Machine Learning (2024) 113:4445–4486
https://doi.org/10.1007/s10994-023-06501-y

Distributed and explainable GHSOM for anomaly detection in sensor networks

Paolo Mignone[1,2] · **Roberto Corizzo**[1,3] · **Michelangelo Ceci**[1,2,4]

Received: 8 March 2023 / Revised: 20 October 2023 / Accepted: 16 December 2023 /
Published online: 22 January 2024

Abstract
The identification of anomalous activities is a challenging and crucially important task in sensor networks. This task is becoming increasingly complex with the increasing volume of data generated in real-world domains, and greatly benefits from the use of predictive models to identify anomalies in real time. A key use case for this task is the identification of misbehavior that may be caused by involuntary faults or deliberate actions. However, currently adopted anomaly detection methods are often affected by limitations such as the inability to analyze large-scale data, a reduced effectiveness when data presents multiple densities, a strong dependence on user-defined threshold configurations, and a lack of explainability in the extracted predictions. In this paper, we propose a distributed deep learning method that extends growing hierarchical self-organizing maps, originally designed for clustering tasks, to address anomaly detection tasks. The SOM-based modeling capabilities of the method enable the analysis of data with multiple densities, by exploiting multiple SOMs organized as a hierarchy. Our map-reduce implementation under Apache Spark allows the method to process and analyze large-scale sensor network data. An automatic threshold-tuning strategy reduces user efforts and increases the robustness of the method with respect to noisy instances. Moreover, an explainability component resorting to instance-based feature ranking emphasizes the most salient features influencing the decisions of the anomaly detection model, supporting users in their understanding of raised alerts. Experiments are conducted on five real-world sensor network datasets, including wind and photovoltaic energy production, vehicular traffic, and pedestrian flows. Our results show that the proposed method outperforms state-of-the-art anomaly detection competitors. Furthermore, a scalability analysis reveals that the method is able to scale linearly as the data volume presented increases, leveraging multiple worker nodes in a distributed computing setting. Qualitative analyses on the level of anomalous pollen in the air further emphasize the effectiveness of our proposed method, and its potential in determining the level of danger in raised alerts.

Keywords Anomaly detection · Self-organizing maps · Distributed learning · Sensor networks · Explainable AI

Editors: Dino Ienco, Roberto Interdonato, Pascal Poncelet.

Extended author information available on the last page of the article

1 Introduction

Anomaly detection is a challenging and important task in a variety of real-world domains. Its goal is to identify observations that are partially or entirely irrelevant as they are not generated by an assumed and unknown stochastic model (Chandola et al., 2007). For tasks, several statistical approaches have been originally proposed, both parametric (Urvoy & Autrusseau, 2014; Rousseeuw & Leroy, 2005; McCallum et al., 2000; Eskin et al., 2002; Duda et al., 2006; Jordan & Jacobs, 1994) and non-parametric (Hofmeyr et al., 1998; Javitz et al., 1991; Desforges et al., 1998). A different class of approaches is that of information theory-based approaches, which analyze the information content of a data set using different information-theoretic measures such as entropy, relative entropy etc. (Lee & Xiang, 2001; Arning et al., 1996; Li & Vitányi, 1993; He et al., 2005; Noble & Cook, 2003). Another perspective is that of machine learning-based approaches for anomaly detection, which aim to identify patterns or observations in data that deviate from the expected behavior. Approaches involve training a model based on historical data, which can then be used to identify instances that do not conform to the observed patterns. Anomaly detection has a wide range of applications in various fields, including fault detection in system diagnostics, credit card fraud, and intrusion detection in cybersecurity (Hale et al., 2019; Lebichot et al., 2021; Aldweesh et al., 2020).

In urban areas, anomalies can be identified in sensor network data. This is the case of anomalous traffic jams or pedestrian flows, which could be promptly detected so as to prevent potential security threats. The detected anomalies could be timely transmitted to security operators who could, in turn, make informed decisions for the situation at hand. Anomaly detection for pedestrian flow can be employed, for instance, to promptly detect large crowds in unfitting areas (possibly due to unauthorized protests). Another useful application in this context is the detection of environmental anomalies, such as abnormal air pollutants, which can potentially compromise citizens' health (Zhang et al., 2022). Anomaly detection is also relevant in the context of smart grids and power systems, which provide a wide range of relevant application scenarios mainly indicating overloads, malfunctions, etc.

One common approach for anomaly detection in sensor network data is to use supervised or unsupervised machine learning techniques. The goal is to identify an anomalous, not expected, behaviour for one or many values from different sensors simultaneously, considering the specific temporal and spatial coordinates of the considered observations so as to take into account their spatio-temporal autocorrelation (Kou et al., 2006; Shekhar et al., 2001; Corizzo et al., 2021).

Although supervised and semi-supervised machine learning methods are often adopted for anomaly detection, one of the most challenging issue is determined by the limited availability of labeled anomaly data. Indeed, the task of manually labeling the anomalous events requires: i) the availability of human experts; ii) the ability to identify real events that could represent anomalous scenarios in the ground truth; iii) a consistent labeling effort by sequentially reviewing large-scale historical data, which is a time-consuming and error-prone task; iv) labels can be affected by the so-called "contamination" problem, that is, can be inaccurate up to a certain percentage. These motivations usually lead researchers to prefer unsupervised machine learning methods, which are able to work with unlabeled data, avoiding the effort of manually labeling normal and anomalous events.

A good compromise between supervised and unsupervised approaches is that of semi-supervised methods that require a limited number of labels. They can also successfully

deal with the contamination problem (Wang et al., 2022). However, a proper estimation of the contamination rate is required to yield a satisfactory model performance. An inaccurate estimation will otherwise negatively result in an inaccurate decision boundary for the models. For all these reasons, our study proposes an unsupervised approach to anomaly detection.

Additional challenges arise with large-scale high-dimensional data, which is commonly present in sensor networks (Thudumu et al., 2020). Big data techniques such as parallel and distributed computing, as well as data stream monitoring and processing tools, should be used to efficiently process and analyze such data in a timely manner. These techniques can be used to allow for early detection of anomalies (Reddy Shabad et al., 2021).

The most popular category of approaches for machine-learning based anomaly detection tasks in the literature is that of one-class learning methods. Popular methods include One-Class Support Vector Machines (OCSVM) (Schölkopf et al., 2000), Isolation Forest (Liu et al., 2008), Auto-Encoders (Najafabadi et al., 2015; Sakurada & Yairi, 2014; Zhou & Paffenroth, 2017; Chong & Tay, 2017), Angle-Based Outlier Detection (ABOD) (Kriegel et al., 2008; Pham & Pagh, 2012; Jahromi et al., 2022), and COPula-based Outlier Detection (COPOD) (Li et al., 2020). A less popular but intriguing alternative is provided by Self-Organizing Maps (SOM). A SOM (Kohonen, 1990) is a neural-network-based model for prototype-based clustering, which works by mapping high-dimensional input data into a 2-dimensional space implemented as a grid of neurons called *feature map*. The main difference with respect to the classical neural networks is the fact that in SOMs the final model is represented by the feature maps, instead of a matrix of weights. SOM approaches present the advantage of supporting fully unsupervised model training, as well as the ability to analyze data presenting multiple densities, and visualize it through the learned feature map representations (Qu et al., 2021). However, both one-class learning methods and SOM-based approaches present a number of limitations that constrain their effectiveness in the complex real-world anomaly detection setting described above. First, they are usually not designed for handling large-scale data generated by sensor networks. Second, once the predictive models are trained, a complex threshold configuration for the final predictive function is required and usually depends on a user-defined setting that is difficult to estimate and it is subject to change over time. Third, they do not focus on the explainability of the detected anomalies, assuming that the domain experts will be able to understand the raised alerts and the underlying motivations that triggered them. Fourth, despite their broad applicability in many domains, they are scarcely adopted in real-world domains involving sensor network data, such as smart grids and in urban public safety applications.

Although a number of research works have separately addressed such issues, there is a substantial lack of methods that holistically combine effective model capabilities to address the needs of sensor network data analysis. In this paper, we fill this gap by proposing an anomaly detection method that jointly addresses these issues.

Our SOM-based anomaly detection method exploits the attractive properties of SOMs described above and adapts them to fit the context of sensor network data. Specifically, we consider GHSOMs (Growing Hierarchical SOMs), which are particularly suitable to describe the space of normal instances around the neurons and they are sensitive to data belonging to low-density spaces. When a single SOM is not adequate enough to describe the normal data, the algorithm allows the model to be further extended (with additional rows, columns, or entire SOMs) to further fit the data distribution under analysis. However, since GHSOMs are mostly limited to clustering and data visualization, we extend the Spark-GHSOM (Malondkar et al., 2018) algorithm to learn GHSOMs capable of solving anomaly detection tasks. To support the analysis of large-scale data arising in our domains

of interest, we devise distributed algorithms based on the map-reduce programming paradigm, which allow us to perform model training efficiently using multiple computational nodes. After the training stage, our anomaly detection component leverages the learned model to identify anomalies in unseen data points based on their distance relationship with respect to the learned normal data distribution. To overcome the limitation of manually set threshold configurations, we devise an automated threshold tuning strategy to improve the robustness of the method to noisy instances (or anomalies) within the training set. Finally, to support practitioners and end users in better understanding model predictions, we propose an explainability component that leverages feature rankings to reveal which features determined the predicted outcome, and should be given higher attention.

Our extensive experiments and comparative analysis with real-world datasets highlight the merits of our proposed method from both a quantitative and qualitative viewpoint. Specifically, experiments were conducted for different sensor network data applications, i.e., *renewable energy analysis* and *urban physical threat detection* such as vehicle traffic monitoring, pedestrian flow analysis, and monitoring of anomalous pollen level in the air, which are less commonly explored in anomaly detection literature despite their real-world relevance. Furthermore, we analyzed also time-series for the metrics of different *Yahoo! web services*.

The remainder of the paper is structured as follows. Section 2 surveys relevant related works. Section 3 describes our proposed method in terms of its different components. Section 4 discusses the analyzed data, the experimental settings, and discusses the results obtained by our study. Finally, Sect. 5 concludes the paper and provides relevant directions for future work.

2 Background

2.1 Statistical and information-theoretic methods

Statistical approaches consider as outliers those observations that are partially or wholly irrelevant as they are not generated by an assumed stochastic model (Anscombe, 1960). These approaches require a training phase for the statistical model estimation (estimating the distribution parameters) and a testing phase where a test instance is compared to the model to determine if it is an outlier or not. Parametric approaches leverage statistical tests, such as the Grubb's test (Urvoy & Autrusseau, 2014) that assumes a normal data distribution. Parametric approaches also include regression techniques, which fit a regression model on the data (Rousseeuw & Leroy, 2005). A number of techniques assume a Markovian nature of the data when modeling sequential data (McCallum et al., 2000). Since real world scenarios are often characterized by different distributions, some techniques take this aspect into consideration with mixtures of probability distributions (Eskin et al., 2002). Parameter estimation techniques can also be used to estimate the parameters for each of the above cases (Duda et al., 2006; Jordan & Jacobs, 1994). Non-parametric approaches do not assume any knowledge of the data distribution. One of the most widely used techniques is histogram analysis (Hofmeyr et al., 1998), only efficient with univariate data. Some approaches (Javitz et al., 1991) compute histograms for each feature separately and detect outliers independently in each dimension. A popular non-parametric approach is to estimate the probability density function using Parzen windows (Desforges et al., 1998).

From the information theory viewpoint, techniques involve different information theoretic measures such as entropy, information gain, information cost, typically in an unsupervised fashion. (Lee & Xiang, 2001) exploited information measures to detect outliers in a sequence of operating system calls. The main aim is to find the regularity of a data set in order to detect outliers while inducing irregularities in the data. Arning et al. (1996) proposed the overall dissimilarity by exploiting the Kolmogorov data complexity (Li & Vitányi, 1993). He et al. (2005) leverage the Local Search Algorithm (LSA) to reduce the dataset, reducing the entropy of the remaining data set. In Noble and Cook (2003), the authors discovered patterns in networked data with the aim to detect changes over time in terms of the previously mentioned information measures. A change in the encoding of a pattern indicates a meaningful change in the data.

Statistical and information-theoretic approaches have strong theoretical foundations and have shown robustness in different applications. However, they present limited effectiveness in the presence of complex sensor network data characterized by time-evolving, multi-density and large-scale data.

2.2 Anomaly detection with learned models

Anomalous data instances could be classified as point, collective, and contextual (Chandola et al., 2009). In this work, we mainly focus on detecting contextual anomalies, where the context is described through the spatial and temporal dimensions of data (Kou et al., 2006; Shekhar et al., 2001). An example could be represented by a sudden change in car traffic level during the weekend in a suburban area of the city.

One-Class Support Vector Machines (OCSVM) (Schölkopf et al., 2000) is a classical method that learns a separating hyperplane in a high-dimensional space. After the training stage, the hyperplane model produced by OCSVM can classify a new data observation as regular/normal or different/anomaly with respect to the training data distribution, according to its geometrical position within the decision boundary.

Isolation Forest (Liu et al., 2008), exploits an ensemble of tree-based models, and computes an isolation score for each data observation. Such a score is calculated as the average path length from the root of the tree to the node containing the single observation. The shorter the path, the easier is to isolate an observation from the others due to significant differences in values w.r.t. training data instances.

Methods based on auto-encoders and stacked auto-encoders (Najafabadi et al., 2015; Sakurada & Yairi, 2014; Zhou & Paffenroth, 2017; Chong & Tay, 2017) have demonstrated superior performance constructing representations with a low reconstruction error through non-linear combinations of input features (Bengio, 2009). Moreover, the rise in popularity of deep learning led to modern approaches for neural network model training, including Stochastic Gradient Descent (SGD), mini-batch training, and adaptive gradients, which resulted in an increased efficiency for autoencoder-based anomaly detection approaches, among other neural network architectures (Kashyap, 2022; Kingma & Ba, 2015; Draxler et al., 2018). On the other hand, reconstruction-based approaches based on auto-encoders may present a reduced accuracy when anomalies are caused by a few of the available features, which result in minimal variations of reconstruction scores compared to benign data. The same consideration could be made in the context of multi-density data, where the exploitation of a single model may be insufficient to catch multiple distributions.

Angle-Base Outlier Detection (ABOD) (Kriegel et al., 2008; Pham & Pagh, 2012) is another popular approach that computes the variance of weighted cosine scores between

each data point and its neighbors and leverages it as the anomaly score. Its key advantages are to efficiently identify outliers in high-dimensional spaces, as well as consider relationships between each point and its neighbors instead of relationships among neighbors, which has the potential to reduce the number of false positive detections.

Another recent state-of-the-art anomaly detection method is COPula-based Outlier Detection (COPOD) (Li et al., 2020), which models data by constructing an empirical copula, i.e., a multi-variate cumulative distribution function with a uniform marginal probability distribution for each feature in the [0, 1] interval. During detection, the method leverages the copula to predict the tail probabilities of each data sample to determine the degree of its "extremeness".

Despite the diversity, robustness, and high performance showcased by these methods in a number of domains, they present one or more limitations, i.e. the inability to deal with large-scale data, the lack of explainability for their predicted anomalies, the dependence on users for threshold settings, and the inability to analyze data presenting multiple densities.

Similarly to auto-encoder neural network models, SOM-based approaches can be trained in a fully unsupervised manner using background data, without explicit use of labels. Another similarity is that, unlike more complex neural network approaches, they both adopt a simple cost function: while auto-encoders adopt reconstruction error, SOM-based approaches leverage the quantization error. However, GHSOM approaches present a structural advantage over auto-encoder models, since they present a tree-like multi-level expandable model structure that support the analysis of data at multiple levels of density granularity. The authors in Muñoz and Muruzábal (1998) leverage the interneuron distance matrix and the projection of the trained map via Sammon's mapping to detect outliers in artificial and real data. The work in Palomo et al. (2010) proposes a hierarchical SOM with a particular focus on the reduction of the number of user-dependent parameters. The authors showcase its effectiveness in detecting anomalies with the KDD Cup 1999 intrusion detection dataset. Another SOM-based work on the same application domain is proposed in Ippoliti and Zhou (2012), where the authors propose four enhancements: threshold-based training, dynamic input normalization, feedback-based quantization, and prediction confidence filtering. Common limitations of these works include the inability to deal with large-scale data, and a lack of model explainability capabilities. In this work, we aim to simultaneously address these challenges by proposing our distributed explainable SOM-based approach for anomaly detection.

2.3 Anomaly detection from sensor networks

Among the many application domains that analyze data generated by sensor networks in order to identify anomalies, below we examine some works specifically related to the application domains considered in this study, namely "anomaly detection in electrical networks", "detection of anomalies from urban data" and "detection of anomalies in the use of web services".

2.3.1 Anomaly detection in electrical networks

The digitization of the energy infrastructure is a process that offers benefits both for consumers and service providers. Thanks to the first advent of the industrial control systems, and currently the industrial internet of things, it is possible to allow a growing level of control and supervision (Borges Hink et al., 2014; Pan et al., 2015a, 2015b, 2015c; Shin

et al., 2020). One of the most interesting results is the growing availability of applications for monitoring the huge amount of data generated in real time and promptly identifying any malfunctions, thefts or improper overloads. This has motivated a growing interest in anomaly detection from data generated in power grids (e.g. at power plants, distribution stations, etc.) (Himeur et al., 2020; Su et al., 2023).

In Himeur et al. (2020), the authors examined thirty-one databases with different features, such as the period of collection, geographical location, sampling rate of collected data, number of monitored households and so forth. Among the different tasks analyzed, the authors also analyzed and discussed energy consumption dataset for the task of anomaly detection for reducing wasted energy. Similarly, in Su et al. (2023), the authors collected a dataset from real-world industrial solar-cell production lines. The dataset was analyzed by learning predictive models for the anomaly detection task and performed a comparative analysis with four state-of-the-art methods. In De Benedetti et al. (2018), the authors analysed energy production data streams from photovoltaic systems and compared the output of a trained model in order to analyse the vectors of residuals which are aggregated over 1-day and analyzed to detect a potential system degradation. Triangular Moving Average (TMA) is considered in analyses to automatically determine the window size. The proposed anomaly detection method was used to generate daily maintenance alerts. In Reddy Shabad et al. (2021), the authors analyzed smart grid power systems faults focusing on discriminating among normal condition, natural disturbances, or cyberattacks. In Malki et al. (2022), the authors integrated anomaly detection methods to improve the maintenance of power systems and control as a fundamental part of the smart city concept. The authors in Takiddin et al. (2022) detected electricity theft in smart grids through a deep auto-encoder for anomaly detection.

Although these works are specifically tailored for the analysis of energy data, they present one or more of the following limitations: *i)* Limited support in providing predictive models capable of explaining the reasons for which anomalies have been detected. *ii)* Inability to deal with large-scale data. This issue is typically tackled via sampling approaches, which limit their generalization capabilities to a narrow selection of data points. *iii)* Inability to take into account multi-sourced spatio-temporal information (Corizzo et al., 2021) such as ambient conditions, outside weather footprints, as well as energy plant characteristics such as voltages (Ceci et al., 2020), which would greatly increase the scope and the robustness of anomaly detection models (Himeur et al., 2021).

2.3.2 Urban anomaly detection

Anomaly detection tasks are particularly useful in urban areas, where data is continuously generated by geo-located sensors. Detecting anomalies provides security operators with the opportunity to understand potentially anomalous situations and take the appropriate actions in a timely manner.

Data refers to physical information (e.g. temperature, number of vehicles crossing an intersection, number of pedestrians in a given area, PM10 level at certain physical positions in the city, etc.). The authors in Zhang et al. (2022) proposed a relevant survey on urban anomaly detection. Specifically, they discussed different types of anomalies in the analyzed contexts, i.e., urban anomalies, traffic anomalies, unexpected crowds, environment anomalies, and individual anomalies. Furthermore, the authors emphasized that one of the open challenges that undermines the detection accuracy is posed by noisy urban data.

From a methodological viewpoint, SOM-based approaches have shown promise in this domain. In Riveiro et al. (2017), the authors proposed a framework that provides decision support for the exploration of multidimensional road traffic data via visual artifacts. The method for anomaly detection is based on a classical SOM used for clustering. The number of clusters is optimized via Silhouette cluster analysis. This approach is inspired by a previous work presented in Kraiman et al. (2002) that used a classical approach for anomaly detection based on clustering algorithms, SOMs, and Gaussian Mixture Models (GMMs). However, such methods do not present growing capabilities, i.e. the number of neurons in the model is defined at the beginning and does not increase, which results in a significant modeling power reduction in dynamic data scenarios requiring adaptation.

Detecting anomalies with high accuracy in the urban context requires the simultaneous analysis of sensor data at multiple locations, exploiting the temporal and spatial coordinates of the considered observations (Corizzo et al., 2021; Mignone et al., 2022; Sofuoglu & Aviyente, 2022). In Zhang et al. (2019), the authors proposed a decomposition approach to detect urban anomalies of different types, such as abnormal pedestrian flows and traffic accidents with varying locations and times. Specifically, they distinguish between the normal component, i.e., urban dynamics decided by spatio-temporal features, and the abnormal component that is caused by anomalous events.

Overall, anomaly detection with urban data is a challenging task, since data generated by sensors is inherently large-scale, and the spatial proximity of sensors introduces spatial autocorrelation, which is in contrast with the typical assumption that observations are independently and identically distributed (Stojanova et al., 2012). As a result, anomalies such as traffic congestion or large crowds can be difficult to detect due to their rarity, and the fact that their definition varies based on spatial and temporal data characteristics. In other words, a common limitation of anomaly detection methods for urban data is the inability to consider the relationships between different points in space and time and specific domain-dependent characteristics of the anomalies (Sofuoglu & Aviyente, 2022). In addition, methods are often unable to deal with the additional challenges presented by large-scale data and model explainability.

2.3.3 Anomaly detection in the use of web services

Web applications generate real-time web content from online activities including Internet banking, email, social networking, and search engines. Such web services could be encoded as data streams for monitoring purposes. Anomaly detection tasks in Key Performance Indicators (KPIs) in web applications, i.e., number of orders, service response time, CPU utilization, network throughput, and page view counts, received attention in several applications to protect web services from system failures proactively (Tama et al., 2020). These applications are enabled by sensor networks allowing the collection of real-time data from different sources, such as physical sensors, mobile devices or other data sources. Specifically, a bulk of data is mainly generated and trasmitted by wireless sensor networks (Duan et al., 2019) that can be used to measure and evaluate the performance of web services through KPIs.

In Zhang et al. (2022), the authors proposed an unsupervised method, based on a variational auto-encoder, for learning predictive models that could be used to timely detect anomalies of KPI indicators. The method consists of a module for offline pre-training of *shape models*, through clustering, to select only the closest shape models, in terms of centroid closeness, to the online stream in an adaptive transfer learning strategy. In a similar

manner, transfer learning was considered also in Duan et al. (2019) for anomaly detection tasks in web services. It clusters historical KPIs and then it considers all KPIs in each cluster as input into a shared-hidden-layers variational auto-encoder model. Variational auto-encoders are exploited effectively also in Xu et al. (2018), where the authors analyzed the noise distribution in KPI anomaly detection problems.

In Hagemann and Katsarou (2020), the authors evaluated PCA, auto-encoders and long short-term memory encoder-decoders for anomaly detection in cloud-specific metrics of various Yahoo! services. For the Yahoo! Webscope S5 dataset [71] the results showed that PCA is the most robust and fastest way to detect anomalies, while neural networks without regularization tend to overfit the data.

While these works are clearly effective for anomaly detection in the use of web services, they do not have growing capabilities, do not provide explanations associated with the detected anomalies and do not provide scalable solutions to deal with the challenges presented by large-scale data.

3 Proposed method

In this section we describe in detail our proposed distributed and explainable anomaly detection method. First, we provide an overview of SOM-based model training. Subsequently, three subsections focus on the training process in detail, how threshold auto-tuning is performed, and our approach to enrich model predictions with explanations.

A SOM layer contains a two-dimensional grid of neurons, where each neuron is described by a weight vector. Conceptually, the training stage incrementally presents instances[1] to the model and takes place for a given number of iterations, i.e. *epochs*. During this process, the neuron with the shortest distance for a given instance (also known as the *winner neuron*) is identified, and its neighborhood is adapted to get closer to the instance.

For this purpose, the SOM and GHSOM algorithms leverage a metric called Mean Quantization Error (*MQE*) (Dittenbach et al., 2000). The *MQE* of a neuron is calculated as the total deviation of such neuron from the input instances mapped to it. Another important concept is the *MQE* associated to the entire SOM layer, which is calculated as the average *MQE* of all its constituent neurons.

The first step for model training is to compute the *MQE* of the level-0 neuron with respect to all input instances, denoted as mqe_0.

Subsequently, a first neuron map of 2×2 neurons is created at level-1. This map is trained according to the conventional SOM training process. Once the training process is complete, the map is analyzed and the *MQE* for the map m denoted as MQE_m is calculated. High values of MQE_m indicate that the map m does not accurately represent the input data and, therefore, may require more neurons to reach this goal, which is formalized by the τ_1 training criterion in Eq. (1):

$$MQE_m < \tau_1 \times mqe_p, \tag{1}$$

where mqe_p represents the *MQE* of the parent neuron which is responsible for the expansion of the map m, while τ_1 is a weight value governing the sensitiveness of the single

[1] In this paper, the terms instance, data point, input vector are used interchangeably.

SOM expansion, i.e., the higher the τ_1, the smaller the SOM, which results in a faster model training phase.

The map growing process has the goal to reach the condition in Eq. (1). Specifically, to realize the map growing process, the neuron presenting the highest *MQE* is identified and denoted as the *error neuron e*. Subsequently, its most dissimilar direct neighbouring neuron *d* is selected, and a new row or column of neurons is added into the grid between *e* and *d*.

Each newly inserted neuron is a vector initialized considering the average of the weight vectors at its corresponding adjacent neighbours. By doing so, the process yields an updated (grown) layer, which is evaluated and trained once again. This dual process characterized by growth and training continues until the τ_1 criterion is met. When the τ_1 criterion is satisfied, each neuron in the map is then analyzed according to the criterion in Eq. (2) (τ_2 criterion):

$$mqe_k < \tau_2 \times mqe_0, \tag{2}$$

where mqe_k represents the *MQE* of the k^{th} neuron under analysis, and τ_2 is a weight value governing the sensitiveness of the hierarchy expansion, i.e., the higher the τ_2, the smaller the hierarchy, which results in a faster model training phase.

The neurons which do not satisfy the τ_2 criterion are expanded into new maps at the next level of hierarchy. These new maps undergo the same process of training, growth and hierarchical expansion as the level-1 map.

The training of the GHSOM model stops when all the neurons in the maps at the last level of the hierarchy satisfy Eq. (2). The resulting GHSOM structure thus contains multiple SOM layers arranged as a hierarchy, with each SOM representing the data at a finer granularity than its parent layer.

3.1 Detailed training process

The first step of the GHSOM algorithm consists in computing the global value of dissimilarity in the input dataset denoted by mqe_0 (i.e. the Mean Quantization Error of the level-0 neuron). First, the mean of all input vectors m_0 in the dataset is computed. Second, to obtain the value of mqe_0, the mean distance of the input vectors from the mean vector is computed as follows:

$$mqe_0 = \frac{1}{n} \sum_{x_{(\cdot,i)} \in C_n} \|m_0 - x_{(\cdot,i)}\|, \tag{3}$$

where C_n denotes the set of n input instances, $x_{(\cdot,i)}$ denotes the vector representing the i-th instance currently presented to the model, and mqe_0 denotes the overall dissimilarity of the input dataset. In our work, we replace mqe_0 with the classical variance, denoted as var_0, as a measure of deviation that is more robust to outliers.

Subsequently, in the training process, a 2×2 SOM layer is created. If this is the layer at level 1, the neuron weight vectors are initialized at random. For each subsequent level, SOM layers can be initialized as a function of their parent neuron and their neighbours. To realize this goal, we adopt the approach devised in Chan and Pampalk (2002). Figure 1 graphically describes the training of a single SOM starting from level 1.

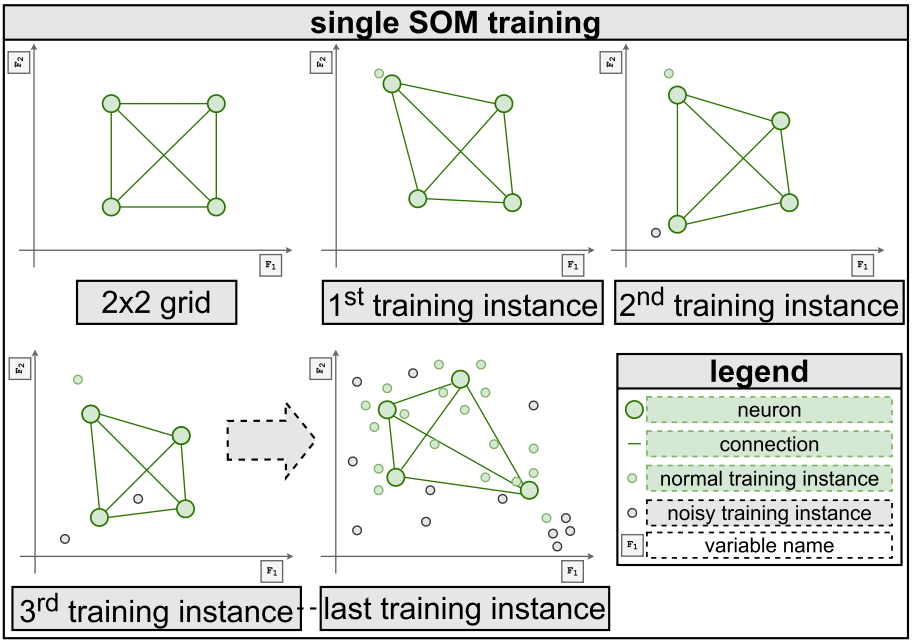


Fig. 1 Training process of a single SOM layer showing the adaptation of the layer throughout the presentation of multiple training instances (normal and noisy)

Let *epochs* be the number of desired iterations for SOM model training. For each training instance $x_{(\cdot,i)}$ in the dataset, the winner neuron $w(x_{(\cdot,i)})$ is identified. Such neuron is considered to adapt a SOM S at epoch t (hereinafter $S(t)$), represented as a matrix of L-dimensional neurons defined as:

$$S(t) = \{m_{(\cdot,k,k')}(t)\}_{(k,k')}, \tag{4}$$

where $m_{(\cdot,k,k')}(t)$ represents a L-dimensional neuron in position k and k' of $S(t)$. The adaptation process can be formalized as:

$$m_{(\cdot,k,k')}(t+1) = m_{(\cdot,k,k')}(t) + \left[\frac{\sum_{i=1}^{n} h(w(x_{(\cdot,i)}), m_{(\cdot,k,k')}(t)) \times (x_{(\cdot,i)} - m_{(\cdot,k,k')}(t))}{\sum_{i=1}^{n} h(w(x_{(\cdot,i)}), m_{(\cdot,k,k')})} \right]_{l=1,\ldots,L}, \tag{5}$$

where $h(\cdot,\cdot)$ is the neighborhood function defined as:

$$h(m^{(1)}_{(\cdot,k_1,k'_1)}(t), m^{(2)}_{(\cdot,k_2,k'_2)}(t)) = \exp\left(-\frac{(|k_1 - k'_1| + |k_2 - k'_2|)^2}{2\sigma(t)^2} \right), \tag{6}$$

and $\sigma(t)$ corresponds to the width of the neighborhood function:

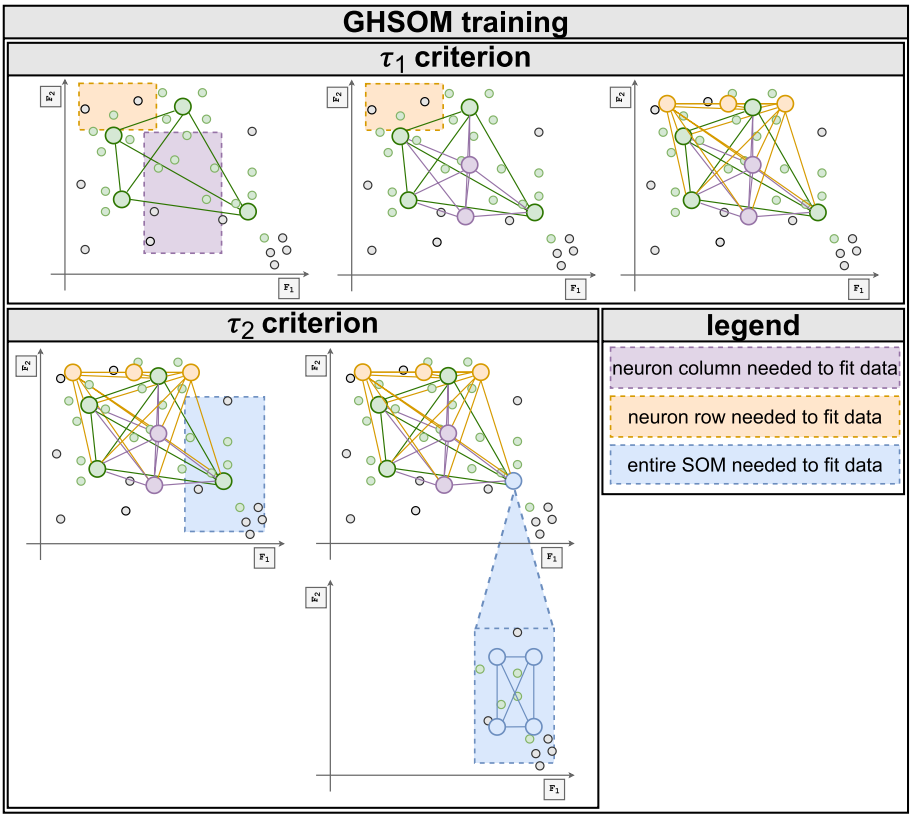


Fig. 2 A graphical representation of the GHSOM training phase following the τ_1 and τ_2 criteria. While τ_1 controls the horizontal (rows) and vertical (columns) growth of a single SOM layer, τ_2 controls the hierarchical growth, generating new SOM layers. The entire model is capable to analyze data at multiple densities

$$\sigma(t) = \left(\frac{\sqrt{R^2 + C^2}}{2}\right) \times \exp\left(-\frac{t}{epochs} \times \log\left(\frac{\sqrt{R^2 + C^2}}{2}\right)\right), \tag{7}$$

where R and C denote the number of rows and columns in the SOM, respectively.

Once the SOM has been trained, the growing process (consisting in the creation of new rows and columns in the SOM) and the hierarchical growth process (consisting in the addition of new SOMs which represent data at a finer granularity level) follow the GHSOM training procedure described above. Each SOM layer is evaluated for the τ_1 criterion for two-dimensional growth. Once the τ_1 criterion is satisfied (Eq. 1), we evaluate each neuron for the hierarchical growth considering the τ_2 criterion (Eq. 2). Algorithm 1 and Fig. 2 describe the GHSOM training process.

Algorithm 1 GHSOM model training

function GHSOM($dataset, \tau_1, \tau_2, epochs$)
  compute var_0
  $p\_list \leftarrow$ create list to identify the parent layer and neuron of the current layer
  $p\_layer \leftarrow 0$ ▷ For level-0 layer
  $p\_neuron \leftarrow 0$ ▷ Only neuron in level-0 layer
  $p\_list.add(p\_layer, p\_neuron)$
  $curr\_layer\_id \leftarrow 0$
  $i \leftarrow 0$
  while $i <$ p_list **do**
    $(curr\_p\_layer, curr\_p\_neuron) \leftarrow p\_list.get(i)$
    $curr\_dataset \leftarrow$ select instances from $dataset$ for the current layer
    $curr\_layer \leftarrow$ create a new layer of size 2×2 neurons
    $curr\_layer\_id \leftarrow curr\_layer\_id + 1$
    $flag\_2d\_growth \leftarrow false$
    repeat
      For each instance $x_{(\cdot,i)} \in curr\_dataset$ train $curr\_layer$ SOM as in Formula 5
        ▷ see distributed implementation in Algorithm 2
      evaluate quality of $curr\_layer$ (according to its variance)
      if $curr\_layer$ does not satisfy the τ_1 2-D growth criterion **then**
        grow the current layer
        $flag\_2d\_growth \leftarrow true$
      else
        $flag\_2d\_growth \leftarrow false$
      end if
    until $flag\_2d\_growth$ is $true$
    $expand\_neuron\_set \leftarrow$ create empty set
    for all $neuron \in curr\_layer$ **do**
      if $neuron$ does not satisfy the τ_2 hierarchical growth criterion **then**
        add $neuron$ to $expand\_neuron\_set$
        $p\_list.add(curr\_layer\_id, neuron.id)$
      end if
    end for
    $dataset \leftarrow$ map instances to winner neurons
    save the current SOM model
    $p\_list.remove(i)$
    i = i+1
  end while
end function

Algorithm 2 Distributed SOM training

function MAP(instance x, current layer $neuron\_layer$)
  $w(x_{(l,i)}) =$ winner neuron in the SOM layer for the current data instance $x_{(\cdot,i)}$
  for all $m_{(\cdot,k,k')} \in neuron\_layer$ **do**
    for all $l \in 1, \ldots, L$ **do**
      $numerator\_component[l] \leftarrow h(w(x_{(l,i)}), m_{(l,k,k')}(t)) \times (x_{(l,i)} - m_{(l,k,k')}(t))$
    end for
    $denominator\_component \leftarrow h(w(x_{(l,i)}), m_{(l,k,k')}$
    $yield(neuron.id, (numerator\_component, denominator\_component))$
  end for
end function

function REDUCEBYKEY($neuron\_id, val\_list =$
  $[(numerator\_component, denominator\_component), \ldots]$)
  $numerator \leftarrow 0$
  $denominator \leftarrow 0$
  for all $partial\_values \in val\_list$ **do**
    for all $l \in 1, \ldots, L$ **do**
      $numerator[l] \leftarrow numerator[l] \oplus partial\_values.numerator\_component[l]$
    end for
    $denominator \leftarrow denominator + partial\_update.denominator\_component$
  end for
  $updated\_weight\_vector \leftarrow numerator \cdot 1/denominator$
  $yield(neuron\_id, updated\_weight\_vector)$
end function

3.2 Multi-density anomaly detection

Once the entire set of training instances is processed, the learned SOM with its constituent neurons represents a *spatial memory* of the training instances. This capability allows our model to be trained with training instances that mostly refer to normal cases that may just be marginally affected by noise (background data)[2], without representing anomalies in the model. By doing so, the final predictive model has the ability to discriminate between normal and anomalous cases considering the distance between a new unlabeled instance and the part of the model that describes a subset of normal instances and is the most closely related to the new instance.

Once the set of neurons within the hierarchical structure of SOMs is set, the hierarchy obtained can thus be used to tackle the anomaly detection task. In particular, when a new unlabeled instance is provided to the hierarchy, the algorithm looks for its winner neuron. Once found, it is used to predict the class (anomaly/not anomaly) of the new instance, based on the distance between the instance and the winner neuron.

More formally, let $x_{(\cdot,i)}$ be the new example to be considered, and $\mathcal{N}$ the entire set of neurons of the trained hierarchy of SOMs defined as $\mathcal{N} = \bigcup_{S \in \mathcal{G}} m_{(\cdot,k,k')} \in S$,, where S is a single SOM once the training process is completed, and $\mathcal{G}$ represents the entire hierarchical GHSOM model with its constituent SOMs. The closest winner neuron to $x_{(\cdot,i)}$ is defined as:

$$w(x_{(\cdot,i)}) = \underset{w \in \mathcal{N}}{\operatorname{argmin}} \{dist(x_{(\cdot,i)}, w)\}. \tag{8}$$

Therefore, the unlabeled instance is considered an anomaly if the following inequality holds:

$$dist(x_{(\cdot,i)}, w(x_{(\cdot,i)})) > mqe_0 + tf \times \sigma, \tag{9}$$

where $dist(a, b)$ is the Euclidean distance between two input vectors a and b, σ is the standard deviation of the distances among the training instances and the neurons after the training, and *tf* the threshold governing the sensitiveness of the final predictive function.

Conceptually, tackling anomaly detection with this hierarchical spatial memory has the ability to naturally fit multi-density distributions, which can be comprised of sub-distributions. In this context, samples can assume values within sub-ranges of a larger distribution, and anomalies may be hidden at different density levels. Formalizing the multi-density distribution can be achieved leveraging the notion of mixture of distributions. More formally, assuming that the underlying data distribution is normal, a multi-density distribution can be defined using the following notation:

$$F(\mathbf{X}) = \sum_{i=1}^{D} w_i \frac{\exp\left(-\frac{1}{2}(X - \boldsymbol{\mu}_\mathbf{i})^T {\boldsymbol{\Sigma}_\mathbf{i}}^{-1} (\mathbf{X} - \boldsymbol{\mu}_\mathbf{i})\right)}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}_\mathbf{i}|}} = \sum_{i=1}^{D} w_i \, F_i(\mathbf{X}), \tag{10}$$

where $F(\mathbf{X})$ is the (multi-density) multidimensional probability density function (PDF), which combines several (single-density) multidimensional normal distributions $F_i(\mathbf{X})$. In the formula, $\mathbf{X} \in \mathbb{R}^p$ represents a data point as a p-dimensional vector, D is the total number of single-density multidimensional distributions, $\boldsymbol{\mu}_\mathbf{i}$ and $\boldsymbol{\Sigma}_\mathbf{i}$ are the p-dimensional

[2] Note that our model does not exploit labels during training. As a result, background data containing a degree of noise or anomalies is acceptable.

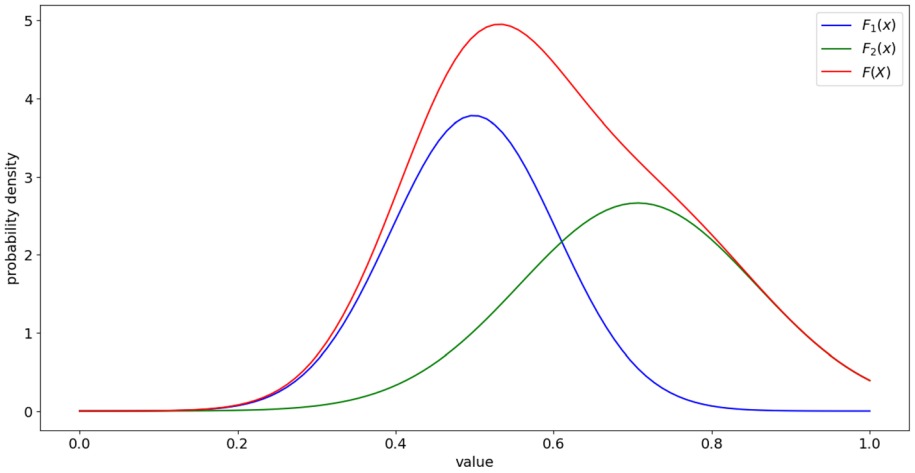


Fig. 3 An example of a multi-density distribution following Eq. (10)

mean vector and the positive finite covariance matrix of size $p \times p$ of the i-th distribution respectively. $|\mathbf{\Sigma_i}|$ represents the determinant of the covariance matrix. w_i represents the weight assigned to each i-th multidimensional single-density normal distribution governing the various shapes and peaks that the $F(\mathbf{X})$ can assume. A graphical representation of the resulting multi-density distribution $F(\mathbf{X})$ is depicted in Fig. 3.

We note that, in our work, we assume that all parameters of the multi-density distribution are not known in advance, but can be automatically learned by the GHSOM model during the training process. We argue that identifying anomalies with a hierarchy of SOM models should, in principle, allow for a more precise detection of anomalies than a single flat model, which is the most widely adopted approach.

If we refer to a single-density distribution F_i from the entire multi-density distribution $F(\mathbf{X})$, a data point is classified as normal by the GHSOM model if it can be recognized by at least one single-density distribution. Therefore, the set of normal data points can be defined as

$$N = \bigcup_{i=1}^{D} N_i, \text{ where } N_i \subset \mathbb{R}^p \text{ is the set of examples generated according to } F_i\,. \quad (11)$$

Therefore, we can define the set of anomalous instances as $A = \mathbb{R}^p \setminus N$, that is, the set of each other possible p-dimensional data point resulting out-of-distribution w.r.t. $F(\mathbf{X})$. Or, in other terms, the set of each possible p-dimensional data point that does not follow any single-density distribution $F_i(\mathbf{X})$.

This aspect of our method is particularly relevant considering that multi-density data and anomalies can be identified in many real-world applications with sensor data contexts. In vehicular traffic analysis, for instance, levels of traffic could be minimal during the night and reach their peak at noon, due to events such as the end of the school day. The same phenomenon can be observed in domains such as pedestrian flows, brightness levels, and so forth. In addition to variations of behavior due to different temporal phases, multiple densities can be also identified in geo-distributed settings. In these settings, multiple geographic locations naturally exhibit a varying degree of intensity for a phenomenon under

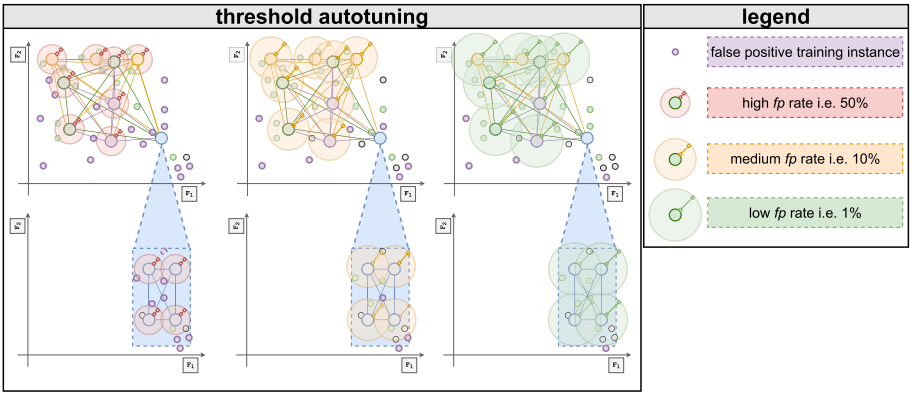


Fig. 4 The *tf* autotuning phase to determine the final predictive function w.r.t. the considered *fp*% tolerance. From left to right, it is possible to observe a reduction of the *fp* rate, achieved through threshold adaptation. This process is illustrated by the growing radius of the decision boundary of the model

analysis, due to their inherent spatial characteristics and external factors such as weather conditions, which are also subject to seasonal dependencies. Examples also include renewable energy production and air pollen distribution.

3.3 Threshold autotuning for anomaly detection

The key idea of our autotuning approach is to leverage the predictive function to compute distances between neurons (*normal prototypes*) and unlabeled instances. The assumption is that the distance between anomalous instances and prototypes is, in principle, greater than the distance between normal cases and prototypes. Following this assumption, our method identifies the correct threshold factor (*tf*) used in the predictive function according to a false positive rate (*fp*%) tolerance within the training set.

Algorithms 3–4 and Fig. 4 further illustrate the *tf* autotuning process w.r.t. the considered *fp*% .

Since the precise configuration of *tf* is often impractical and error-prone in real settings, this step is crucial to support properly calibrating the predictive function. Domain experts and final users can more easily understand *fp*% as the rate of false alarms which corresponds to the sensitivity of the final predictive model. Furthermore, it is possible to properly configure the *fp*% to account for noisy or contaminated instances residing in training data. Let us consider a case with background data that only contains normal cases (free of anomalies). A straightforward scenario is to define $fp\% = 0$ so that no false alarms are issued on training data. Another case is that of background data containing noise or contamination. In such a case, setting a low percentage of false alarms, i.e., $fp\% = 1, 2, \ldots, n$, where n is the estimated contamination rate, will allow the model to be more robust and account for the estimated anomalies residing in training data.

Once the ratio of acceptable false positives w.r.t. the training set cardinality is achieved, the final predictive model is configured for detecting anomalies in real-domain sensor network data scenarios. Obviously, the higher *fp*% the lower the precision, and the higher the recall of anomalies. In any case, *fp*% is independent on the data distribution, while *tf* is not. A graphical representation of the predictive phase is provided in Fig. 5.

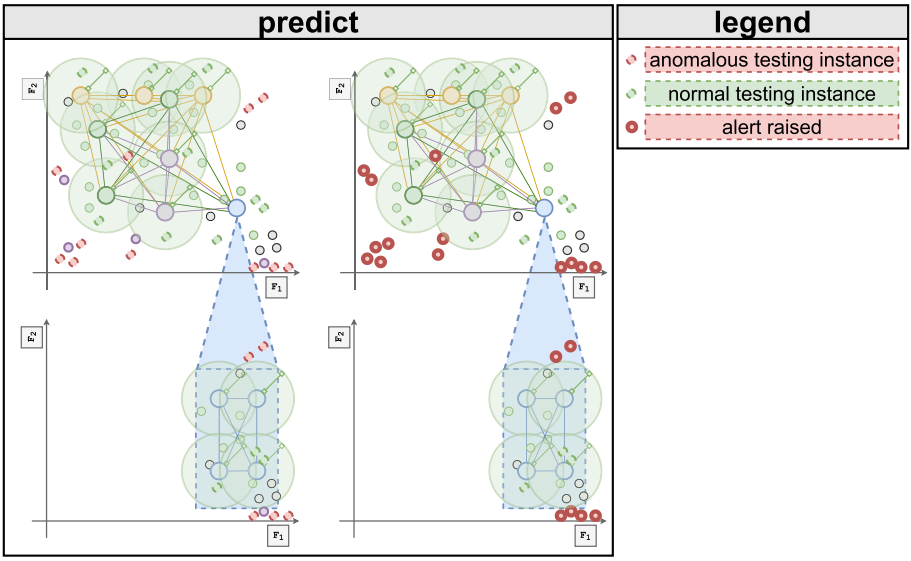


Fig. 5 A graphical illustration of the predictive phase. At inference time, all instances within the model's decision boundary are considered as normal, while all instances outside the decision boundary are classified as anomalies. Geometrically, the decision boundary defined through the radius (right side of Eq. 9) and the prediction are based on the distance between the instance and the winning neuron (left side of Eq. 9)

3.4 Explainable anomaly detection

The anomaly detection step supports two types of output, depending on the desired level of detail. The simplest approach provides the classification of an unlabeled instance in the form of a Boolean response (normal/anomaly). This type of output is useful to raise alerts when the instance is an anomaly. Its drawback is that interpreting the raised alert could be difficult for domain experts. To deal with this issue, we propose a second type of output that combines the boolean response with a feature ranking that indicates the importance that each feature had in the anomaly identification process. Feature importance is a numerical value between 0 and 1, indicating how anomalous the value expressed by the feature is with respect to the training data. The sum of all features importance values in the feature ranking is equal to 1. To estimate feature importance, we compute the distance between the current instance under analysis and the winner neuron. Specifically, in a normalized space, the ranking is proportional to the contribution provided by each single vector component in the Euclidean distance between $x_{(\cdot,i)}$ and $w(x_{(\cdot,i)})$. More formally, the feature importance function for the instance $x_{(\cdot,i)}$, $f_{imp}(x_{(\cdot,i)}, l)$, is computed as follows:

$$f_{imp}(x_{(\cdot,i)}, l) = \frac{(x_{(\cdot,i)}[l] - w(x_{(\cdot,i)})[l])^2}{\sum_{l'=1}^{L}(x_{(\cdot,i)}[l'] - w(x_{(\cdot,i)})[l'])^2} \tag{12}$$

where l represents the feature index, $l' \in \{1, 2, ..., L\}$ a varying feature index, L the feature set cardinality, while the ranking is defined as follows:

$$rank(x_{(\cdot,i)}) = \nabla(\{f_{imp}(x_{(\cdot,i)}, 1), f_{imp}(x_{(\cdot,i)}, 2), ..., f_{imp}(x_{(\cdot,i)}, L)\}) \tag{13}$$

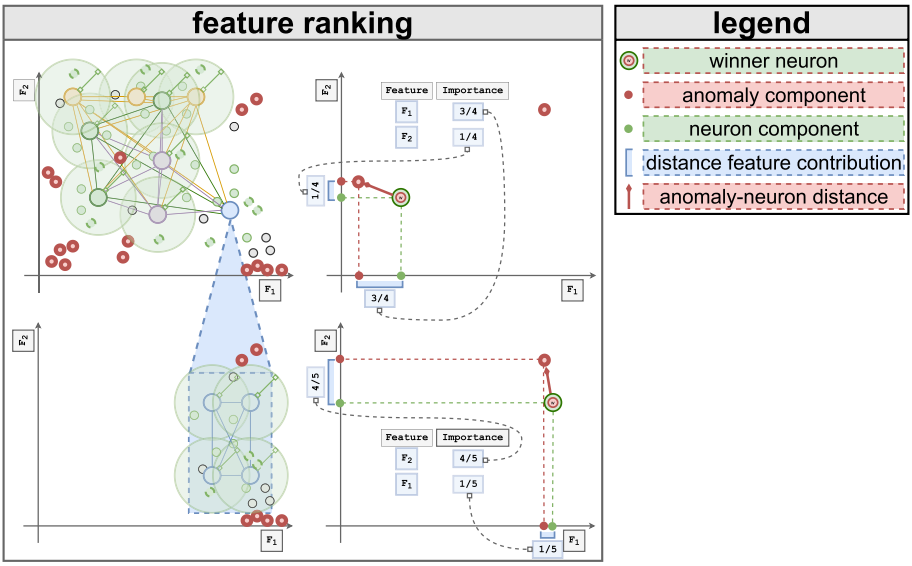


Fig. 6 A graphical illustration of the feature ranking for anomaly explanation in two dimensions ($L = 2$). The features F1 and F2 have an importance 0.75 and 0.25 respectively for the detected anomaly on the top figure, while for the bottom figure, F1 and F2 have an importance 0.2 and 0.8 respectively

where ∇ is a descending ordering operator.

This approach effectively supports the identification of feature(s) that contributed the most for detecting the anomaly and, therefore, enhancing our method with explainability capabilities. Algorithms 4 and 5 include the pseudo-code for the prediction and the explainability phases of our method. In Fig. 6, we graphically illustrate the instance-based feature ranking process for each detected anomaly.

```
function EstimateThreshold(max_fp, max_tf, step, train_instances, model)
    map_instance_min_dist ← ∅
    avg_d ← train_instances.get_avg_distances()
    std_dev_d ← train_instances.get_std_dev_distances()
    for all train_instance ∈ train_instances do
        min_dist ← ∞
        for all neuron N ∈ model do
            d ← Euclidean_dist(N, train_instance)
            if d < min_dist then
                min_dist ← d
            end if
        end for
        map_instance_min_dist ← map_instance_min_dist ∪ ⟨train_instance, min_dist⟩
    end for
    current_fp ← 0
    tf ← max_tf
    while current_fp < max_fp do
        for all ⟨train_instance, min_dist⟩ ∈ map_instance_min_dist do
            thresh ← avg_d + (tf * std_dev_d)
            if min_dist > thresh then
                current_fp ← current_fp + 1
            end if
        end for
        tf ← tf − step                    ▷ decreasing tf increases the possibility to have false positives
    end while
    emit(thresh)                          ▷ final threshold for the predictive function
end function
```

```
function DistributedPredict(neurons, thresh, dataset)
    predictions ← dataset.map(instance ⟹ {                    ▷ start mapping instance-prediction
    min_dist ← ∞
    closest_neuron ← null
    for all neuron ∈ neurons do
        dist ← EuclideanDist(neuron, instance)
        if dist < min_dist then
            min_dist ← dist
            closest_neuron ← neuron
        end if
        ranking ← FeatureRanking(instance, closest_neuron, min_dist)        ▷ see Algorithm 5
    end for
    if minDist > thresh then
        (instance, min_dist, anomaly, ranking)
    else
        (instance, min_dist, normal, ranking)
    end if
})                                                            ▷ close distributed map
    emit(predictions)
end function
```

```
function FeatureRanking(test_instance, closest_neuron, Euclidean_dist)
    L ← test_instance.size()                                  ▷ get the number of vector components
    distances ← EmptyArray[Double]
    ranking ← EmptyMap[Int, Double]
    for all l ∈ 1, . . . , L do                               ▷ for each vector component
        diff ← (test_instance[l] − closest_neuron[l])^2       ▷ vector component difference
        distances(l) ← diff
    end for
    for all l ∈ 1, . . . , L do
        feature_importance ← distances(l)/Euclidean_dist
        ranking.put(l, feature_importance)
    end for
    ranking ← ranking.toSeq.sortWith(_._2 > _._2).toMap       ▷ sort by feature importance
    emit(ranking)
end function
```

4 Experiments

In this section, we present our experiments and discuss their quantitative and qualitative outcomes. We considered different application domains that could be supported by anomaly detection tasks. First, quantitative analyses were conducted for anomaly detection in the context of electrical grids with two real-world datasets (Wind NREL, PV Italy). To set up the training regimen in these experiments, we adopted sliding windows of different sizes (Gama, 2010). Second, we carried out experiments for urban anomaly detection by analyzing two real world datasets: pedestrian flows encountered in the Oslo city and vehicular traffic flow in one Italian city. The training regimen followed for this task is that of landmark windows, where models are trained with all data available until a given time point (Gama, 2010).

Third, the experiments with landmark windows for anomaly detection in web services are presented by considering the Yahoo! dataset (Hagemann & Katsarou, 2020, https://research.yahoo.com.

Further experiments are reported in Appendices 1 and 2. Specifically, in Appendix 1, we present the scalability analysis conducted to emphasize the capability of the proposed method to distribute the computational workload over multiple nodes. While, in Appendix 2, we present the qualitative analyses for detecting anomalous allergenic pollens in the air

for the Veneto region in Italy, emphasizing the capability of the proposed method to discriminate among different types of anomalies occurring in this context.

4.1 Quantitative results

Comparative analyses were performed to quantitatively assess the effectiveness of the proposed method when compared with state-of-the-art anomaly detection methods. In line with our discussion of existing works, we considered popular methods from the most representative class of approaches to address the task of interest in our study, i.e., one class learning. Indeed, this class of approach offers the flexibility to learn a model from an initial (regular) data distribution that is able to flag data that significantly differ from the learned distribution. In particular, we considered five well-known and widely adopted competitor methods: *OCSVM* (Schölkopf et al., 2000), *Isolation Forest* (Liu et al., 2008), *ABOD* (Kriegel et al., 2008; Pham & Pagh, 2012; Jahromi et al., 2022), *COPOD* (Li et al., 2020), and an *Auto-encoder* architecture that detects anomalies based on the reconstruction error (Beggel et al., 2019; Zhou & Paffenroth, 2017). Such methods were configured by performing ablation analyses on a set of parameters' values suggested in the respective original papers. Specifically, for OCSVM, we considered two different kernel functions, i.e., Linear and Radial Basis Function (RBF), and we varied the μ parameter within the set $\mu \in \{0.5, 1.0\}$.

For Isolation Forest, we considered the number of trees in the ensemble $n\_estimators \in \{10, 20, 50\}$, while the whole feature set was considered for every single tree.

For ABOD and COPOD, we used the default configuration proposed in the pyod (Zhao et al., 2019) library. For ABOD, the number of neighbors considered for each data point is set to 10 (Zhao et al., 2019).

For the auto-encoders, we used the architecture suggested by Bengio et al. in Bengio (2012). Specifically, we used a standard contractive model architecture with four layers using the Sigmoid activation function. Additional experiments using ReLU activations did not yield any significant reduction in model training time nor differences in terms of model performance. We experimented with different negative powers of 10 for the configuration of the *learning_rate* and with different powers of 2 for the *batch_size*. Preliminary experiments suggested that the different configurations did not provide a significant difference in terms of performance metric values. Therefore, the experiments were executed with the following configuration: $epochs = 50$, $learning\_rate = 0.0001$, $batch\_size = 32$. Since auto-encoders are sensitive to the embedding space dimensionality, we considered two sizes proportional to the feature set cardinality of the input space. Specifically, let $orig\_dim$ be the number of features of the original dataset, we considered embedding spaces of $dim \in \{orig\_dim/4, orig\_dim/2\}$ sizes. This choice allowed us to be flexible with respect to different feature dimensionalities of the different datasets. During the inference phase, if the reconstruction error of an unlabeled instance exceeds $p \times \sigma$ (corresponding to a 3-sigma rule when $p = 3$) from the average reconstruction error observed on the training set, the instance is marked as an anomaly.

In our proposed method, the τ_1 and τ_2 parameters were set considering a range of values balancing between computational time, model complexity (number of neurons), and model accuracy. We selected $\tau_1, \tau_2 \in \{0.9, 0.8, 0.7, 0.6, 0.5\}$. This choice takes into consideration the fact that model training time becomes unsustainable with values smaller than 0.5 since the quality criteria become too difficult to satisfy and that, on the other hand, the accuracy

is likely negatively affected by values higher than 0.9 due to the shallowness of the model that this choice implies. After preliminary experiments, the final configurations used in the experiments reported in our results are: Wind NREL ($\tau_1 = 0.9$, $\tau_2 = 0.9$), PV Italy ($\tau_1 = 0.7$, $\tau_2 = 0.9$), Oslo Pedestrian Flow ($\tau_1 = 0.5$, $\tau_2 = 0.9$), vehicular traffic ($\tau_1 = 0.8$, $\tau_2 = 0.9$), and Yahoo! web services ($\tau_1 = 0.8$, $\tau_2 = 0.8$).

4.1.1 Energy results

In this section, we describe the experiments and analyze the results extracted for anomaly detection in the context of electrical grids. To this aim, we considered the following two datasets.

- *Wind NREL.* This dataset[3] includes time series data from five wind farms of wind speed and production recorded every 10 minutes for a two-year period from January 1, 2005, to December 31, 2006. The data was then aggregated at an hourly level.
- *PV Italy.* The dataset includes data collected every 15 minutes (from 2:00 AM to 8.00 PM, every day) by sensors at 17 photovoltaic power plants located in Italy, spanning from January 1st, 2012 to May 4th, 2014. Anomalies consists in perturbations of the correct attribute values. This is done on 25% of instances and 50% of the features. Further information on the data preprocessing steps can be found in Corizzo et al. (2021); Ceci et al. (2016).

For both datasets, we consider the following features: latitude and longitude of each plant; day and hour; altitude and azimuth; weather conditions, i.e., ambient temperature, irradiance, pressure, wind speed, wind bearing, humidity, dew point, cloud cover, and a descriptive weather summary. Weather conditions are either measured (training phase) or forecasted (detection phase). In particular, all weather observations were extracted using the Forecast.io API, except for the expected altitude and azimuth, which were extracted from SunPosition[4], and the expected irradiance (PV Italy dataset only), that was extracted from PVGIS[5].

We performed experiments considering sliding windows of 60 consecutive days for model training. Once models are trained, their anomaly detection performance is assessed on data observed the following day (61st day), considered as the prediction day. Experiments are repeated 10 times with 10 distinct sliding window selections, and the obtained results are averaged.

Figures 7 and 8 show the detailed results, in terms of macro F1-Score, for each of the 10 considered prediction days (for Wind NREL and PV Italy, respectively), while Fig. 9 shows the ablation analyses according to different *fp*% tolerance settings chosen for the predictive models. According to our ablation analyses, we selected 2% and 9% of false positive tolerance for Wind NREL and PV Italy, respectively. On average, with the Wind NREL dataset, the proposed method achieved an improvement of 9.5% w.r.t. the second ranked method OCSVM. Regarding PV Italy, our method achieved an improvement of 3.7% compared to the second ranked method OCSVM (see Table 3).

[3] https://www.nrel.gov/wind
[4] https://www.susdesign.com/sunposition
[5] https://re.jrc.ec.europa.eu/pvg_tools/en

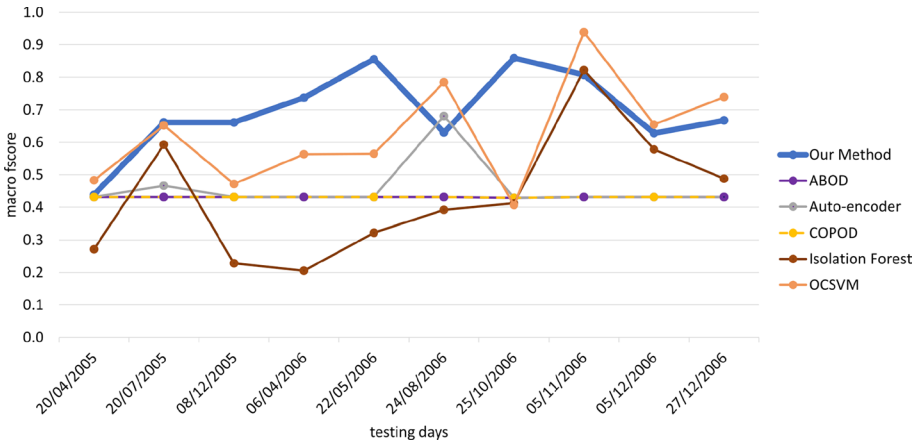


Fig. 7 Results in terms of macro F1-Score for each of the considered prediction days (Wind NREL dataset). The performance curves reported consider the best configuration for each method

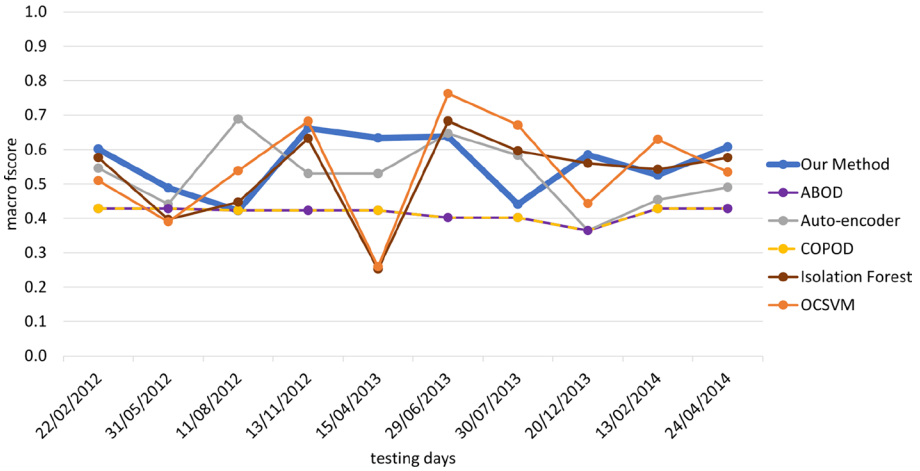


Fig. 8 Results in terms of macro F1-Score for each of the considered prediction days (PV Italy dataset). The performance curves reported consider the best configuration for each method

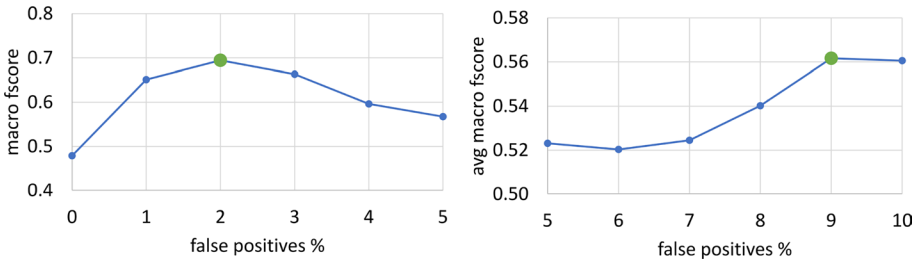


Fig. 9 Ablation analysis for threshold autotuning conducted considering a range of false positive percentage for Wind NREL (left) and PV Italy (right)

Table 1 Descriptive features of the vehicular traffic dataset

Feature name	Description
TimeId	Starting time of the 5-minutes time window
No_approaching	# vehicles with direction "approaching"
No_leaving	# vehicles with direction "leaving"
No_unknown	# vehicles with direction "unknown" (i.e., unclassified)
X	GPS latitude
Y	GPS longitude
Camera_name	The name of the camera/sensor
Month	The month of the measurement
Day	The day of the measurement
Hour	The hour of the measurement
Minute	The minute of the measurement

4.1.2 Vehicular traffic results

The experiments were conducted considering the vehicle traffic observed in an Italian city. Specifically, 93 sensors located at every access to the city center were considered for data collection. For each sensor, the GPS position was exploited for geo-tagging instances with latitude and longitude coordinates. Data was collected continuously (ISO 8601, resolution: seconds) for the time frame between November 8, 2021 and November 23, 2021. Instances were aggregated every 5 minutes in order to quantify the number of vehicles approaching or leaving the city center, for a total of 370, 775 instances. Table 1 summarizes the descriptive features of the dataset.

Experiments were performed considering a landmark time window approach, where models are trained with all available data observed so far to predict on subsequent time steps. Specifically, for each hour of the time frame covered by the dataset, we trained a model to predict the possible anomalies occurring in the next hour. Multiple train and prediction sets were created where the n-th split includes n hours for training and the $(n + 1)$-th hour for prediction. Since anomalies were not provided, 36 of the available prediction windows randomly selected were perturbed by adding random values in the interval [0,100] to the number of vehicles with direction approaching and unknown, without changing the original values for the vehicles leaving the area. The idea was to simulate an anomalous but realistic scenario when cars are only approaching without the possibility to leave the city center. Therefore, we trained models from the beginning of the time series until the hour that precedes the prediction window. By doing so, we collected 36 predictive models, trained with a progressively increasing amount of training data. Quantitative results are illustrated in Fig. 10.

Results show that, at the beginning of the considered time frame, models are more unstable, possibly due to the distinct distribution of patterns observed during the day and the night. However, after 150 hours of training, predictive models appear more stable and accurate, as expected. In the latest time windows it is possible to observe, in some cases, an unsatisfactory model performance. This behavior can be observed in three specific situations with limited or no sunlight: November 19th 2021 (10:05pm), November 20th, 2021 (7:05pm), and November 20th (11:05pm), as shown by the three drops in the final part of

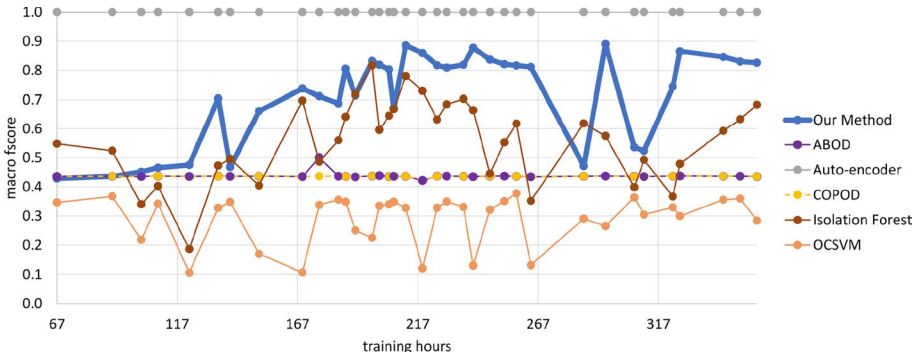


Fig. 10 Vehicular traffic analysis. Each curve considers the best configuration for each method in terms of macro F1-Score (y-axis). The x-axis represents the width of the time window spanning the entire time frame of the analyzed dataset

the curve. This could be explained by a reduced number of observations available during the night, motivated by the reduced vehicular traffic.

It can be noted that Auto-encoder yielded perfectly accurate predictions with this dataset, resulting in a macro F1-Score of 1.0. This behavior was expected and can be attributed to the perturbation strategy adopted, which is fully compliant with the $3 \times \sigma$ rule used for the Auto-encoder predictive function, and results in a simplification of the task for this particular method. It should be also noted that experiments with Auto-encoder required a significantly high training time, due to training regimen followed for this dataset. Moreover, since the method does not provide explanations for its inferences, it fails to provide a trade-off between accuracy and explainability.

For the proposed method, experiments were conducted to automatically tune the *tf* parameter. To this aim, we considered different values for the *fp*% parameter, i.e., the tolerance on the percentage of training instances labeled as anomalies by our method that actually represent normal cases in the training set. Considering *fp*% rather than *tf* simplifies the adoption of the tuning mechanism from the perspective of non-expert security operators, since it is easier to understand. To set up the correct value of *fp*%, we considered the latest window containing all the training instances (i.e., after 358 training hours). As shown in Fig. 11, the highest macro F1-Score was obtained by considering 0.02% of false positives. We note that the results improved as the number of false positives within the training set is reduced, as visible in the decreasing curve shown in the top-left sub-figure. This is an expected result, since a low number of errors on training data corresponds, in principle, to a more robust predictive model that avoids recognizing normal cases as anomalous. Figure 11 also highlights that with 0% of false positives the model has a tendency to become flat, considering all the instances as normal cases. This behavior should be avoided, since the final models would become useless in practical real scenarios, and would not be able to discriminate between normal and anomalous instances. To avoid this phenomenon, we tuned the *fp*% parameter between 0% and 0.1% as shown in Fig. 11. Values close to 0% emphasized that the proposed method is capable to tolerate a few noisy instances that could be significantly different from historical data. This behavior is motivated by a potential degree of contamination in training data, i.e. anomalous instances considered as normal and not removed from training instances. This phenomenon is frequent in real applications such as vehicle flow analysis, where the effort of removing car accidents or unconventional

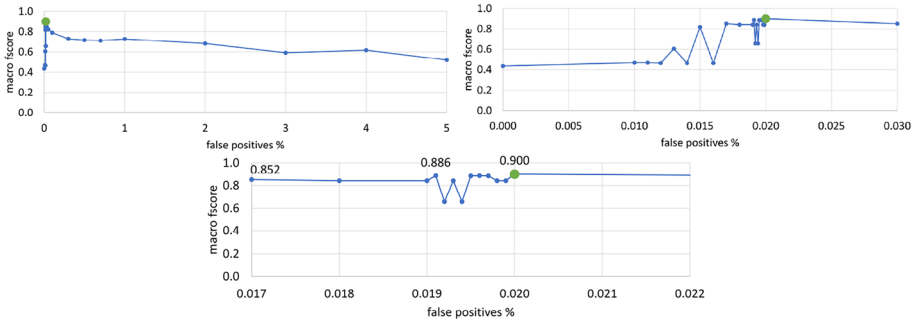


Fig. 11 Ablation analysis for the proposed method (vehicular traffic data). A different number of false positives within the training set were considered to identify the ideal parameter configuration. Macro F1-Score values w.r.t. close to zero false positive percentages emphasized that a few false positives are necessary to better tune predictive models for this dataset

vehicle flows due to municipal road works is too high. As a result, such instances are likely to remain within the training set. However, it is remarkable to observe that our method was capable of handling this problem via the specific $fp\%$ parameter tuning.

4.1.3 Oslo pedestrians flow analysis

We considered a dataset describing the flow of people in the city of Oslo between two areas identified by the following GPS coordinates:

- $\langle 59.91301053377869, 10.733979291595263 \rangle$
- $\langle 59.912625158578805, 10.734914979884614 \rangle$

Collected data covers a time period ranging from April to May 2019 to 2022. This time frame was decided in order to catch the abrupt increase in pedestrian flows occuring during the Constitution Day of Norway on May 17. To this aim, we considered all instances of May 17 as anomalous, whereas the remaining instances were considered as normal cases, i.e. regular pedestrian flows. The detailed descriptions of the features covered by this dataset are reported in Table 2.

In Fig. 12, we show the results in terms of macro F1-Score. We note that, for this dataset, we did not resort to any perturbation strategy, since it already contained real anomalies.

4.1.4 Yahoo! web services

We considered the Yahoo! S5 dataset[6] (https://research.yahoo.com) containing real time-series with labeled anomalies for the metrics of various Yahoo! services. The dataset consists of 67 univariate time series that were joined considering the temporal dimension as the join key to obtain a multivariate dataset. In the raw version of the dataset, the ground truth label information is available separately (anomaly/normal) for each

[6] https://webscope.sandbox.yahoo.com/catalog.php?datatype=s%20&did=70.

Table 2 Descriptive features of the Oslo pedestrians flow dataset

Feature name	Description
interval_start_time	Timestamp for the current 5-minute time window
X	GPS latitude
Y	GPS longitude
dateTimeDayofTheWeek	The number of the day of the week
Year	The year of the measurement
Month	The month of the measurement
Day	The day of the measurement
Hour	The hour of the measurement
Minute	The minute of the measurement
People	# people passing over the sensor

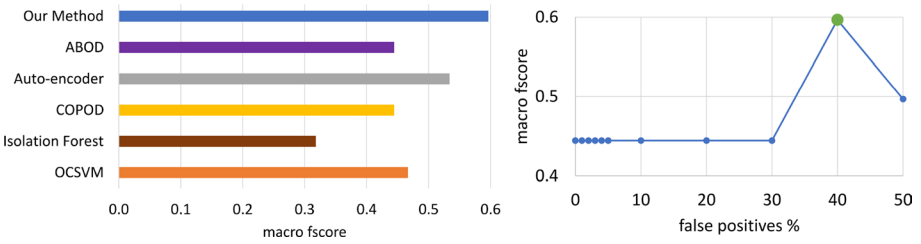


Fig. 12 Results for the Oslo pedestrian flows dataset (left) and ablation analysis for the *fp%* parameter optimized w.r.t. the macro F1-Score (right)

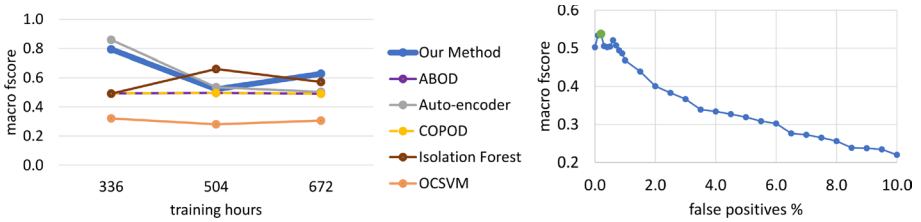


Fig. 13 Results for the Yahoo! web services dataset (left) and ablation analysis for the *fp%* parameter optimized w.r.t. the macro F1-Sscore (right)

feature. One time point refers to a specific hour of production traffic. Our experiments consider a training window of two weeks and a testing window of one week, resulting in landmark training windows of increasing time: two, three, and four weeks. We removed anomalies from the training windows and we considered a testing instance as anomalous if the time window contained at least 3 out of the 67 features labeled as anomalous. We considered 3 features since choosing 1 or 2 features would have resulted in several anomalies being skipped in the training sets, leading to empty or very small training window sizes, while choosing 4 features would have resulted in testing windows without anomalies, leading to a more complex evaluation. In Fig. 13, we show the results in terms of macro F1-Score. The results show that, on average, our method outperforms

Table 3 Macro F1-Score summary of the comparative analysis for all the datasets describing five different real-domain applications

Method	Wind NREL	PV Italy	Vehicular traffic	Oslo pedestrians	Yahoo!
ABOD	0.43	0.42	0.44	0.44	0.49
Auto-encoder	0.46	0.53	**1.00**	0.53	0.64
COPOD	0.43	0.42	0.44	0.44	0.49
Isolation forest	0.43	0.53	0.56	0.32	0.57
OCSVM	0.63	0.54	0.42	0.47	0.30
Our method	**0.69**	**0.56**	0.72	**0.60**	**0.65**

The best results are highlighted in bold

Table 4 Average ranking of all methods considered in our experiments

Method	Wind NREL	PV Italy	Vehicular traffic	Oslo pedestrians	Yahoo!	avg rank
ABOD	4	5	4	4	4	4.2
Auto-encoder	3	3	**1**	2	2	2.2
COPOD	4	5	5	4	4	4.4
Isolation forest	6	4	3	6	3	4.4
OCSVM	2	2	6	3	6	3.8
Our method	**1**	**1**	2	**1**	**1**	**1.2**

The best results are highlighted in bold

competitor methods. It can be seen that the proposed method presents an opposite behaviour w.r.t. Isolation Forest for this dataset. Our interpretation of this result is that the second evaluation window contains more peripheral anomalies that lie very close to normal instances. These anomalies are likely easier to detect with simple classification rules rather than with clusters of neurons that model multi-densities. However, our method presents a higher performance than Isolation Forest in the other two evaluation windows. Ablation analysis was conducted by considering $fp\% \in [0, 1]$ by step 0.1. Figure 13 (right) showed that the best results obtained with this dataset are when $fp\% = 0.2$ attesting that this dataset could contain few to no noisy or anomalous instances within the training windows.

4.2 Quantitative analysis

Overall, the experiments show that the proposed method outperforms the considered state-of-the-art methods in the majority of the proposed scenarios described through different datasets (see Table 3). In Table 4 we show the ranking achieved by each method on all considered datasets, and their final average ranking. It can be noted that the proposed method is the best performing for all datasets except for the vehicular traffic analysis, where Auto-encoder outperforms all other approaches. We attribute this result to the perturbation strategy followed for this dataset, which matches with the biases of the predictive function of the Auto-encoder method and, as a result, translates to a simplified setting. The OCSVM method positions itself in the middle, providing an acceptable second-best performance for Wind NREL and PV Italy, a sub-par performance for Oslo pedestrians,

and an unsatisfactory performance for the Vehicular traffic dataset and on Yahoo!. We can observe that the widely adopted Isolation Forest method suffers in most of the considered applications. Indeed, such method presents high robustness when the anomalies lie at the boundaries of training instances, and are easy to isolate by means of very simple trees with limited depth. A similar behavior is observed for COPOD, and ABOD, resulting in a subpar anomaly detection performance. Therefore, in cases where the anomalies are very close to normal instances, better suited methods are characterized by the ability of catching finer diversities among normal and anomalous cases. In such a setting, we attribute the superior performance of our method to its ability to model multiple densities in the background data thanks to multiple SOMs.

5 Conclusion and future works

In this paper, we proposed a distributed and explainable SOM-based anomaly detection method. The method supports the analysis of background data characterized by multiple densities, by means of multiple SOMs arranged as a hierarchy. Moreover, the method is able to process large-scale data occurring in real-world domains leveraging the map-reduce programming paradigm and the Apache Spark framework. Differently than popular anomaly detection systems, the proposed method is capable of automatically identifying the threshold for the classification rule during the training stage. Its sensitiveness can be also configured to tolerate different levels of false positives during the training stage. Furthermore, the proposed method is enhanced with an explainability component that facilitates the interpretation of predicted anomalies leveraging instance-based feature ranking. The results show the effectiveness of the proposed approach, both qualitatively and quantitatively, in five different real-world applications. Qualitative analysis emphasized that the explainability component is effective in highlighting the severity of the detected anomalies. In future work, we will extend our automatic threshold estimation mechanism to automatically learn the ideal false positive percentage, without any dependence on user inputs. Moreover, possible extensions of the method include its adaptation to biological domains, where the identification of possible variations in subcellular genetic information is important to detect the onset of diseases.

Appendix 1. Scalability analysis

A key characteristic of our proposed method is its ability to distribute the computational workload over multiple nodes. In order to assess this capability, we conducted a scalability analysis. considering three synthetically generated datasets of 1, 2, and 3 million instances respectively, consisting of 10 features. Our evaluation of the method's ability to exploit additional computational nodes involved measuring the execution times and calculating the Speedup and Scaleup factors. Our computational cluster is equipped with one driver and three worker nodes with the following characteristics: 6 Intel(R) Core(TM) i7-8700 CPUs @ 3.20GHz and 64.0 GB RAM.

The results, as shown in Figs. 14, 15, and 16 highlight that our method is able to take advantage of additional nodes to efficiently process large-scale data. Specifically, as the data size increases, we observe an increase of the speedup factors with a minimal communication overhead using 3 computational nodes w.r.t 2 and 1. The resuls in terms of both

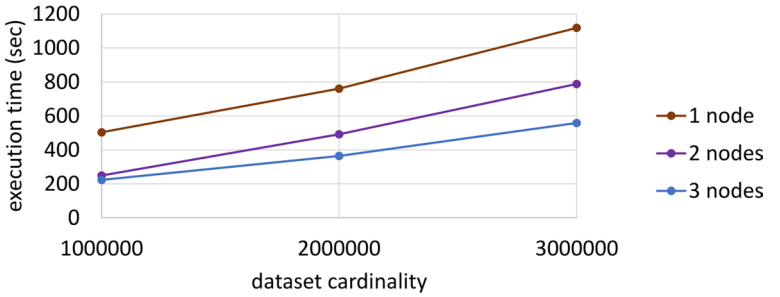


Fig. 14 The running times measured for conducting the experiments on synthetic datasets of 1,2, and 3 millions instances with 10 features

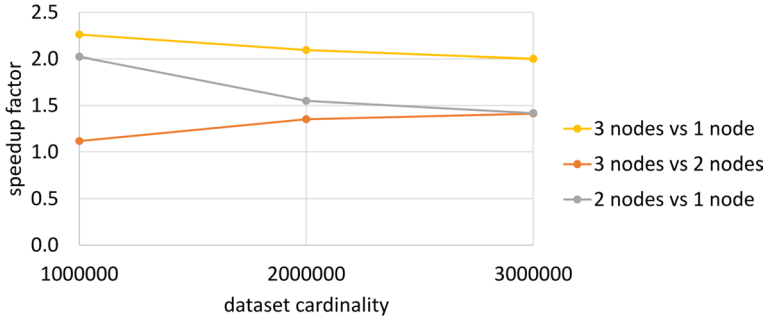
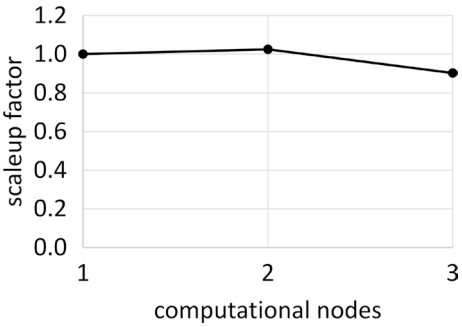


Fig. 15 Speedup factor calculated as the ratio between execution times observed with a single computational node (1) and with multiple computational nodes (2,3)

Fig. 16 Scaleup factor with simultaneously increasing problem complexity and computational nodes, i.e., 1 million instances with 1 node, 2 millions with 2 nodes, and 3 millions with 3 nodes



speedup and scaleup practically demonstrate that our proposed method presents satisfactory scalability.

This property translates in a significant advantage w.r.t. other methods, since it enables the analysis of arbitrarily large datasets by simply provisioning additional nodes to the cluster computing infrastructure. Other methods, on the other hand, may entirely fail to analyze large-scale data or may be impacted by significant delays, resulting in a major pitfall that limits their applicability in real-world applications.

Appendix 2. Qualitative analysis

2.1 Qualitative analysis

In this section, we show the capability of our proposed method to quantify the severity of different detected anomalies resorting to the instance-based feature ranking approach. We considered the task of pollen analysis using data collected by Arpa Veneto, an Italian environmental monitoring institution focused on measuring air and water quality over time. The extracted data contains daily environmental information from the city of Padova (Italy), in the time frame from years 2014 to 2019. Data is observed at a time granularity of one observation per day, with the exception of days in which measurements are not present. Data was extracted using the open data API of the ARPA Veneto website[7]. The information extracted includes the following features:

- Date of measurement (dd/mm/yyyy)
- Air temperature at 2 meters (°C), respectively mean, min and max temperature
- Total precipitation on the day (mm)
- Humidity at 2 metres (%), respectively min and max humidity
- Concentrations of allergenic pollen in the air (granules/m^3) per family, for a total of 25 families.

The bar plots in Fig. 17 highlight that the temperature attributes, as well as the minimum humidity, follow a normal distribution, whereas distributions of precipitation and

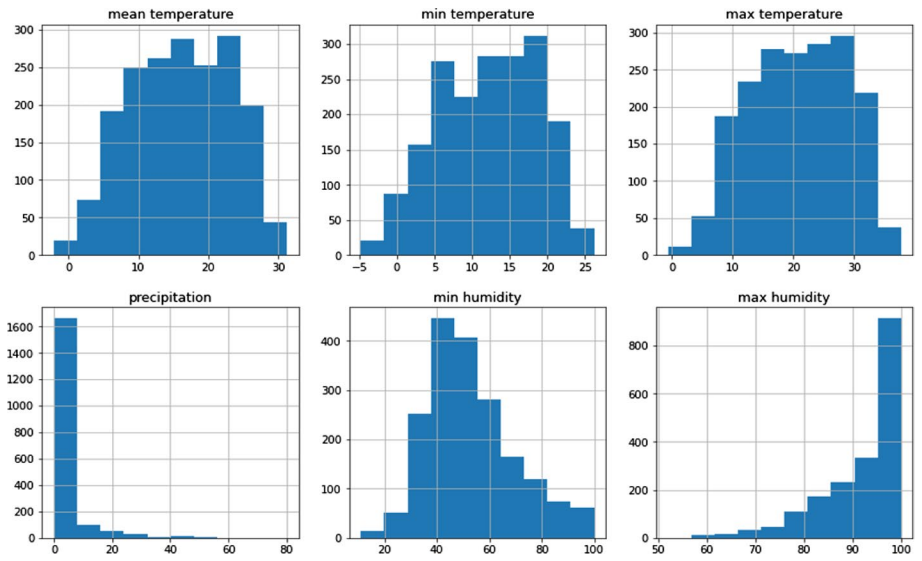


Fig. 17 Data distributions for the attributes mean, min and max temperature, precipitation, min and max humidity

[7] https://www.arpa.veneto.it/

Table 5 Reference table for the precipitation attribute

Rainfall intensity	mm/6 h	mm/12 h	mm/24 h
Weak	[0, 5]	[0, 10]	[0, 15]
Moderate	[0, 15]	[10, 30]	[15, 45]
Strong	[15, 30]	[30, 60]	[45, 90]
Very strong	[30, +∞]	[60, +∞]	[90, +∞]

max humidity are asymmetrical. The 25 families of pollens are considered as 25 distinct descriptive features. For the aim of the anomaly detection task, the focus was on determining whether a given instance/measurement belonging to the dataset is representative of a scenario characterized by anomalous levels of pollens in the air. To carry out an evaluation of the anomaly detection models, in this step, the main focus was to identify pollen levels that semantically describes an anomalous scenario depending on value frequencies. To this aim, we considered a publicly available source of data from an Italian network for aerobiological monitoring[8] and manually annotated the pollen dataset. In this experiment, we exploited some conclusions drawn from the feature ranking returned by our method. The first step is to identify, for each feature, reference tables containing concentration classes. An example of a reference table for the precipitation attribute is provided in Table 5[9].

We formulated the anomaly detection problem by attributing an anomaly weight for each value in the reference table in the [0, 1] range, where values 0 and 1 are associated with the lowest and highest anomalous values, respectively. Table 6 shows the anomaly weights for the precipitation attributes, associated with each value in the reference table. Similarly, we computed the frequencies for the other 25 pollen families considered as descriptive features (see Fig. 18).

This step allows us to enable a quantitative evaluation of the output generated by our method considering the available reference tables. To this end, we adopted a measure that we refer to as *anomaly level @ k (a_lv@k)*. Such measure has the advantage to emphasize the most severe anomalies, quantifying their level of severity cumulatively. This key feature is often needed in order to build an anomaly detector capable of issuing alerts only for severely anomalous cases, reducing the number of false alarms. Formally, given k, the k-th position in the feature ranking of the current prediction instance, the anomaly level at the k-th position is defined as $a\_lv@k = \sum_{i=1}^{k} w_i$, where w_i is the anomaly weight of the feature at the top i-th position of the feature ranking, with $i \leq k$ if the reference table of the feature at the top i-th position is defined, $w_i = 0$ otherwise, for each possible value of such feature, due to the lack of a reference map among the anomalous cases and the values of that specific feature. Thus, leveraging the anomaly level, it is possible to identify instances containing more anomalous values. By keeping track of the value assumed by the anomaly level at each position k in the feature ranking, it is possible to graphically represent the trend of the anomaly level. For instances identified as anomalous, we expect to see curves with a high slope at the first positions of the feature ranking and a gradual stabilization, until they no longer grow.

[8] http://www.pollnet.it/valori_di_riferimento_it.asp.

[9] https://www.arpa.piemonte.it/rischinaturali/approfondimenti/meteo/fenomeni_parametri_meteo/pioggia.html

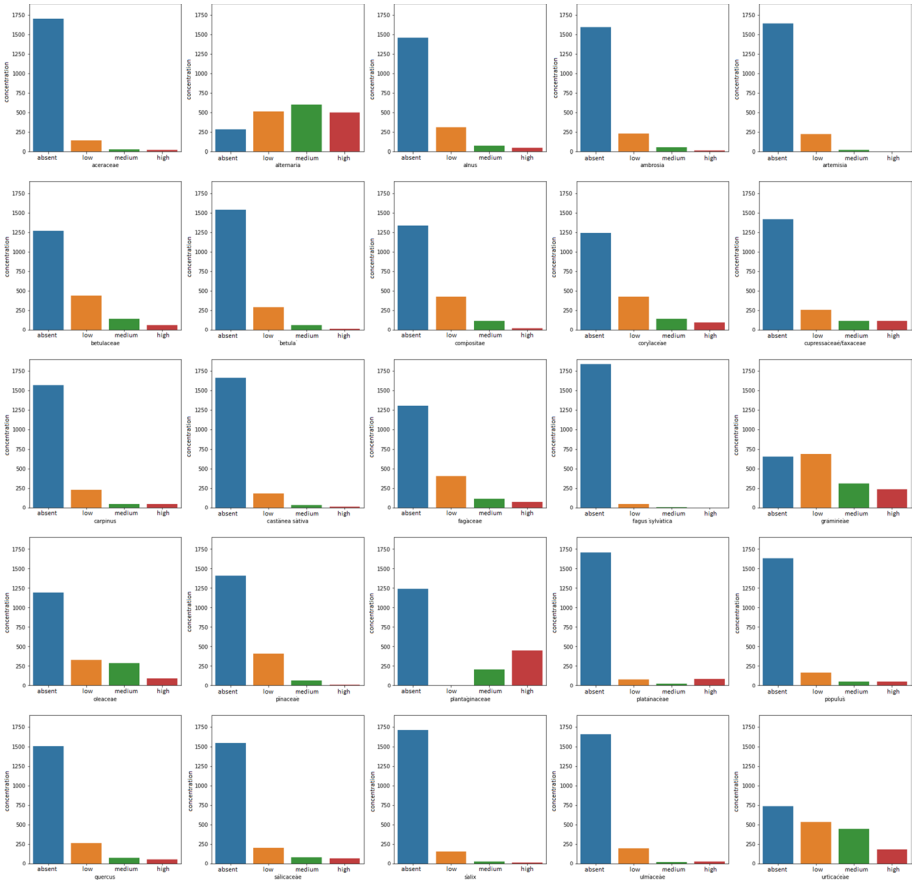


Fig. 18 Visualization of frequencies for each pollen family

Table 6 Anomaly weight for each concentration value of the precipitation attribute

Rainfall intensity	mm/6 h	mm/12 h	mm/24 h
Weak	0.00	0.00	0.00
Moderate	0.34	0.34	0.34
Strong	0.67	0.67	0.67
Very strong	1.00	1.00	1.00

We note that our approach does not necessarily rely on the availability of reference tables for all features. To deal with this issue, we assign an anomaly weight of 0 to features with missing reference tables. By doing so, our cumulative approach used for top-k anomaly level curves is still able to distinguish the severity of different anomalies using the available information for a subset of the features.

Figure 19 shows the trend of the anomaly level for each instance identified as anomalous. Let us consider the curve characterized by the highest anomaly level (green), which identifies the day characterized by the highest amount of anomalous values with

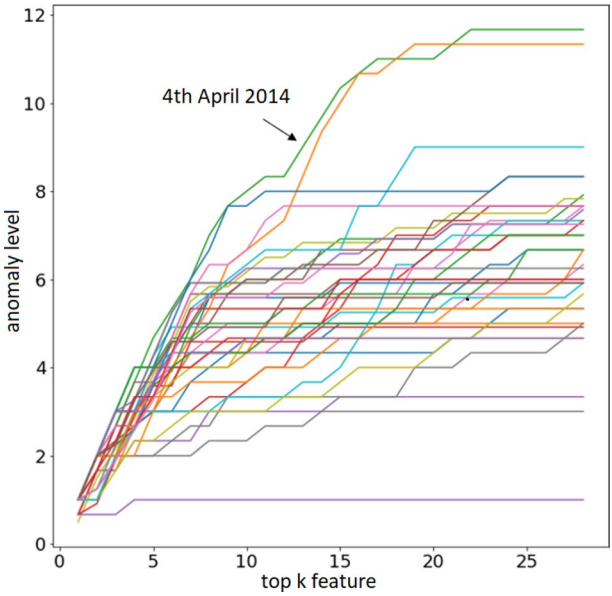


Fig. 19 Anomaly level curves for each considered day. Each curve represents a daily instance detected as anomaly, described through the top-ranked features and the anomaly level calculated from the ground truth

Table 7 Feature ranking for the most anomalous instance ($a\_lv@10 = 8.01$)

Top ranked position	Feature name	Anomaly weight	Anomaly level
1	Urticaceae	1.00	a_lv@1 = 1.00
2	Salix	0.67	a_lv@2 = 1.67
3	Cupressaceae/Taxaceae	1.00	a_lv@3 = 2.67
4	Fagaceae	1.00	a_lv@4 = 3.67
5	Quercus	1.00	a_lv@5 = 4.67
6	Pinaceae	0.67	a_lv@6 = 5.34
7	Corylaceae	0.67	a_lv@7 = 6.01
8	Platanaceae	1.00	a_lv@8 = 7.01
9	Betulaceae	0.67	a_lv@9 = 7.68
10	Fagus Sylvatica	0.33	a_lv@10 = 8.01

respect to other days, considering the descriptive features. The day identified is 4 April 2014, as indicated in Fig. 19.

Our objective is to understand whether the feature ranking returned by our method is consistent with the anomalous values recorded during that day. Table 7 describes the most anomalous instance in terms of anomaly level among the anomalous instances detected by the proposed method.

The features at the top in the feature ranking describe pollen families which, as expected, are characterized by high concentrations during the spring season. This approach helps us to discriminate among different anomalous scenarios, and devoting more attention to more anomalous situations. We can observe that the computation of the anomaly level assumes independence among features and, as mentioned above, the existence of a small set of reference tables for the features under analysis.

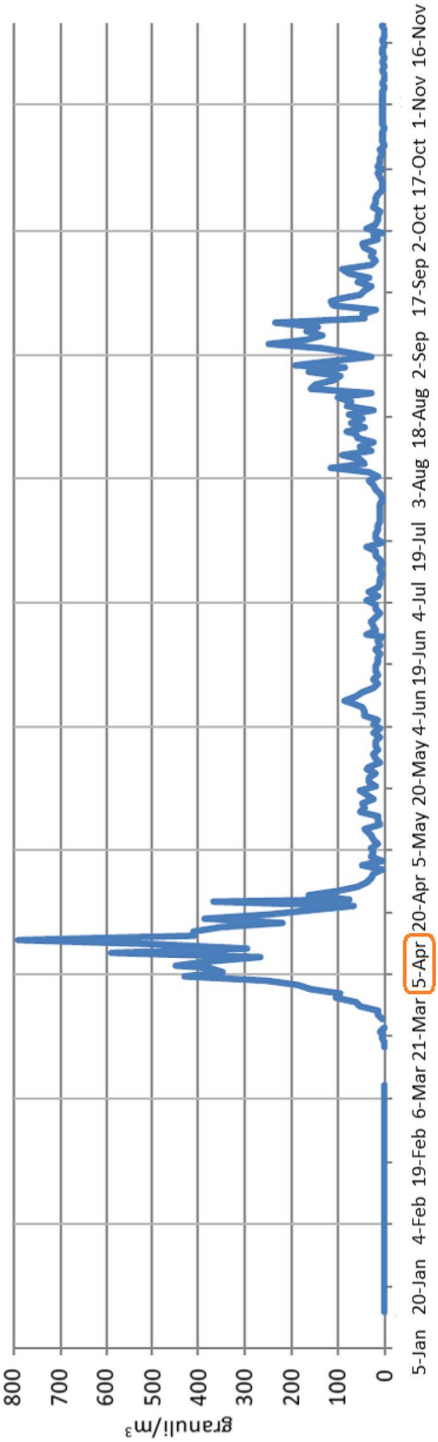


Fig. 20 Average daily pollen concentration of Urticaceae in terms of granules per cubic meter (gr/m^3), the year 2014

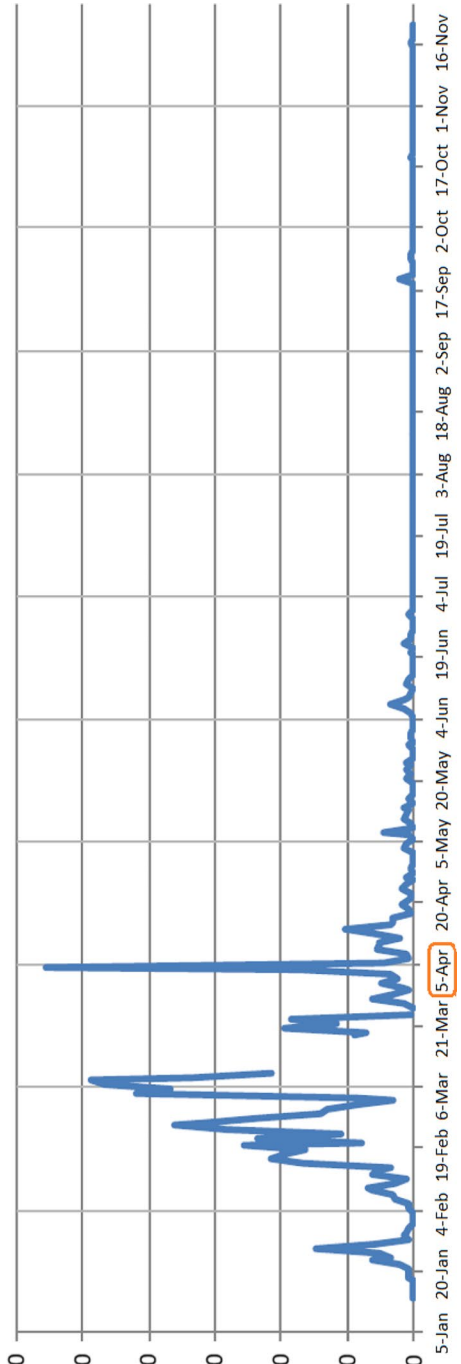


Fig. 21 Average daily pollen concentration of Cupressaceae/Taxaceae in terms of granules per cubic meter (gr/m^3), the year 2014

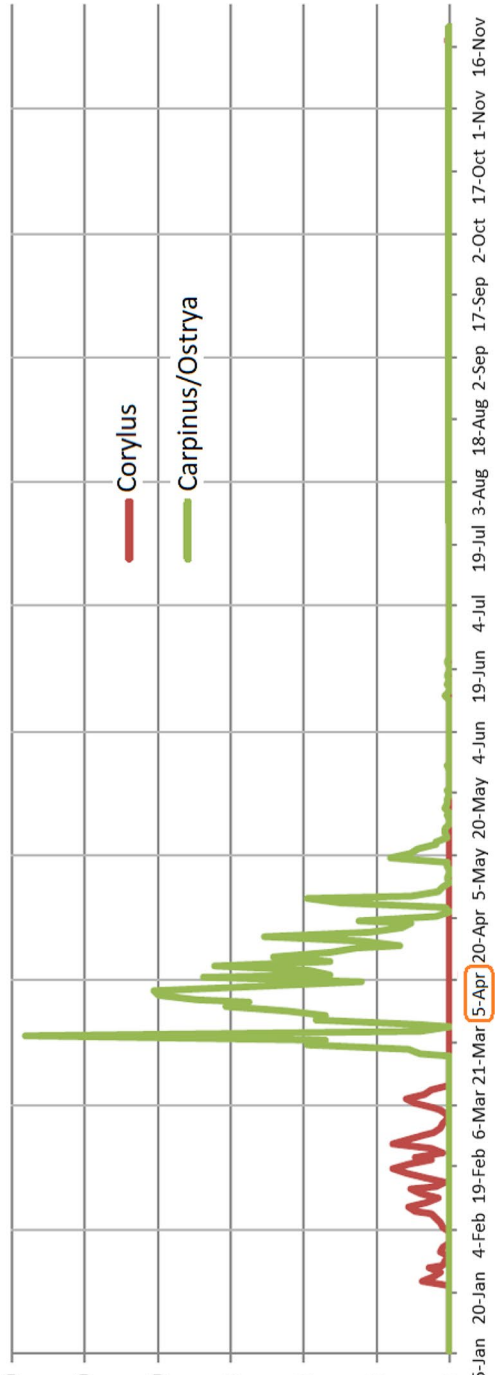


Fig. 22 Average daily pollen concentration of Corylaceae - Corylus and Carpinus/Ostrya in terms of granules per cubic meter (gr/m^3), the year 2014

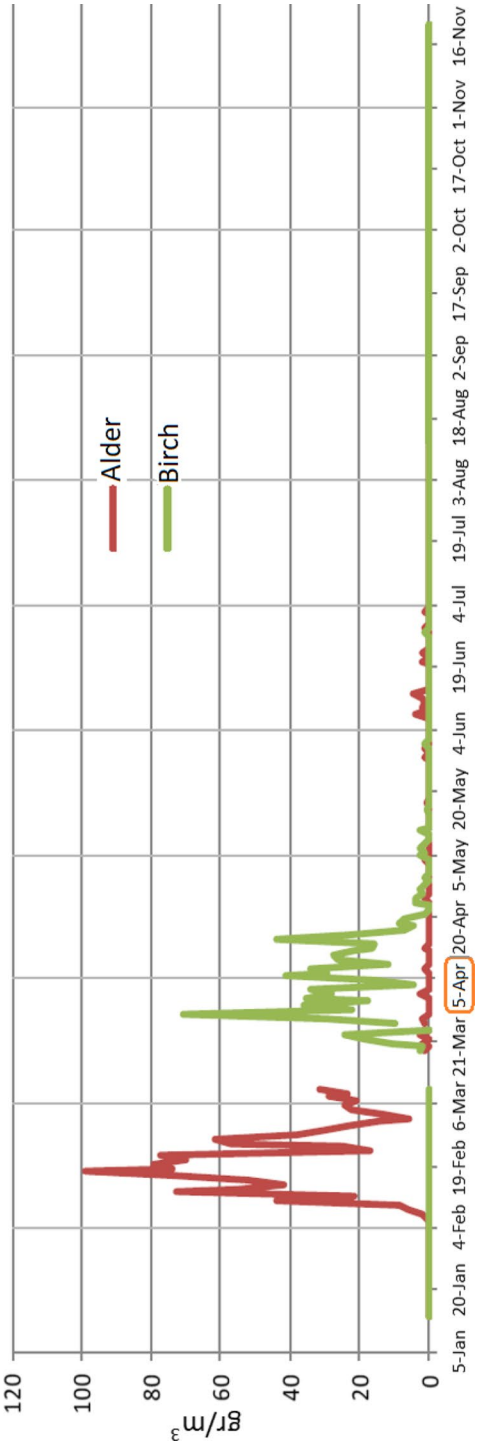


Fig. 23 Average daily pollen concentration of Betulaceae - Alder and Birch in terms of granules per cubic meter (gr/m^3), the year 2014

Figures 20, 21, 22, and 23 show the annual concentrations of pollens Urticaceae, Cupressaceae / Taxaceae, Corylaceae, and Betulaceae relative to the year 2014. These plots have been acquired from the ARPA report on allergenic fungal pollens and spores of Veneto, the Italian region[10]. From these results, it is clearly visible that high concentrations of the described pollens occurred simultaneously on 4 April 2014. Furthermore, considering that the plots are represented using different scales, the feature ranking associated with 4 April 2014 appears consistent with the trend assumed by the plots in that period. Overall, these results show that our method is able to discriminate anomalies based on their level of severity, and to present them in a visual and understandable manner for end users.

Author contributions The authors contributed equally to this work.

Funding Open access funding provided by Università degli Studi di Bari Aldo Moro within the CRUI-CARE Agreement. The authors acknowledge the support of the European Commission through the H2020 project "IMPETUS-Intelligent Management of Processes, Ethics and Technology for Urban Safety" (Grant n. 883286). Dr. Paolo Mignone acknowledges the support of Apulia Region through the project "Metodi per l'ottimizzazione delle reti di distribuzione di energia e per la pianificazione di interventi manutentivi ed evolutivi" (CUP H94I20000410008, Grant n. 7EDD092A) in the context of "Research for Innovation—REFIN".

Data availability Data and materials used in this work are available at: http://www.di.uniba.it/~mignone/systems/distributed_explainable_anomaly_detection/index.html.

Code availability The software implemented and used in this work is available at: http://www.di.uniba.it/~mignone/systems/distributed_explainable_anomaly_detection/Software.7z.

Declarations

Conflict of interest The authors declare that they have no conflict of interests.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Chandola, V., Banerjee, A., & Kumar, V. (2007). Outlier detection: A survey. *ACM Computing Surveys, 14*, 15.

[10] https://www.arpa.veneto.it/temi-ambientali/pollini/file-e-allegati/rapporto-2014/RAPPORTO_Pollini_2014.pdf

Urvoy, M., & Autrusseau, F. Application of grubbs' test for outliers to the detection of watermarks. In *Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security* (pp. 49–60) (2014).

Rousseeuw, P. J., & Leroy, A. M. Robust regression and outlier detection (2005).

McCallum, A., Freitag, D., & Pereira, F. C. (2000). Maximum entropy Markov models for information extraction and segmentation. In *Icml* (Vol. 17, pp. 591–598).

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., & Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Applications of data mining in computer security* (pp. 77–101). Springer.

Duda, R. O., & Hart, P. E. (2006). *Pattern classification*. John Wiley & Sons.

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation, 6*(2), 181–214.

Hofmeyr, S. A., Forrest, S., & Somayaji, A. (1998). Intrusion detection using sequences of system calls. *Journal of Computer Security, 6*(3), 151–180.

Javitz, H. S., & Valdes, A. (1991). The SRI IDES statistical anomaly detector. In *IEEE Symposium on Security and Privacy* (pp. 316–326).

Desforges, M., Jacob, P., & Cooper, J. (1998). Applications of probability density estimation to the detection of abnormal conditions in engineering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 212*(8), 687–703.

Lee, W., & Xiang, D. (2001). Information-theoretic measures for anomaly detection. In *Proceedings 2001 IEEE Symposium on Security and Privacy, S &P 2001* (pp. 130–143).

Arning, A., Agrawal, R., & Raghavan, P. (1996). A linear method for deviation detection in large databases. In: A. Press (Ed.), *KDD'96* (pp. 164–169).

Li, M., & Vitányi, P. M. B. (1993). *An introduction to Kolmogorov complexity and its applications*. Springer.

He, Z., Deng, S., & Xu, X. (2005). An optimization model for outlier detection in categorical data. In D.-S. Huang, X.-P. Zhang, & G.-B. Huang (Eds.), *Advances in intelligent computing* (pp. 400–409). Springer.

Noble, C. C., & Cook, D. J. (2003). Graph-based anomaly detection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '03* (pp. 631–636). Association for Computing Machinery. https://doi.org/10.1145/956750.956831

Hale, W. T., Wilhelm, M., Palmer, K. A., Stuber, M. D., & Bollas, G. M. (2019). Semi-infinite programming for global guarantees of robust fault detection and isolation in safety-critical systems. *Computers & Chemical Engineering, 126*, 218–230. https://doi.org/10.1016/j.compchemeng.2019.04.007

Lebichot, B., Paldino, G. M., Siblini, W., He-Guelton, L., Oblé, F., & Bontempi, G. (2021). Incremental learning strategies for credit cards fraud detection. *International Journal of Data Science and Analytics, 12*(2), 165–174. https://doi.org/10.1007/s41060-021-00258-0

Aldweesh, A., Derhab, A., & Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based System, 189*, 105124. https://doi.org/10.1016/j.knosys.2019.105124

Zhang, M., Li, T., Yu, Y., Li, Y., Hui, P., & Zheng, Y. (2022). Urban anomaly analytics: Description, detection, and prediction. *IEEE Transactions on Big Data, 8*(3), 809–826. https://doi.org/10.1109/TBDATA.2020.2991008

Kou, Y., Lu, C. T., & Chen, D. (2006). Spatial weighted outlier detection. In *Proceedings of the 2006 SIAM International Conference on Data Mining 2006* (pp. 614–618). SIAM.

Shekhar, S., Lu, C. T., & Zhang, P. (2001). Detecting graph-based spatial outliers: Algorithms and applications (a summary of results). In *ACM SIGKDD* (pp. 371–376).

Corizzo, R., Ceci, M., Pio, G., Mignone, P., & Japkowicz, N. (2021). Spatially-aware autoencoders for detecting contextual anomalies in geo-distributed data. In C. Soares & L. Torgo (Eds.), *Discovery science* (pp. 461–471). Springer.

Wang, G., Zhan, Y., Wang, X., Song, M., & Nahrstedt, K. (2022). Hierarchical semi-supervised contrastive learning for contamination-resistant anomaly detection (Vol. 13685, pp. 110–128). https://doi.org/10.1007/978-3-031-19806-9_7

Thudumu, S., Branch, P., Jin, J., & Singh, J. J. (2020). A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data, 7*(1), 42. https://doi.org/10.1186/s40537-020-00320-x

Reddy Shabad, P. K., Alrashide, A., & Mohammed, O. (2021). Anomaly detection in smart grids using machine learning. In *IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society* (pp. 1–8). https://doi.org/10.1109/IECON48115.2021.9589851

Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., & Platt, J. C. Support vector method for novelty detection. In *Advances in neural information processing systems* (pp. 582–588) (2000)

Liu, F. T., Ting, K. M., & Zhou, Z. H. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* pp. 413–422. IEEE.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data, 2*(1), 1–21.

Sakurada, M., & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *MLSDA 2014* (pp. 4–11).

Zhou, C., & Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In *ACM SIGKDD 2017* (pp. 665–674).

Chong, Y. S., & Tay, Y. H. (2017). Abnormal event detection in videos using spatiotemporal autoencoder. In *International Symposium on Neural Networks* (pp. 189–196). Springer.

Kriegel, H., Schubert, M., & Zimek, A. (2008). Angle-based outlier detection in high-dimensional data (pp. 444–452). https://doi.org/10.1145/1401890.1401946

Pham, N., & Pagh, R. (2012). A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data (pp. 877–885). https://doi.org/10.1145/2339530.2339669

Jahromi, A. F., Hajiloei, M., Dehghani, Y., & Lahoninezhad, S. (2022). Improved subspace-based and angle-based outlier detections for fuzzy datasets with a real case study. *Journal of Intelligent & Fuzzy Systems, 42*(6), 5471–5481. https://doi.org/10.3233/JIFS-211955

Li, Z., Zhao, Y., Botta, N., Ionescu, C., & Hu, X. (2020). Copod: Copula-based outlier detection. In *2020 IEEE International Conference on Data Mining (ICDM)* (pp. 1118–1123). https://doi.org/10.1109/ICDM50108.2020.00135

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE, 78*(9), 1464–1480. https://doi.org/10.1109/5.58325

Qu, X., Yang, L., Guo, K., Ma, L., Sun, M., Ke, M., & Li, M. (2021). A survey on the development of self-organizing maps for unsupervised intrusion detection. *Mobile Networks and Applications, 26*(2), 808–829.

Malondkar, A., Corizzo, R., Kiringa, I., Ceci, M., & Japkowicz, N. (2018). Spark-GHSOM: Growing hierarchical self-organizing map for large scale mixed attribute datasets. *Information Sciences*. https://doi.org/10.1016/j.ins.2018.12.007

Anscombe, F. J. (1960). Rejection of outliers. *Technometrics, 2*(2), 123–146.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*. https://doi.org/10.1145/1541880.1541882

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning, 2*(1), 1–127. https://doi.org/10.1561/2200000006

Kashyap, R. V. (2022). A survey of deep learning optimizers-first and second order methods. *CoRR*arXiv:2211.15596

Kingma, D. P., & Ba, J. Adam: A method for stochastic optimization. In Y. Bengio, Y. LeCun (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, 7–9 May, 2015, Conference Track Proceedings*.

Draxler, F., Veschgini, K., Salmhofer, M., & Hamprecht, F. A. Essentially no barriers in neural network energy landscape. In J. G. Dy, A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, 10–15 July, 2018. Proceedings of Machine Learning Research*. PMLR.

Muñoz, A., & Muruzábal, J. (1998). Self-organizing maps for outlier detection. *Neurocomputing, 18*(1), 33–60. https://doi.org/10.1016/S0925-2312(97)00068-4

Palomo, E. J., Ortiz-de-Lazcano-Lobato, J. M., Domínguez, E., & Luque, R. M. (2010). An anomaly detection system using a ghsom-1. In *The 2010 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–7). https://doi.org/10.1109/IJCNN.2010.5596967

Ippoliti, D., & Zhou, X. (2012). A-ghsom: An adaptive growing hierarchical self organizing map for network anomaly detection. *Journal of Parallel and Distributed Computing, 72*(12), 1576–1590. https://doi.org/10.1016/j.jpdc.2012.09.004

Borges Hink, R. C., Beaver, J. M., Buckner, M. A., Morris, T., Adhikari, U., & Pan, S. (2014). Machine learning for power system disturbance and cyber-attack discrimination. In *2014 7th International Symposium on Resilient Control Systems (ISRCS)* (pp. 1–8). https://doi.org/10.1109/ISRCS.2014.6900095

Pan, S., Morris, T., & Adhikari, U. (2015). Classification of disturbances and cyber-attacks in power systems using heterogeneous time-synchronized data. *IEEE Transactions on Industrial Informatics, 11*(3), 650–662. https://doi.org/10.1109/TII.2015.2420951

Pan, S., Morris, T., & Adhikari, U. (2015). Developing a hybrid intrusion detection system using data mining for power systems. *IEEE Transactions on Smart Grid, 6*(6), 3104–3113. https://doi.org/10.1109/TSG.2015.2409775

Pan, S., Morris, T. H., & Adhikari, U. (2015). A specification-based intrusion detection framework for cyber-physical environment in electric power system. *International Journal of Network Security, 17*, 174–188.

Shin, H. K., Lee, W., Yun, J. H., & Kim, H. (2020). HAI 1.0: HIL-based augmented ics security dataset. USENIX Association. https://doi.org/10.5555/3485754.3485755

Himeur, Y., Alsalemi, A., Bensaali, F., & Amira, A. (2020). Building power consumption datasets: Survey, taxonomy and future directions. *Energy and Buildings, 227*, 110404. https://doi.org/10.1016/j.enbuild.2020.110404

Su, B., Zhou, Z., & Chen, H. (2023). Pvel-ad: A large-scale open-world dataset for photovoltaic cell anomaly detection. *IEEE Transactions on Industrial Informatics, 19*(1), 404–413. https://doi.org/10.1109/TII.2022.3162846

De Benedetti, M., Leonardi, F., Messina, F., Santoro, C., & Vasilakos, A. (2018). Anomaly detection and predictive maintenance for photovoltaic systems. *Neurocomputing, 310*, 59–68. https://doi.org/10.1016/j.neucom.2018.05.017

Malki, A., Atlam, E.-S., & Gad, I. (2022). Machine learning approach of detecting anomalies and forecasting time-series of IoT devices. *Alexandria Engineering Journal, 61*(11), 8973–8986. https://doi.org/10.1016/j.aej.2022.02.038

Takiddin, A., Ismail, M., Zafar, U., & Serpedin, E. (2022). Deep autoencoder-based anomaly detection of electricity theft cyberattacks in smart grids. *IEEE Systems Journal, 16*(3), 4106–4117. https://doi.org/10.1109/JSYST.2021.3136683

Ceci, M., Corizzo, R., Japkowicz, N., Mignone, P., & Pio, G. (2020). Echad: Embedding-based change detection from multivariate time series in smart grids. *IEEE Access, 8*, 156053–156066.

Himeur, Y., Ghanem, K., Alsalemi, A., Bensaali, F., & Amira, A. (2021). Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives. *Applied Energy, 287*, 116601. https://doi.org/10.1016/j.apenergy.2021.116601

Riveiro, M., Lebram, M., & Elmer, M. (2017). Anomaly detection for road traffic: A visual analytics framework. *IEEE Transactions on Intelligent Transportation Systems, 18*(8), 2260–2270. https://doi.org/10.1109/TITS.2017.2675710

Kraiman, J. B., Arouh, S. L., & Webb, M. L. (2002). Automated anomaly detection processor. https://doi.org/10.1117/12.474940

Mignone, P., Malerba, D., & Ceci, M. (2022). Anomaly detection for public transport and air pollution analysis. In *2022 IEEE International Conference on Big Data (Big Data)* (pp. 2867–2874). https://doi.org/10.1109/BigData55660.2022.10020470

Sofuoglu, S. E., & Aviyente, S. (2022). Gloss: Tensor-based anomaly detection in spatiotemporal urban traffic data. *Signal Processing, 192*, 108370. https://doi.org/10.1016/j.sigpro.2021.108370

Zhang, M., Li, T., Shi, H., Li, Y., & Hui, P. (2019). A decomposition approach for urban anomaly detection across spatiotemporal data (pp. 6043–6049). https://doi.org/10.24963/ijcai.2019/837

Stojanova, D., Ceci, M., Appice, A., & Džeroski, S. (2012). Network regression with predictive clustering trees. *Data Mining and Knowledge Discovery, 25*(2), 378–413. https://doi.org/10.1007/s10618-012-0278-6

Tama, B. A., Nkenyereye, L., Islam, S. M. R., & Kwak, K.-S. (2020). An enhanced anomaly detection in web traffic using a stack of classifier ensemble. *IEEE Access, 8*, 24120–24134. https://doi.org/10.1109/ACCESS.2020.2969428

Duan, X., Chen, N., & Xie, Y. (2019). Intelligent detection of large-scale KPI streams anomaly based on transfer learning. In H. Jin, X. Lin, X. Cheng, X. Shi, N. Xiao, & Y. Huang (Eds.), *Big data* (pp. 366–379). Springer.

Zhang, S., Zhong, Z., Li, D., Fan, Q., Sun, Y., Zhu, M., Zhang, Y., Pei, D., Sun, J., Liu, Y., Yang, H., & Zou, Y. (2022). Efficient KPI anomaly detection through transfer learning for large-scale web services. *IEEE Journal on Selected Areas in Communications, 40*(8), 2440–2455. https://doi.org/10.1109/JSAC.2022.3180785

Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y., Chen, J., Wang, Z., & Qiao, H. (2018). Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. In *Proceedings of the 2018 World Wide Web Conference. WWW '18* (pp. 187–196). International World Wide Web Conferences Steering Committee. https://doi.org/10.1145/3178876.3185996

Hagemann, T., & Katsarou, K. (2020). Reconstruction-based anomaly detection for the cloud: A comparison on the yahoo! webscope s5 dataset. In *Proceedings of the 2020 4th International Conference on Cloud and Big Data Computing. ICCBDC '20* (pp. 68–75). Association for Computing Machinery. https://doi.org/10.1145/3416921.3416934

Yahoo! webscope dataset ydata-labeled-time-series-anomalies-v1_0. https://research.yahoo.com

Dittenbach, M., Merkl, D., & Rauber, A. (2000). The growing hierarchical self-organizing map. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium* (Vol. 6, pp. 15–19). IEEE.

Chan, A., & Pampalk, E. (2002). Growing hierarchical self organising map (ghsom) toolbox: Visualisations and enhancements. In *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02* (Vol. 5, pp. 2537–25415). https://doi.org/10.1109/ICONIP.2002.1201952

Gama, J. (2010). Knowledge discovery from data streams.

Beggel, L., Pfeiffer, M., & Bischl, B. Robust anomaly detection in images using adversarial autoencoders. In *ECMLPKDD 2019* (pp. 206–222). Springer.

Zhao, Y., Nasrullah, Z., & Li, Z. (2019). Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research, 20*(96), 1–7.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures (pp. 437–478).

Ceci, M., Corizzo, R., Fumarola, F., Malerba, D., & Rashkovska, A. (2016). Predictive modeling of pv energy production: How to set up the learning task for a better prediction? *IEEE Transactions on Industrial Informatics, 13*(3), 956–966.

Authors and Affiliations

Paolo Mignone[1,2] · Roberto Corizzo[1,3] · Michelangelo Ceci[1,2,4]

✉ Paolo Mignone
paolo.mignone@uniba.it

Roberto Corizzo
rcorizzo@american.edu

Michelangelo Ceci
michelangelo.ceci@uniba.it

[1] Department of Computer Science, University of Bari Aldo Moro, Bari, Italy

[2] Big Data Lab, National Interuniversity Consortium for Informatics (CINI), Rome, Italy

[3] Department of Computer Science, American University, Washington, DC 20016, USA

[4] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia