaly. As a consequence of this algorithm, the anomaly prediction appears as a continuous line in the middle plot of the figure we refer to. Table 7 shows the metrics calculated using the 300-sample sequence and the original anomaly points. The results indicate that the window is necessary for metrics to reflect the model's performance. Another consequence of this detection strategy is that outliers from a single sample are ignored and do not trigger the anomaly alert. The anomaly alert is only activated with 15 or more samples identified as such. It means that *anomalies* lasting less than three seconds are ignored because the sampling rate is 0.2s.

4.4. IIoT anomaly classifier validation

We utilized a cross-validation technique to evaluate the classifier model and ensure that the model performs well on new data. This involves splitting the data into subsets, training the model on some sets, and validating it on the remaining ones. Usually, the dataset is divided into equally sized subsets or folds. However, our data is stored in 15 CSV files of different sizes, each containing one or more failures or attacks. So, each of these files represents a fold. The model is trained on k-1 folds and validated on the remaining fold for each. This process is repeated k times, with each fold used exactly once as the validation data. Performance metrics (precision, recall, F1-score) are calculated for each iteration. The final performance estimate is obtained by averaging these metrics across all folds. The results of this cross-validation are shown in Table 8. The average result provides a precision of 0.952, indicating a very low False Positive (FP) rate, typically related to remaining effects after the anomaly's cause has ceased; recall is 0.878, corresponding to a higher rate of False Negatives (FN) than FP. FN usually occurs right at the beginning of the anomaly due to a delay in detection or in the middle when some samples are erroneously labeled as normal. Finally, the average F1-score is 0.91 for the evaluation set. When calculating precision, recall, and F1-score separately, we obtain 0.945, 0.797, and 0.862 for attacks and 0.954, 0.908, and 0.927 for failures, inferring that our system detects failures better than attacks.

In Fig. 14, you can see the graphical representation of the classification results during cross-validation for the edge1 device. All anomalies were identified, with a few datasets showing a small number of samples labeled as events. In the edge1_13 dataset, the number of samples incorrectly labeled as events slightly increased. After validating 15 samples (3 s) in the input sequence, the classifier in the middle eliminates false positives not associated with real anomalies detected by the detector on the left. The plots on the right display the actual anomalies.

Table 8
Cross-Validation for IIoT Anomaly Classifier.

File	Anomaly	Precision	Recall	F1-Score
edge1_11.csv	F1, F2	0.96	0.82	0.89
edge1_12.csv	F1, F1	1	0.8	0.89
edge1_13.csv	F2, F2	0.81	0.92	0.86
edge1_14.csv	A2	0.9	0.74	0.81
edge2_9.csv	F2, F1	0.89	0.97	0.93
edge2_10.csv	F1, F1	0.99	0.75	0.85
edge2_11.csv	F2	1	1	1
edge2_12.csv	A2, A1, A1, A1, A1	0.91	0.88	0.89
edge3_8.csv	F1, F2	0.97	0.99	0.98
edge3_9.csv	F1, F1	1	0.81	0.9
edge3_10.csv	F2	1	1	1
edge3_12.csv	A2, A1, A1, A1, A1	0.97	0.85	0.91
edge4_10.csv	F1	0.88	1	0.94
edge4_11.csv	F1, F2	1	0.93	0.96
edge4_12.csv	A2, A2	1	0.72	0.84
Total:		0.95	0.88	0.91

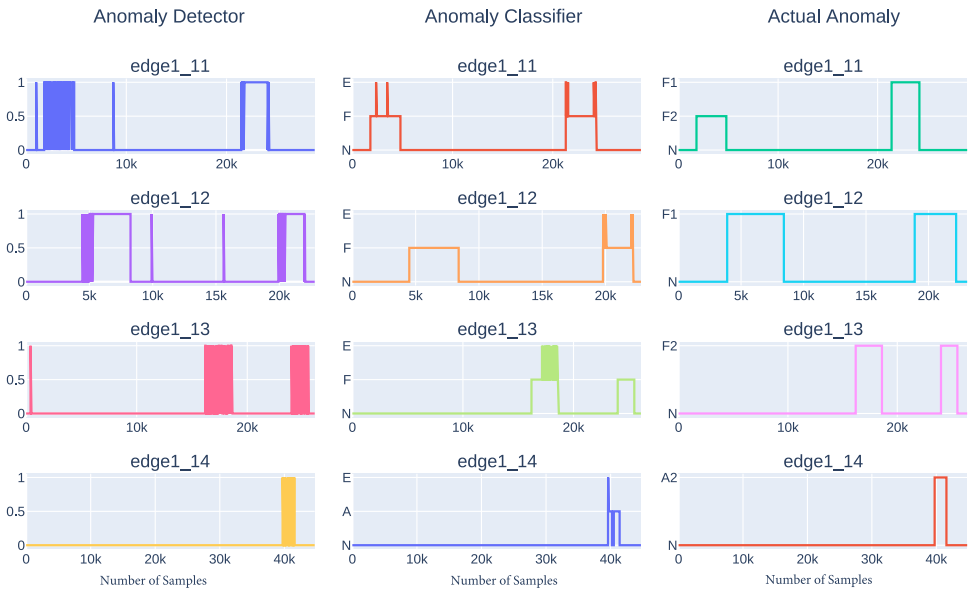

Fig. 14. Graphical result for cross-validation with edge1 device data. Here, '1' represents 'anomaly,' and '0' represents 'normal.' Letters represent A: attack, E: event, F: failure, F1: temperature failure, F2: misconfigured failure, and A2: DoS attack.

4.5. Graphical interpretability of the model

Interpretability is "the ability to explain in understandable terms to a human" [49]. Interpretability is essential in machine learning applications as it helps build trust, enhances understanding, and supports better decision-making. [50,51]. The graphical method is a model-agnostic approach that compares input and output data to create various visualizations. This process helps identify the causes of anomalies and enhances our understanding of the underlying patterns in the data. [52]. This method assumes the input data contains all necessary information to explain the detected anomalies [50,53].

Fig. 15 shows how the difference between the original and reconstructed signals exceeds the threshold value (dashed line) when a temperature failure occurs in the edge device CPU. This way, the user can validate which inputs influence the autoencoder's anomaly detection.

4.6. Comparison with other models

New anomaly detection and a classification models based on LSTM layers were trained to compare their performance with those implemented in the anomaly classification framework. An autoencoder has been implemented using LSTM coupled with dropout layers for both the encoder and decoder. The output is wrapped with a TimeDistributed layer that applies a Dense layer to each time step of the input tensor separately. LSTMs are a type of recurrent neural network (RNN) ideal for handling sequence data, such

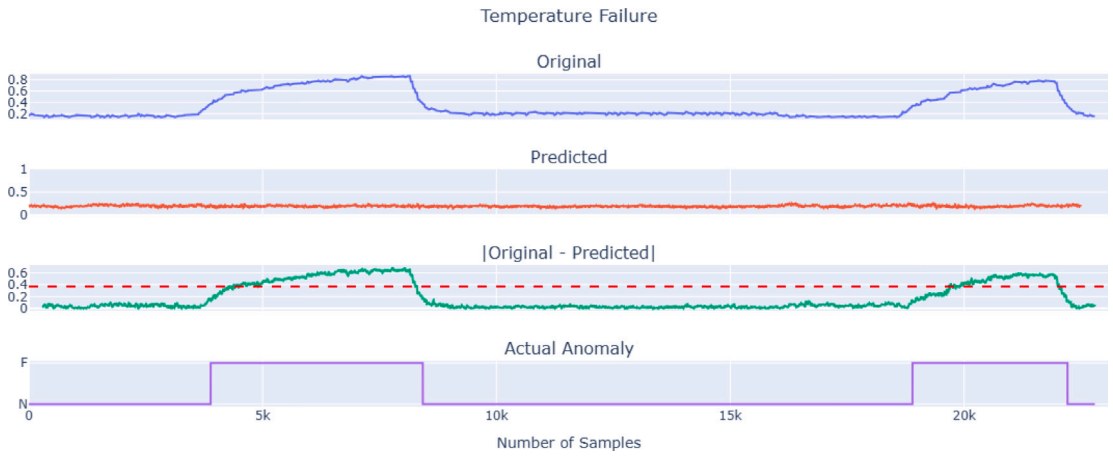

Fig. 15. Graphical interpretability of the model. The signal reconstructed graph by the autoencoder and the threshold value allow the user to observe the features that lead to anomaly detection.

Table 9
Comparison between anomaly detector and classifier models. Conv1d-based autoencoder and transformer-based classifier show better precision, recall, and F1 score metrics.

Block	Model	Precision	Recall	F1-Score
Detector	Conv1d	0.973610	0.913190	0.942433
	LSTM	0.962211	0.870235	0.913915
Classifier	Transformer	0.893578	0.944319	0.918198
	LSTM	0.880367	0.894792	0.887471

as time series, due to their specialized architecture, which is particularly effective at capturing long-term dependencies in sequence data. The LSTM encoder is responsible for compressing the input sequence into a fixed-size context vector that represents the essential features of the input data. At the same time, the decoder tries to reconstruct the original input sequence. The TimeDistributed layer wraps another layer, such as a Dense layer, and applies it to each time step of the input tensor independently. This implies that the same layer weights are used for each time step, but the operation is carried out separately for each slice.

The anomaly classifier uses LSTM layers alternating with Dropout to reduce the risk of overfitting and a TimeDistributed wrapping of a dense layer. An LSTM network comprises a series of LSTM units, each responsible for maintaining a memory cell that can retain information across time steps. Table 9 shows the capacity of each model to avoid false negatives (recall) and false positives (precision) and the harmonic mean of both (F1). Although LSTM-based models also performed well, the convolutional-based autoencoder is the best detector, while the transformer-based classifier performs best in the three metrics.

4.7. Model validation with new anomalies

In the preceding section, we thoroughly analyzed the cross-validation results. The proposed system successfully identified all anomalies and did not produce any false positives that triggered alarms for non-existent anomalies. This section examines the classifier's performance to uncover anomalies not part of the training set. We utilized two files containing sensor disconnection failures (F3) to evaluate the model. The results, as illustrated in Fig. 16, indicate that the anomalies were indeed detected, but they were classified as normal events by the classifier. Consequently, the system activated an alert for the human operator to classify these anomalies manually. While this outcome is not ideal, it is nonetheless noteworthy that the system successfully detected the occurrence of an anomaly.

4.8. Deploying the IIoT anomaly classifier in a testbed

To ensure timely anomaly detection, it is recommended that the classifier be run on each edge device. The deployment results are depicted here. The TFLiteConverter function set from the TensorFlow library generated a lightweight version of the classifier and detector models to run on a Raspberry PI3. Validation is performed with data collected at the same time that the training data was collected. The goal here is to evaluate the performance of the models when running a lightweight version of them on an edge device. Fig. 17 shows the four anomalies evaluated. There are no false positives that generate unnecessary alerts in the graph. In most cases, some samples are classified as events, meaning the detector classified them as anomalies, but the classifier labeled them as normal; however, they are very few compared to the correctly classified data, so the human operator does not doubt whether it is a failure or an attack. The classification graph of the dataset called "edge2_F1" shows a delay in detection; it starts around

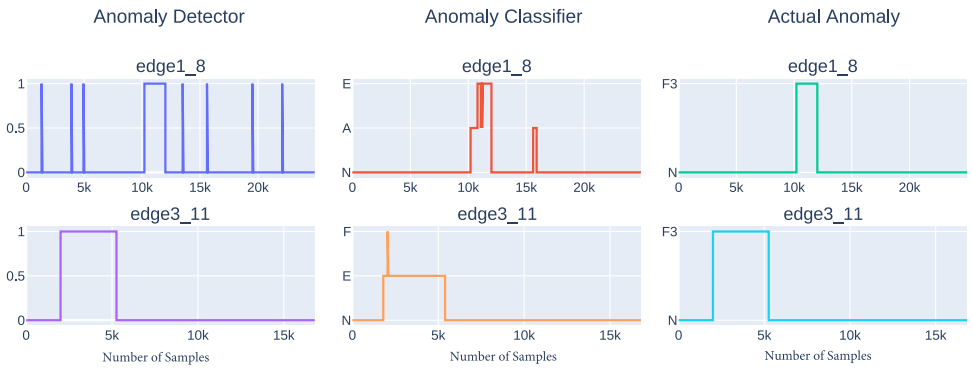

Fig. 16. Model validation with new data. F3 anomalies were detected but classified as events.

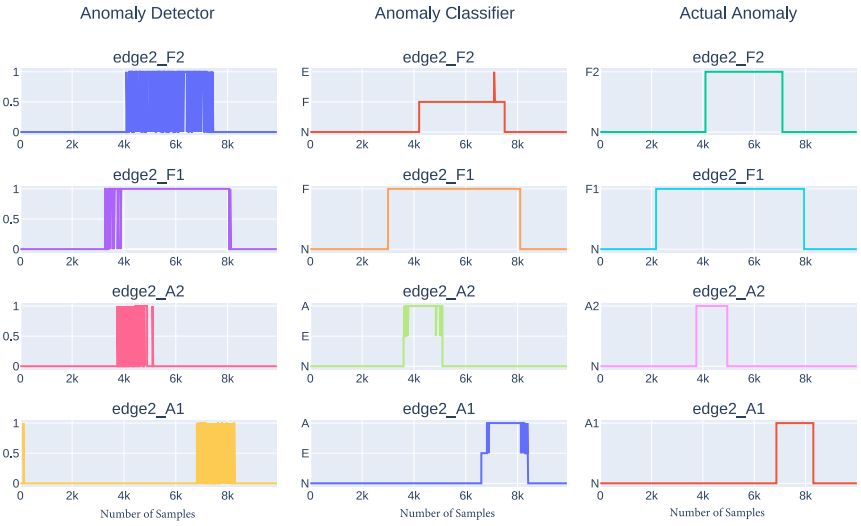

Fig. 17. Evaluating the IIoT anomaly classifier in the testbed. Detector labels samples as normal (0) or anomaly (1), and classifier labels as normal (N), event (E), failure (F), and attack (A).

Table 10
Evaluating the IIoT Anomaly Classifier in a Testbed.

File	Anomaly	Precision	Recall	F1-score
edge2_F2	Misconfigured Failure	0.88	0.97	0.92
edge2_F1	Temperature Failure	1	0.84	0.91
edge2_A1	Data Injection Attack	0.86	0.93	0.89
edge2_A2	DoS Attack	0.91	0.88	0.89
Average		0.91	0.90	0.90

sample 3k, but the original anomaly (on the right) starts near 2k. This is consistent with what was observed throughout the entire experimental process with the temperature failures (F1). Table 10 shows the precision, recall, and F1-score values of models running on the edge device. They are consistent with those obtained using the original model in Google Collaboratory, where the system shows slightly better performance classifying failures than attacks.

Table 11 shows the size of the files containing the lightweight versions of the detector and classifier created with LTLite. Also, it shows the average time the edge device takes to run these models a single time with a sequence of 300 samples. This time does not include sample collection and preprocessing, which could be done parallel with evaluating the models.

Table 11
Execution time of the models on the testbed for a sequence of 300 samples.

Model	Size (kB)	Time (s)
Detector (Autoencoder)	361	0.06799
Classifier (Transformer)	116	0.074955

5. Discussion

The results demonstrate the effectiveness and robustness of the Framework to implement IIoT Anomaly classifiers within an Industrial Internet of Things (IIoT) system. The case study's key contributions include creating a comprehensive IIoT dataset with labeled anomalies. The dataset generated for this study fills a significant gap in available IIoT datasets, which often lack the diversity and specificity needed for comprehensive anomaly detection and classification. By using a testbed that emulates a real-life four-belt conveyor system, we were able to produce a dataset with detailed labels for various types of anomalies, including high temperature (F1), miscalibration (F2), disconnection (F3), Denial of Service (DoS) attacks (A2), and data injection attacks (A1).

The unsupervised anomaly detection model architecture (64-32-7 Conv1D layers in the encoder and 7-32-64 Conv1DT layers in the decoder) provided the best balance of precision, recall, and F1-score. This model's optimization, using metrics that do not involve true negatives (TN), ensures a realistic evaluation of its effectiveness in practical scenarios where false positives (FP) and false negatives (FN) are critical considerations. Using a sliding window technique to validate anomaly detection significantly improved the model's performance. The model reduces the likelihood of false alarms triggered by transient outliers by requiring at least 15 consecutive samples to exceed the threshold value before labeling an anomaly. This strategy aligns with the operational requirements of real-world CPS and IIoT systems, where stability and reliability are paramount.

The two-stage classification approach, involving an initial detection by the autoencoder followed by a detailed classification using a transformer-based model, proved effective. With its optimized architecture (3 encoders and 20 head attentions), the transformer-based classifier achieved high precision and recall, particularly distinguishing between failures and attacks. The cross-validation results, with an average precision of 0.952 and an F1-score of 0.91, indicate that the model is reliable and can generalize well to new data. One notable aspect of the study was the model's ability to detect but not classify unseen anomalies (e.g., sensor disconnection failures, F3). These anomalies were identified but labeled as normal events, triggering an alert for human intervention. This outcome highlights the model's sensitivity and the importance of having a fallback mechanism where human operators can review and classify new types of anomalies. This feature is crucial for maintaining the system's adaptability and continuous improvement.

This anomaly detection and classification framework for an Industrial Internet of Things (IIoT) system is designed to be modular and flexible, allowing it to be adapted to various industrial environments. Since it includes tasks ranging from data collection and anomaly selection to generating alerts related to different types of anomalies. It can be applied to different types of machinery, sensor configurations, and operational conditions by focusing on essential aspects such as data collection, processing, anomaly detection, and classification techniques. The framework can be customized to meet specific industry requirements, enabling the detection of anomalies in areas like manufacturing processes, energy management systems, or supply chain monitoring. Furthermore, its capability to handle diverse data types and varying levels of variability allows it to scale and integrate seamlessly with both new and legacy industrial systems. This flexibility supports real-time insights and predictive maintenance across multiple sectors.

6. Conclusion

In this work we propose a dual-model framework for anomaly detection and classification for data-intensive environments, such as those encountered in IIoT systems. The separate but complementary models for detecting and classifying anomalies leverage advanced machine learning techniques, optimized processing methods, and contextual integration to significantly enhance anomaly management's accuracy and efficiency. Our comprehensive framework describes a new approach to detecting and categorizing anomalies in IIoT environments. Combines CPS domain knowledge with data-driven insights from IIoT technologies. This framework addresses the challenges associated with anomaly classification and provides a solution that can be adapted to various applications, from cybersecurity to industrial monitoring. By integrating advanced machine learning techniques with robust preprocessing and evaluation methods, this framework offers a powerful tool for identifying and responding to anomalies.

Future work in enhancing our IIoT anomaly classification framework focuses on addressing its limitations related to scalability, adaptability, and robustness. Key areas for improvement include integrating diverse data sources such as time series, images, and sounds and leveraging distributed architectures like federated learning to enhance anomaly detection at both local and global levels. Although our framework detected new anomalies, it could not classify them correctly; a new version could include clustering techniques to classify new anomalies between failures and attacks automatically. Future work can also evaluate models that detect anomalies of short duration and other techniques to improve models' interpretability. Incorporating domain-specific knowledge, advanced variable selection and data augmentation techniques, and security measures will strengthen the framework's performance. Expanding the framework to other industrial sectors, fostering self-improving systems, and encouraging interdisciplinary collaborations will refine its adaptability, ensuring its effectiveness in dynamic industrial environments.

CRediT authorship contribution statement

Martha Rodríguez: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Diana P. Tobón:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Danny Múnera:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of Generative AI and AI-assisted technologies in the writing process

While preparing this work, the authors used Grammarly and ChatGPT-4 to check for spelling, grammar, and syntax errors and improve readability. After using this tool/service, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

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

Data will be made available on request.

References

[1] H. Benaddi, M. Jouhari, K. Ibrahimi, J. Ben Othman, E.M. Amhoud, Anomaly detection in industrial IoT using distributional reinforcement learning and generative adversarial networks, Sensors 22 (21) (2022) 8085.
[2] J. Li, M.S. Othman, H. Chen, L.M. Yusuf, Optimizing IoT intrusion detection system: feature selection versus feature extraction in machine learning, J. Big Data 11 (1) (2024) 36.
[3] X. Liu, Y. Du, Towards effective feature selection for iot botnet attack detection using a genetic algorithm, Electronics 12 (5) (2023) 1260.
[4] A. Karkouch, H. Mousannif, H. Al Moatassime, T. Noel, Data quality in internet of things: A state-of-the-art survey, J. Netw. Comput. Appl. 73 (2016) 57–81.
[5] J. Byabazaire, G. O'Hare, D. Delaney, Data quality and trust: Review of challenges and opportunities for data sharing in iot, Electronics 9 (12) (2020) 2083.
[6] M. Soori, B. Arezoo, R. Dastres, Internet of things for smart factories in Industry 4.0, a review, Internet Things Cyber-Phys. Syst. (2023).
[7] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, D.O. Wu, Edge computing in industrial Internet of Things: Architecture, advances and challenges, IEEE Commun. Surv. Tutor. 22 (4) (2020) 2462–2488.
[8] A.A. Mirani, G. Velasco-Hernandez, A. Awasthi, J. Walsh, Key challenges and emerging technologies in industrial IoT architectures: A review, Sensors 22 (15) (2022) 5836.
[9] H. Nizam, S. Zafar, Z. Lv, F. Wang, X. Hu, Real-time deep anomaly detection framework for multivariate time-series data in industrial iot, IEEE Sens. J. 22 (23) (2022) 22836–22849.
[10] H. Zhang, Y. Xia, T. Yan, G. Liu, Unsupervised anomaly detection in multivariate time series through transformer-based variational autoencoder, in: 2021 33rd Chinese Control and Decision Conference, CCDC, IEEE, 2021, pp. 281–286.
[11] S. Tuli, G. Casale, N.R. Jennings, Tranad: Deep transformer networks for anomaly detection in multivariate time series data, 2022, arXiv preprint arXiv:2201.07284.
[12] H.T. Truong, B.P. Ta, Q.A. Le, D.M. Nguyen, C.T. Le, H.X. Nguyen, H.T. Do, H.T. Nguyen, K.P. Tran, Light-weight federated learning-based anomaly detection for time-series data in industrial control systems, Comput. Ind. 140 (2022) 103692.
[13] C. Ding, J. Zhao, S. Sun, Concept drift adaptation for time series anomaly detection via transformer, Neural Process. Lett. 55 (3) (2023) 2081–2101.
[14] M. Rodríguez, D.P. Tobón, D. Múnera, Anomaly classification in industrial Internet of Things: A review, Intell. Syst. Appl. (2023) 200232.
[15] B. Yang, J. Ye, S. Coshatt, W. Song, F. Zahiri, Data-driven approach for detection of physical faults and cyber attacks in manufacturing motor drives, in: 2022 IEEE Energy Conversion Congress and Exposition, ECCE, IEEE, 2022, pp. 1–6.
[16] K. Zhang, C. Keliris, T. Parisini, M.M. Polycarpou, Identification of sensor replay attacks and physical faults for cyber-physical systems, IEEE Control Syst. Lett. 6 (2021) 1178–1183.
[17] M. Ganjkhani, M. Gilanifar, J. Giraldo, M. Parvania, Integrated cyber and physical anomaly location and classification in power distribution systems, IEEE Trans. Ind. Inform. 17 (10) (2021) 7040–7049.
[18] M. Younan, E.H. Houssein, M. Elhoseny, A.A. Ali, Challenges and recommended technologies for the industrial internet of things: A comprehensive review, Measurement 151 (2020) 107198.
[19] L. Liang, S. Liu, Event-triggered distributed attack detection and fault diagnosis, IEEE Trans. Instrum. Meas. 72 (2022) 1–11.
[20] M. Ryalat, H. ElMoaqet, M. AlFaouri, Design of a smart factory based on cyber-physical systems and internet of things towards Industry 4.0, Appl. Sci. 13 (4) (2023) 2156.
[21] Z. Niu, W. Guo, J. Xue, Y. Wang, Z. Kong, L. Huang, A novel anomaly detection approach based on ensemble semi-supervised active learning (ADESSA), Comput. Secur. 129 (2023) 103190.
[22] E. de Matos, R.T. Tiburski, C.R. Moratelli, S. Johann Filho, L.A. Amaral, G. Ramachandran, B. Krishnamachari, F. Hessel, Context information sharing for the Internet of Things: A survey, Comput. Netw. 166 (2020) 106988.
[23] G. Sun, J. Li, J. Dai, Z. Song, F. Lang, Feature selection for IoT based on maximal information coefficient, Future Gener. Comput. Syst. 89 (2018) 606–616.
[24] A.N. Tarekegn, M. Giacobini, K. Michalak, A review of methods for imbalanced multi-label classification, Pattern Recognit. 118 (2021) 107965.
[25] L. Cui, Z. Dong, H. Xu, D. Zhao, Triplet attention-enhanced residual tree-inspired decision network: A hierarchical fault diagnosis model for unbalanced bearing datasets, Adv. Eng. Inform. 59 (2024) 102322.
[26] I. Silva, J. Eugenio Naranjo, A systematic methodology to evaluate prediction models for driving style classification, Sensors 20 (6) (2020) 1692.

[27] P. Yan, A. Abdulkadir, P.-P. Luley, M. Rosenthal, G.A. Schatte, B.F. Grewe, T. Stadelmann, A comprehensive survey of deep transfer learning for anomaly detection in industrial time series: Methods, applications, and directions, IEEE Access (2024).
[28] I. Araf, A. Idri, I. Chairi, Cost-sensitive learning for imbalanced medical data: a review, Artif. Intell. Rev. 57 (4) (2024) 1–72.
[29] P. Mooijman, C. Catal, B. Tekinerdogan, A. Lommen, M. Blokland, The effects of data balancing approaches: A case study, Appl. Soft Comput. 132 (2023) 109853.
[30] X. Berger, RPI monitor, 2022, https://github.com/XavierBerger/RPi-Monitor.
[31] Y. Li, X. Peng, J. Zhang, Z. Li, M. Wen, DCT-GAN: dilated convolutional transformer-based GAN for time series anomaly detection, IEEE Trans. Knowl. Data Eng. 35 (4) (2021) 3632–3644.
[32] J. Kim, H. Kang, P. Kang, Time-series anomaly detection with stacked transformer representations and 1D convolutional network, Eng. Appl. Artif. Intell. 120 (2023) 105964.
[33] G. Du, J. Zhang, N. Zhang, H. Wu, P. Wu, S. Li, Semi-supervised imbalanced multi-label classification with label propagation, Pattern Recognit. (2024) 110358.
[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.
[35] A. Chevrot, A. Vernotte, B. Legeard, CAE: Contextual auto-encoder for multivariate time-series anomaly detection in air transportation, Comput. Secur. 116 (2022) 102652.
[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[37] A.A. Mantha, A. Hussain, G. Ravikumar, HIL testbed-based auto feature extraction and data generation framework for ML/DL-based anomaly detection and classification, in: 2024 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference, ISGT, IEEE, 2024, pp. 1–5.
[38] D.J. Atul, R. Kamalraj, G. Ramesh, K.S. Sankaran, S. Sharma, S. Khasim, A machine learning based IoT for providing an intrusion detection system for security, Microprocess. Microsyst. 82 (2021) 103741.
[39] B. Si, Z. Ni, J. Xu, Y. Li, F. Liu, Interactive effects of hyperparameter optimization techniques and data characteristics on the performance of machine learning algorithms for building energy metamodeling, Case Stud. Therm. Eng. (2024) 104124.
[40] F. Zeng, M. Chen, C. Qian, Y. Wang, Y. Zhou, W. Tang, Multivariate time series anomaly detection with adversarial transformer architecture in the internet of things, Future Gener. Comput. Syst. 144 (2023) 244–255.
[41] Y. Fang, Y. Yao, X. Lin, J. Wang, H. Zhai, A feature selection based on genetic algorithm for intrusion detection of industrial control systems, Comput. Secur. 139 (2024) 103675.
[42] X. Sáez-de Cámara, J.L. Flores, C. Arellano, A. Urbieta, U. Zurutuza, Clustered federated learning architecture for network anomaly detection in large scale heterogeneous IoT networks, Comput. Secur. 131 (2023) 103299.
[43] J. Jithish, B. Alangot, N. Mahalingam, K.S. Yeo, Distributed anomaly detection in smart grids: a federated learning-based approach, IEEE Access 11 (2023) 7157–7179.
[44] X. Wang, Y. Wang, Z. Javaheri, L. Almutairi, N. Moghadamnejad, O.S. Younes, Federated deep learning for anomaly detection in the internet of things, Comput. Electr. Eng. 108 (2023) 108651.
[45] M. Shafiq, Z. Tian, A.K. Bashir, X. Du, M. Guizani, CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine-learning techniques, IEEE Internet Things J. 8 (5) (2020) 3242–3254.
[46] F.A. Mazarbhuiya, M. Shenify, A mixed clustering approach for real-time anomaly detection, Appl. Sci. 13 (7) (2023) 4151.
[47] M.A. Belay, S.S. Blakseth, A. Rasheed, P. Salvo Rossi, Unsupervised anomaly detection for iot-based multivariate time series: Existing solutions, performance analysis and future directions, Sensors 23 (5) (2023) 2844.
[48] J.U. Ko, K. Na, J.S. Oh, J. Kim, B.D. Youn, A new auto-encoder-based dynamic threshold to reduce false alarm rate for anomaly detection of steam turbines, Expert Syst. Appl. 189 (2022) 116094.
[49] P. Linardatos, V. Papastefanopoulos, S. Kotsiantis, Explainable ai: A review of machine learning interpretability methods, Entropy 23 (1) (2020) 18.
[50] D. Prasad, S. Hampapura Sripada, Neural network-based anomaly detection models and interpretability methods for multivariate time series data, 2023.
[51] D. Mane, A. Magar, O. Khode, S. Koli, K. Bhat, P. Korade, Unlocking machine learning model decisions: A comparative analysis of LIME and SHAP for enhanced interpretability, J. Electr. Syst. 20 (2s) (2024) 1252–1267.
[52] R. Jiang, Y. Xue, D. Zou, Interpretability-aware industrial anomaly detection using autoencoders, IEEE Access 11 (2023) 60490–60500.
[53] Q.s. Zhang, S.C. Zhu, Visual interpretability for deep learning: a survey, Front. Inf. Technol. Electron. Eng. 19 (1) (2018) 27–39.