A Deep Neural Framework for Fault Detection in IoT-based Sensor Networks*

Simona Cicero[4], Massimo Guarascio[1][0000−0001−7711−9833], Antonio Guerrieri[1][0000−0003−1469−9484], Simone Mungari[1,2,3][0000−0002−0961−4151], and Andrea Vinci[1][0000−0002−1011−1885]

[1] ICAR-CNR, Rende (CS), Italy. Email: name.surname[at]icar.cnr.it
[2] University of Calabria, Rende (CS), Italy
[3] Revelis s.r.l., Rende (CS), Italy
[4] Independent Researcher, Amantea (CS), Italy. Email: simona.cicero@gmail.com

Abstract. Effective disaster management and early fault detection are crucial for preserving safety and operational efficiency in smart workplaces and buildings. These environments rely on networks of interconnected IoT sensors to monitor parameters such as temperature, humidity, occupancy, and energy usage. Promptly identifying and responding to anomalies or faults using data from these sensors is vital to prevent minor issues from escalating into major disruptions or safety hazards. Recently, Artificial Intelligence techniques have proven to be effective solutions in this scenario, offering sophisticated tools for analyzing complex data patterns. Nevertheless, developing reliable models needs to face a number of important challenges, including the lack of labeled data, unbalanced class distributions, and the need for zero-shot anomaly detection. Moreover, models must be lightweight to run efficiently on IoT devices with limited computational resources and energy supply. To address these challenges, we propose an unsupervised deep learning framework for fault detection in IoT-based sensor networks. Our solution employs a `Sparse U-Net` architecture, an autoencoder equipped with skip connections and sparse layers to enhance learning and robustness to noise. It uses online training with a sliding-window approach to adapt to evolutions in data distribution without requiring prior knowledge of anomalies. Extensive experimentation on a real test case demonstrates the framework's effectiveness and efficiency in ensuring the safety of IoT-based smart workplaces.

Keywords: Sensor Networks · Anomaly Detection · Deep Learning · Safety · Fault Detection · Green AI

1 Introduction

Effective disaster management and the early detection of faults are critical components for maintaining safety and operational efficiency in smart workplaces and buildings. In these environments, networks of interconnected IoT sensors

* All authors made equal contributions to this work.

are typically used to monitor diverse kinds of parameters, such as occupancy, humidity, temperature, and energy usage [26]. The ability to promptly identify and respond to anomalies or faults in these systems using the generated data by these networks is crucial for preventing minor issues from escalating into major disruptions or safety hazards.

In more detail, early fault detection involves continuously analyzing data streams from these sensors to reveal any deviations from normal operating conditions. By detecting anomalies early, it is possible to address potential problems before they result in system failures, equipment damage, or unsafe conditions for occupants. For instance, identifying an overheating HVAC system component early can prevent a potential fire hazard and avoid costly repairs or replacements. Similarly, detecting irregular patterns in energy consumption can help pinpoint inefficiencies or unauthorized usage, contributing to overall operational efficiency.

In the context of disaster management, early detection systems can significantly improve response times and outcomes during emergency situations. Recently, Artificial Intelligence (AI) and Machine Learning methods emerged as effective solutions to effectively address this task [1]. In particular, they allow for detecting faults early by providing sophisticated tools for analyzing complex data patterns. These algorithms learn from historical data, determining normal behavior and recognizing subtle deviations that may indicate potential issues.

In more detail, using the Deep Learning (DL) paradigm has become the most frequent approach for detecting unexpected behaviors in various application scenarios. DL-based architectures effectively allow for learning anomaly detection models by blending raw data produced by different types of sensors. Deep Neural Networks (DNNs) learn using a hierarchical approach: several levels of non-linear processing units are arranged in a tiered structure. Each architecture level permits the extraction of features at progressively higher levels of abstraction. Consequently, DL-based methods excel at deriving complex data abstractions and representations at multiple levels [8,13]. They are particularly adept at analyzing low-level data in various formats and from different sources.

Nevertheless, developing reliable models requires addressing a number of difficult issues: *(i)* the lack of labeled data, as many real-life contexts do not have labeled datasets and typically include only expected behaviors during the training stage; *(ii)* unbalancing among the classes, "anomalous" classes represent rare events i.e., limited amount of the whole dataset; and *(iii)* zero-shot outlier detection, which consists in recognizing anomalies never emerged before.

In addition, the devised solution should be *Green AI* compliant, as the requirements of the addressed problem. Detection models should be based on lightweight neural architectures: IoT devices in smart workplaces and buildings often operate under strict resource constraints, with limited computational resources and energy supply. The use of lightweight models ensures that fault detection algorithms can run efficiently on these devices without draining their resources.

To tackle the issues mentioned above, this work proposes an unsupervised deep learning framework for revealing faults in IoT-based sensor networks. In more detail, we have developed a Deep Learning-based detection model to ensure safety in Workplaces equipped with IoT-based devices. This model can be learned without prior evidence of abnormal behaviors. Additionally, it employs online training with a sliding-window technique that allows for adapting to the data distribution evolution (*drift*). The adopted neural model is based on the `Sparse U-Net` architecture leveraged in [3]. Essentially, it is an autoencoder (AE) architecture that is equipped with skip connections to speed up the training procedure and sparse dense levels to reduce the effect of the noise on the detector performances. We designed the approach to be unsupervised (labeled data are not required) and lightweight. Hence, it can also be deployed on resource-constrained devices. We conducted extensive experimentation on a real test case to demonstrate both the effectiveness and efficiency of the framework.

The remainder of the work is structured as follows: In Section 2, we survey the current approaches for anomaly detection in Smart Workplaces/Buildings. Section 3 presents the case study considered in this work. Section 4 describes the unsupervised DNN model we used and depicts the detection mechanism utilized to identify unexpected behaviors. In Section 5, we discuss the experimental results achieved for the analyzed case study. Finally, Section 6 concludes the work and proposes future research directions.

2 Related Works

The continuous production of large volumes of data traffic by modern systems has amplified the need for advanced fault management and prevention techniques. In IoT systems, in particular, identifying anomalies in the data streams generated by sensors demands sophisticated and non-trivial approaches.

A wide range of models have been developed over the years, utilizing various frameworks to address different challenges. That is, in this Section, we will cover only the subset of the proposed models closer to our domain.

For instance, in the field of *Internet of Vehicles* (IoV), the authors in [30] utilize a single LSTM model that integrates temporal and data dimensions in its input. The model result is compared against actual data to detect potential anomalies. Similarly, in [7], the authors aim to recognize unexpected events in time series generated by IoT devices. The devised solution, named TSMAE, employs a sequence-to-sequence neural model built on LSTMs to replicate the data that is in input toward the output. Also, the model incorporates a memory module [27], which enhances processing by combining historical data with current observations.

Recently, distributed approaches e.g., federated learning [15, 23], and edge computing [12] have been employed for developing efficient anomaly detection solutions across various applications. These frameworks enable each node belonging to the network to analyze its own local data, eliminating the necessity for data transmission to external units like the cloud. This paradigm is leveraged

in READ-IoT [29], which combines a deductive and inductive approach for identifying anomalies. However, READ-IoT is tailored for specific scenarios, further limiting its applicability. Similarly, in [5], the authors propose a rule-based classifier built from the observed behavior of smart devices, whose aim is to detect and explain patterns of behavioral anomalies.

Authors of [28] propose an approach based on Kalman filters and Convolutional Neural Networks for the real-time detection of sensors' anomalies in the context of Connected And Automated Vehicles. The approach trains a CNN to classify a frame of a multi-sensor time series as normal or anomalous, and the result is further refined by a proper Kalman filter. The approach is assessed on a real-world dataset, but enriched with simulated anomalies.

The work in [10] proposes a framework for distributed supervised sensor-fault diagnosis and detection that is based on AEs, Support vector Machines, and Fuzzy Deep Neural Networks, having the goal of realizing a quick and lightweight fault detection system. The approach was assessed against an artificial and labeled dataset obtained by injecting six kinds of faults on a real-acquired sensor stream. The reviewed works [10,16,28] exploit supervised approaches trained on labeled data, but in real-world scenarios, rarely labeled anomalies are available for the training phase. As a result, unsupervised approaches are often favored over supervised ones.

Other models such as Convolutional Neural Networks [28], Generative Adversarial Networks [14,19] and Reinforcement Learning [18,24] have been proposed to detect anomalies in such contexts, thanks to their representation capabilities.

Our approach differs from the ones presented in this section since it is focused on making safe workplaces monitored by sensor networks through the usage of unsupervised DNNs combined with a sliding window technique.

3 Case Study: `ICAR-LAB`

The case study was implemented in the IoT Laboratory of the ICAR-CNR headquarters in Italy. The laboratory was equipped with a series of IoT devices, including smart sensors and actuators. In particular, ZigBee and Wi-Fi sensors were employed for data acquisition.

As regards the ZigBee sensors, they include: (*i*) a temperature and humidity sensor; (*ii*) a light sensor; (*iii*) a Passive InfraRed (PIR) sensor; (*iv*) two contact switch sensors to detect the laboratory's door and window status (opened/closed); (*v*) a smoke sensor to detect if smoke in the laboratory exceeds a predetermined threshold.

The Wi-Fi sensors employed are the following: (*i*) a current sensor to monitor a fan installed in the laboratory for exchanging air with the outside; (*ii*) a set of sensors for monitoring the indoor air quality. The set comprehends a gas concentration sensor, a carbon dioxide *(CO_2)* sensor, and a carbon monoxide *(CO)* sensor; (*iii*) a sensor measuring the percentage of openness of the curtain in the laboratory; and (*iv*) three smart lamps sending their status.

Figure 1 shows, according to the Design Template introduced in [11], the key components developed for the case study, the flow of information between them, and the type of computing machine on which they were installed and executed. The components, in more detail, include:

- *ZigBee Coordinator*. It acts as a central hub for the ZigBee sensors deployed for environmental monitoring and is connected to each other with a mesh topology. It operates as a gateway between the ZigBee sensor network and the MQTT broker;
- *WiFi Sensors*. All the Wi-Fi devices listed above host their own MQTT client, which allows direct data sending to a Broker;
- *MQTT Broker*. It has been conceived to facilitate communication between the various components of the developed system. The broker (following the publish/subscribe paradigm) collects messages from some of these components and forwards them to the intended destinations;
- *Persistence Agent*. A persistence agent has been implemented to act as an intermediary between the MQTT Broker and the DB;
- *DB*. A MariaDB relational database management system ensures data storage and availability;
- *Data Preprocessor*. This component takes the raw data, which is stored in the database, and pre-processes it so as to prepare the input for the Deep Anomaly Detector Builder;
- *Deep Anomaly Detector Builder*. This component uses the data pre-processed by the Data Preprocessor to train the Deep Anomaly Detector. Once trained, the detector is passed to the Predictor component;
- *Predictor*. It makes predictions based on the data coming from the sensors using the detector trained by the Builder component.

The components ZigBee Coordinator, WiFi Sensors, MQTT Broker, Persistence Agent, and DB are all hosted on a single Raspberry Pi, indicated in the figure as Edge#1, which is powerful enough to handle all of them. The Data Preprocessor, Deep Anomaly Detector Builder, and Predictor components are hosted on another Raspberry Pi node Edge#2. It is worth noting that, although this solution was developed for Raspberry, in this paper, the node Edge#2 will be emulated on a more powerful PC for faster prototyping.

The reference dataset contains 18,430 tuples, corresponding to measurements recorded in real-time (every minute) by the sensors located inside the laboratory in November 2022, and it consists of the following features:

- *normalized_time*, timestamp type;
- *illuminance, blinds.openness, lamp.00.brightness, lamp.01.brightness, lamp.02.brightness, CO, CO_2, temperature, humidity*, numerical type;
- *occupancy*, binary type (0 = not occupied, 1 = occupied);
- *fan*, binary type (0 = off, 1 = on);
- *door.closed, window.closed*, binary type (0 = open, 1 = closed);

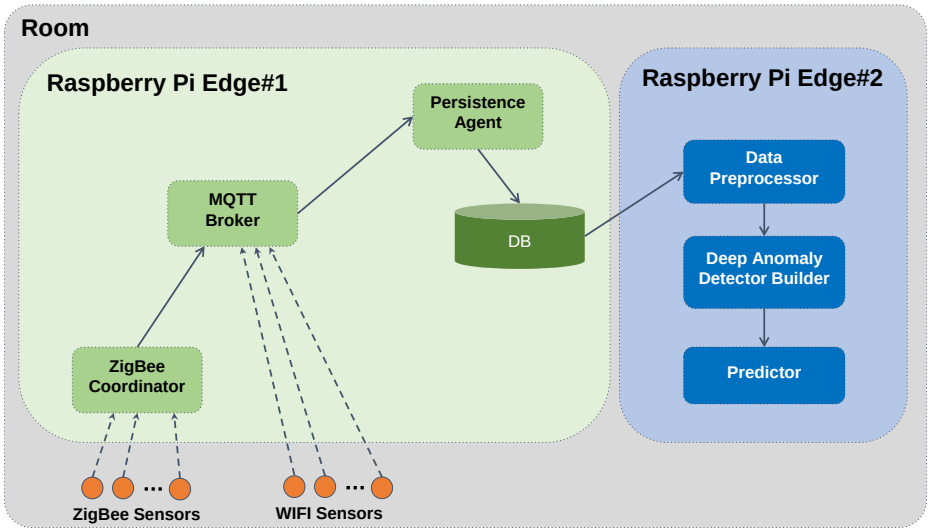

Fig. 1. Main components implemented for the case study.

– *carbon_ monoxide, smoke, gas*, binary type (0 = absent, 1 = present).

Therefore, all attributes are binary and numerical type except for *normalized_ time*, which is configured as a timestamp indicating the time when the data was recorded.

To stress the effectiveness of the detector (even in the absence of some sensors), the fields *carbon_ monoxide*, *smoke*, and *gas* were ignored, as they would have been too indicative of the presence of certain types of anomalies.

The initial dataset was then split temporally into a Training set and a Test set with different sizes: the first 80% of the dataset's tuples are used for the training phase, while the last 20% fills the test set, totaling 14,744 and 3,686 tuples, respectively.

3.1 Synthetic Outlier Generation

In this section, we define the mechanism adopted to inject unexpected behaviors in the data collected from the devices introduced above. This protocol was already introduced in literature in [4] and synthesized in the following.

Considering i as the i-th feature randomly picked from the set of features of the test set, we generated 3 different types of anomalies, namely:

– The *Peak Anomalies*. This anomaly simulates a sensor malfunction and the actual value x_i is replaced with $anomaly(x_i)$ calculated as follows:

$$anomaly(x_i) = \mu_i \pm \alpha \cdot \sigma_i$$

in which α is a real number in $[2, 4]$, μ_i is the average value of feature i, and σ_i is its variance.

- The *Sensor Fault Anomalies*. The value of the picked feature is set to zero in order to mimic a sensor failure. The time interval we set for this anomaly is 15-minutes.
- The *Expert-Induced Anomalies*. The expert intentionally introduces these outliers to mimic 3 distinct settings: *(i)* a fire in the laboratory, *(ii)* a room with a window left opened, and *(iii)* people remaining in the controlled place during after the work time (night). These types of unexpected behaviors concern simultaneous changes in multiple features to simulate a realistic event in the workplace (e.g., if a fire happens, the CO_2 level increases pretty much along with the temperature, and the humidity decreases; when a window is opened, the CO_2 level decreases along with the temperature, but the humidity increases).

Below, we listed how many anomalous events we have generated for each type:

- 100 peak anomalies;
- 25 sensor fault anomalies;
- 10 expert-induced anomalies.

The subsequent step after the anomaly injection in each test set is to group data into time slots according to the strategy introduced in Section 4.2 to compute the metrics to feed the neural model. The sensor measurements are yielded each minute while the statistics are calculated with a 5-minute time window with a step of 1 minute.

4 Neural-based Solution

Here, we define the neural solution developed to detect anomalies in IoT-equipped Environments. Our detector is trained in an unsupervised fashion while including a preprocessing phase to improve its effectiveness.

The primary advantage of using an unsupervised method lies in its ability to activate alarms for unseen abnormal events. This is particularly relevant in IoT-equipped places where incidents and safety cases are often undocumented and unknown in advance.

4.1 DNN Model

The basic concept behind the devised system is to utilize an *autoencoder* (AE) architecture learned against data gathered by a network of IoT devices. In essence, an AE is a Multi-Layer Perceptron (MLP) trained in an unsupervised way (i.e., trained against examples belonging to a single class) that fulfills two main steps: (*i*) it maps the data to be used to feed the model (i.e., a number statistics calculated over the values generated by the available sensors) into a compressed representation (named, *latent space*); (*ii*) it attempts to reconstruct the original input data from this latent representation.

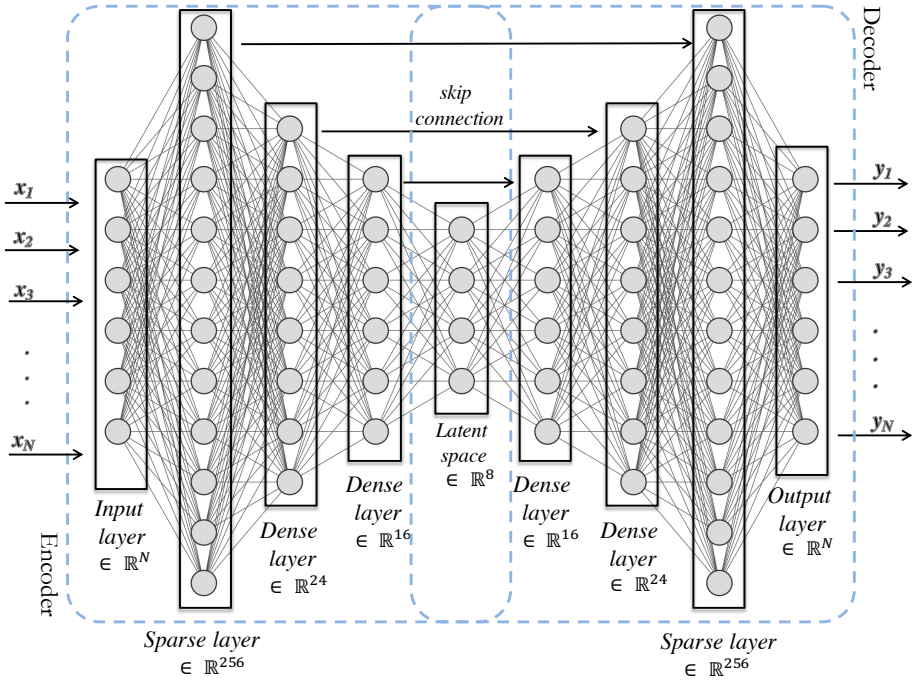

Fig. 2. Anomaly detection model (`Sparse U-Net`) adopted to reveal the presence of fault.

In our scenario, the model is exclusively learned from normal data, where anomalies are absent. The key idea is that data representing normal examples will be reconstructed by the AE with high accuracy. In a nutshell, the two main phases of compression and decompression should not significantly distort the result. By contrast, input anomalies should cause the autoencoder to produce an "abnormal" result. Operationally, the anomaly detector is initially learned on a small portion of normal tuples selected by the operator. Hence, as new data becomes available for the training phase, those most likely to be "normal" are utilized to train the detector further. Anomaly detection is performed based on the discrepancy between the data provided to the model and its result after the computation.

While the reconstruction error has already been used in literature to estimate the anomaly degree of an input example, its application for IoT-equipped buildings (especially smart factories) is again under investigation. As outlined in [2, 9], AEs are considered an effective method for distilling the essential information of an input into a compact, low-dimensional format. Essentially, these DNNs strive to generate an output that nearly replicates the original input.

The neural architecture used to perform the detection is shown in Figure 2 [3]. Essentially, it comprises two primary components, respectively, named the *Encoder* (on the left) and the *Decoder* (on the right).

The input for the neural detector is a set of numerical features $\mathbf{x} = \{x_1, \ldots, x_N\}$ (in the analyzed use case, they are a number of statistics summarizing the behavior in a given time slot).

The encoder allows for producing a latent representation (encoding), defined by the function $\mathbf{z} = \text{enc}(\mathbf{x})$. The decoder generates the detector result $\mathbf{y} = \text{dec}(\mathbf{z})$ from the high-level features produced by the first subnetwork (decoding). Model weights are learned using gradient descent to minimize an appropriate loss function. Specifically, in our scenarios, we employ the Mean Squared Error (MSE) [21] as the loss function, defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{1}$$

The proposed neural model shows two major distinctions w.r.t a traditional AE:

- *Skip Connections*: These links enable direct information flow between layers in a neural network, bypassing intermediate layers and maintaining information and gradients intact [20]. Skip connections enable the creation of deeper NN while avoiding performance and learning issues. It is particularly advantageous as deeper NNs are capable of learning more complex patterns. In addition, they permit the DNNs to learn residual differences, boosting the predictive accuracy and reducing the number of iterations needed for algorithm convergence.
- *Hybrid architecture*: The usage of "*Sparse Dense Layers*" in the architecture enhances the autoencoder's ability to mitigate the effect of the noise noise, which is crucial given that anomalies often manifest as subtle deviations from normal behaviors. Basically, they consist of dense layers with a higher number of computational units compared to the size of the input. However, their "sparsity" is achieved by actively promoting sparse activations within the layer during the learning process, encouraging only a limited number of units to be activated, resulting in non-zero activations for a given input. The main objective of this choice is to simplify the representations learned by the network. Indeed, by using this approach, the Sparse Dense Layer learns a more straightforward and efficient representation of the input. This represents a crucial benefit in scenarios where data dimensionality reduction or feature selection is required. These special layers are placed as the first and last layers of the overall architecture. Both subnetworks are composed of M hidden layers; therefore, they exhibit a symmetrical architecture.

Essentially, skip connections streamline the training procedure by furnishing, as input to each level of the decoder ($D\text{-}DL_i$), excluding the latent level, both the prior level ($D\text{-}DL_{i-1}$) and the complementary level of the encoder

($E\text{-}DL_{M-i+1}$). Meanwhile, Sparse Layers are utilized to produce more discriminating features, promoting the yielding of a more comprehensive latent representation.

4.2 Detection workflow

Figure 3 outlines the detection procedure. We consider monitoring an "*infinite data stream*" which comprises data continuously generated by k IoT devices and fed into the detection system.

At determined intervals, named "time slot" in the diagram, a number of metrics are calculated to capture the state of the environment within each specific time window. This method provides different advantages, e.g., the capability to perform the detection mechanism without overwhelming network traffic and the utilization of minimal computational resources. This is attained by keeping only aggregated statistics during both the training and inference stages. The statistics are derived from sensor measurements and include metrics such as maximum, averages, etc.

So, extracted data are elaborated by our detection model, which undergoes a preprocessing phase to improve the approach's effectiveness. This stage comprises data normalization (in the range $[-1, 1]$) using a MinMax method. In addition, useless features (e.g., the ones with limited variability) are eliminated from the training set.

The core aspect of the entire mechanism involves utilizing a pre-trained AE on normal examples. This AE is utilized to replicate the computed metrics. In particular, MSE is exploited to measure the capability of the Net to replicate the input. Hence, the reconstruction error for the sample under analysis is computed as the MSE between input (x) and output (y). The key component of the detection system lies in this stage: the samples exhibiting a reconstruction error over a given threshold are recognized as anomalous, while the other ones are labeled as normal. The new normal data will be leveraged for the subsequent learning phase of the detection model, guaranteeing its reliability and accuracy in capturing the changing state of the monitored environment. By contrast, when an abnormal behavior is detected, the system activates an alert, signaling a deviation from the expected behavior. This strategy facilitates effective real-time monitoring and simultaneously offers the adaptability to respond to dynamic states.

5 Evaluation

In this section, we delve into an empirical analysis conducted on the Case Study presented in Section 3. In particular, first, *(i)* we define the parameter setting and the metrics used to assess the effectiveness of the devised detector, *(ii)* we discuss the accuracy results, and lastly, (iii) we analyze the efficiency of our model.

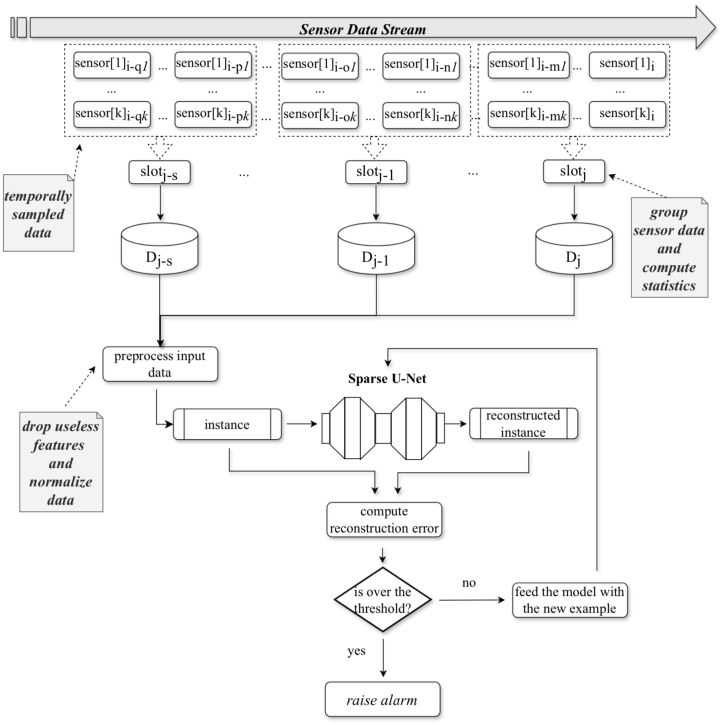

Fig. 3. Flow of the detection procedure.

5.1 Experimental setting

The neural framework described Section 4 is structured into multiple layers. The encoder's sparse layer includes 256 neurons. Subsequently, two MLPs with 24 and 16 neurons, respectively, are applied. Finally, the last layer of the encoder projects the input into a latent space with 8 dimensions. Reflecting the encoder's configuration, the decoder layer has a corresponding structure. A ReLU activation function is applied at each layer [17], with the exception of the output layer, which uses a linear activation function. For the training process, the *Adam* optimizer is employed [22]. The specific parameters used during training are outlined in Table 1.

To thoroughly assess our model, we conducted a comparison of accuracy metrics against those of a Deep Autoencoder (DAE) model. The primary distinctions between our architecture and the DAE are twofold: (*i*) the absence of skip connections in the DAE, and (*ii*) the substitution of the sparse layer with a linear layer that encompasses 28 neurons.

The proposed methodology hinges on a key parameter: the reconstruction error threshold. This is needed in order to be able to discriminate normal from abnormal behaviors. In particular, we devised three strategies for computing the aforesaid threshold that is based on the reconstruction errors calculated

Table 1. `Sparse U-Net` main parameters.

Model Parameters	Values
batch_size	16
num_epoch	32
optimizer	adam
loss	mse

Table 2. Experimental results using *peak anomalies*.

Model	Threshold	Acc	Prec	Rec	F1
`Deep Autoencoder (baseline)`	*98th percentile*	*0.913*	0.900	*0.374*	*0.529*
	max value	0.888	*1.000*	0.146	0.254
	max + tolerance	0.884	*1.000*	0.112	0.202
`Sparse U-Net (Our Detector)`	*98th percentile*	**0.948**	0.919	**0.663**	**0.771**
	max value	0.924	**1.000**	0.422	0.594
	max + tolerance	0.922	**1.000**	0.401	0.573

Table 3. Experimental results using *sensor fault anomalies*.

Model	Threshold	Acc	Prec	Rec	F1
`Deep Autoencoder (baseline)`	*98th percentile*	*0.943*	0.948	*0.580*	*0.720*
	max value	0.924	*1.000*	0.400	0.572
	max + tolerance	0.893	*1.000*	0.154	0.267
`Sparse U-Net (Our Detector)`	*98th percentile*	0.942	0.926	**0.591**	0.722
	max value	**0.946**	**1.000**	0.578	**0.733**
	max + tolerance	**0.946**	**1.000**	0.574	0.729

Table 4. Experimental results using *expert-induced anomalies*.

Model	Threshold	Acc	Prec	Rec	F1
`Deep Autoencoder (baseline)`	*98th percentile*	*0.990*	0.902	*0.925*	*0.914*
	max value	0.981	*1.000*	0.645	0.784
	max + tolerance	0.971	*1.000*	0.460	0.630
`Sparse U-Net (Our Detector)`	*98th percentile*	**0.990**	0.877	**0.960**	**0.916**
	max value	**0.990**	**1.000**	0.815	0.898
	max + tolerance	0.986	**1.000**	0.740	0.851

over the training set: *(i)* 98th percentile; *(ii)* maximum value identified; and *(iii)* combining (i) and (ii) using the following formula: $max + tolerance = max\_value + (max\_value - 98th\ percentile)$.

The detection capabilities of our approach have been measured by considering a number of well-known performance metrics: *Accuracy* (**Acc**), *F-score* (**F1**), *Precision* (**Prec**), and *Recall* (**Rec**) [25]. All the experiments have been performed using a machine equipped with an AMD Ryzen 7 5700U CPU @4.30GHz, 16 Gb RAM, and a 1TB SSD drive.

5.2 Quantitave results and sensitiveness analysis

Tables 2, 3, 4 show the results of the experimentation using different types of anomalies (synthetically injected as detailed in Section 3.1). Bold and italic values indicate the best results for the proposed and the baseline models, respectively. Notably, the `Sparse U-Net` outperforms the competitor model in all the proposed scenarios regarding Accuracy and F1. In particular, injecting *peak anomalies*, our model achieves up to +24.2%. Precision and Recall, in addition to Accuracy and F-Measure, give us some cues on the model's robustness. In particular, looking at the results in Table 2, we can notice that the baseline Deep Autoencoder obtains 90% and 37.4% regarding Precision and Recall, respectively. This behavior underlines how, in specific situations, i.e., scenarios with *peak anomalies*, the aforesaid model is unstable; on the contrary, the `Sparse U-Net` gains more than 30% in terms of Recall on the same scenario.

In addition to the analysis with the competitor, the sensitivity analysis of the threshold parameter highlights another significant aspect of the proposed architecture: the `Sparse U-Net` demonstrates greater robustness by achieving comparable results across the various settings under examination.

5.3 Convergence analysis

As discussed in Section 1, we designed our solution to be both effective and efficient. The latter aspect is relevant since the model will deployed on devices with limited resources and for energy-saving reasons. In this respect, we also conducted a convergence analysis to evaluate the speed with which the model is able to achieve a local optimum. In more detail, Figure 4 shows the loss function values during the learning phase for DAE and `Sparse U-Net`, respectively. Notably, the proposed model converges to a loss value that is 10^2 times lower than the baseline (two orders of magnitude) after 30 epochs, hence exhibiting better performances due to the architectural improvements discussed in Section 4.1. This result empirically confirms our previous statement showing the efficiency of the `Sparse U-Net`.

Lastly, we analyzed the times for the learning and inference stages. As regards the former phase, the whole dataset is processed in ~ 6 seconds (*time per epoch*) while only 5 ms are required for a single batch. In the inference step, less than 1 ms is needed to produce the output for a single instance. These results confirm the efficiency of the proposed approach and its suitability for deployment on small devices (lightweight).

6 Conclusions and Future Works

Early detection of faults in IoT-based sensor networks is crucial for maintaining the reliability and efficiency of smart workplaces and buildings. In this work, we presented a Deep Learning-based framework for detecting anomalies in sensor networks within smart workplaces and buildings. Our framework leverages an

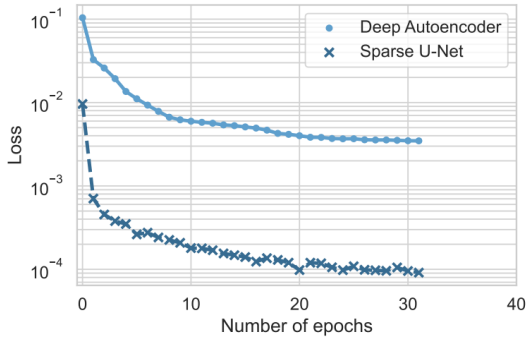

Fig. 4. Loss function trends during training

unsupervised autoencoder architecture to identify deviations from normal behavior, thereby ensuring timely detection of faults and enhancing the safety and operational efficiency of the monitored environments. Extensive experimentation on real-world data demonstrates the effectiveness and efficiency of our approach in maintaining the reliability of IoT systems.

Although the achieved results are promising, several issues are still open. One major issue is the normal evolution in data distributions due to expected behaviors e.g., temperature increases during the season transitions, which can undermine the model's predictive capability (Concept Drift). We are interested in investigating the use of ensemble approaches to enhance the model's robustness against both abrupt and gradual normal changes [6]. Additionally, we aim to explore federated learning to share knowledge among different subnets, thereby making the model more resilient to a broader variety of changes.

Acknowledgments. This work has been partially supported by: (i) the Italian MUR, PRIN 2022 Project "iSafety: Leveraging artificial intelligence techniques to improve occupational and process safety in the iron and steel industry", Prot.: 20222YRP2F, ERC field: PE8, CUP: D53D23003010006; (ii) European Union - NextGenerationEU - National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) - Project: "SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics" - Prot. IR0000013 - Avviso n. 3264 del 28/12/2021; (iii) the Italian MUR, PRIN 2022 Project "COCOWEARS" (A framework for COntinuum COmputing WEARable Systems), Prot. 2022T2XNJE, CUP: B53D23013190006; (iv) MUR on D.M. 351/2022, PNRR Ricerca, CUP H23C22000550005. (v) the Italian Ministry of University and Research (MUR) within the PRIN 2022 program and European Union - Next Generation EU - Project "INSIDER: INtelligent ServIce Deployment for advanced cloud-Edge integRation", grant n. 2022WWSCRR, CUP H53D23003670006)

References

1. Arora, S., Kumar, S., Kumar, S.: Artificial intelligence in disaster management: A survey. In: Saraswat, M., Chowdhury, C., Kumar Mandal, C., Gandomi, A.H.

(eds.) Proceedings of International Conference on Data Science and Applications. pp. 793–805. Springer Nature Singapore, Singapore (2023)
2. Bengio, Y., Pascal, L., Dan, P., Larochelle, H.: Greedy layer-wise training of deep networks. In: Adv. Neur. Inf. Proc. Sys. (NeurIPS), vol. 19, pp. 153–160. MIT Press (2007)
3. Cassavia, N., Caviglione, L., Guarascio, M., Liguori, A., Zuppelli, M.: Learning autoencoder ensembles for detecting malware hidden communications in iot ecosystems. Journal of Intelligent Information Systems (2023). https://doi.org/10.1007/s10844-023-00819-8
4. Cicero, S., Guarascio, M., Guerrieri, A., Mungari, S.: A deep anomaly detection system for iot-based smart buildings. Sensors **23**(23), 9331 (2023)
5. Costa, G., Forestiero, A., Ortale, R.: Rule-based detection of anomalous patterns in device behavior for explainable iot security. IEEE Transactions on Services Computing **16**(6), 4514 – 4525 (2023). https://doi.org/10.1109/TSC.2023.3327822
6. Folino, G., Guarascio, M., Papuzzo, G.: Exploiting fractal dimension and a distributed evolutionary approach to classify data streams with concept drifts. Appl. Soft Comput. **75**, 284–297 (2019). https://doi.org/10.1016/j.asoc.2018.11.009, `https://doi.org/10.1016/j.asoc.2018.11.009`
7. Gao, H., Qiu, B., Barroso, R.J.D., Hussain, W., Xu, Y., Wang, X.: TSMAE: A novel anomaly detection approach for internet of things time series data using memory-augmented autoencoder. IEEE Trans. Netw. Sci. Eng. **10** (2023). https://doi.org/10.1109/TNSE.2022.3163144
8. Guarascio, M., Manco, G., Ritacco, E.: Deep learning. In: Encyclopedia of Bioinformatics and Computational Biology - Volume 1, pp. 634–647. Elsevier (2019). https://doi.org/10.1016/b978-0-12-809633-8.20352-x, `https://doi.org/10.1016/b978-0-12-809633-8.20352-x`
9. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504 – 507 (2006)
10. Jan, S.U., Lee, Y.D., Koo, I.S.: A distributed sensor-fault detection and diagnosis framework using machine learning. Information Sciences **547**, 777–796 (2021). https://doi.org/https://doi.org/10.1016/j.ins.2020.08.068
11. Khan, I., Cicirelli, F., Greco, E., Guerrieri, A., Mastroianni, C., Scarcello, L., Spezzano, G., Vinci, A.: Leveraging distributed ai for multi-occupancy prediction in cognitive buildings. Internet of Things **26**, 101181 (2024). https://doi.org/https://doi.org/10.1016/j.iot.2024.101181
12. Khan, W.Z., Ahmed, E., Hakak, S., Yaqoob, I., Ahmed, A.: Edge computing: A survey. Future Generation Computer Systems **97**, 219–235 (2019). https://doi.org/https://doi.org/10.1016/j.future.2019.02.050
13. Le Cun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
14. Li, D., Chen, D., Jin, B., Shi, L., Goh, J., Ng, S.: 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 - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part IV. Springer. https://doi.org/10.1007/978-3-030-30490-4_56
15. Li, S., Cheng, Y., Liu, Y., Wang, W., Chen, T.: Abnormal client behavior detection in federated learning. CoRR **abs/1910.09933** (2019)
16. Lydia, E.L., Jovith, A.A., Devaraj, A.F.S., Seo, C., Joshi, G.P.: Green energy efficient routing with deep learning based anomaly detection for internet of things (iot) communications. Mathematics **9**(5) (2021). https://doi.org/10.3390/math9050500

17. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. p. 807–814. ICML'10, Omnipress, Madison, WI, USA (2010)
18. Oh, M., Iyengar, G.: Sequential anomaly detection using inverse reinforcement learning. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019. pp. 1480–1490. https://doi.org/10.1145/3292500.3330932
19. Patel, N., Saridena, A.N., Choromanska, A., Krishnamurthy, P., Khorrami, F.: Adversarial learning-based on-line anomaly monitoring for assured autonomy. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018. https://doi.org/10.1109/IROS.2018.8593375
20. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing, Cham (2015)
21. Rosasco, L.and De Vito, E.D., Caponnetto, A., Piana, M., Verri, A.: Are loss functions all the same? Neural Computation **15**(5), 1063–1076 (2004)
22. Ruder, S.: An overview of gradient descent optimization algorithms (2017)
23. Sater, R.A., Hamza, A.B.: A federated learning approach to anomaly detection in smart buildings. ACM Trans. Internet Things **2**(4) (2021). https://doi.org/10.1145/3467981
24. Servin, A., Kudenko, D.: Multi-agent reinforcement learning for intrusion detection: A case study and evaluation. In: ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings. IOS Press. https://doi.org/10.3233/978-1-58603-891-5-873
25. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing & Management **45**(4), 427–437 (2009). https://doi.org/https://doi.org/10.1016/j.ipm.2009.03.002
26. Syed, A.S., Sierra-Sosa, D., Kumar, A., Elmaghraby, A.: Iot in smart cities: A survey of technologies, practices and challenges. Smart Cities **4**(2), 429–475 (2021). https://doi.org/10.3390/smartcities4020024
27. Weston, J., Chopra, S., Bordes, A.: Memory networks. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
28. van Wyk, F., Wang, Y., Khojandi, A., Masoud, N.: Real-time sensor anomaly detection and identification in automated vehicles. IEEE Transactions on Intelligent Transportation Systems **21**(3), 1264–1276 (2020). https://doi.org/10.1109/TITS.2019.2906038
29. Yahyaoui, A., Abdellatif, T., Yangui, S., Attia, R.: Read-iot: Reliable event and anomaly detection framework for the internet of things. IEEE Access **9** (2021). https://doi.org/10.1109/ACCESS.2021.3056149
30. Zhu, K., Chen, Z., Peng, Y., Zhang, L.: Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using lstm. IEEE Transactions on Vehicular Technology **68**(5), 4275–4284 (2019). https://doi.org/10.1109/TVT.2019.2907269