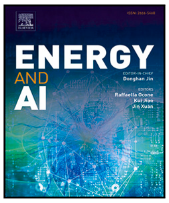
Contents lists available at ScienceDirect

Energy and AI

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



Deep anomaly detection in horizontal axis wind turbines using Graph Convolutional Autoencoders for Multivariate Time series

Eric Stefan Miele [*], Fabrizio Bonacina, Alessandro Corsini

Department of Mechanical and Aerospace Engineering, Sapienza University of Rome, Via Eudossiana 118, Rome, I00184, Italy

A R T I C L E I N F O

Keywords:
Wind turbine
Condition monitoring
Deep anomaly detection
SCADA data
Graph Convolutional Autoencoder
Multivariate Time series
Early fault detection

A B S T R A C T

Wind power is one of the fastest-growing renewable energy sectors instrumental in the ongoing decarbonization process. However, wind turbines are subjected to a wide range of dynamic loads which can cause more frequent failures and downtime periods, leading to ever-increasing attention to effective Condition Monitoring strategies. In this paper, we propose a novel unsupervised deep anomaly detection framework to detect anomalies in wind turbines based on SCADA data. We introduce a promising neural architecture, namely a Graph Convolutional Autoencoder for Multivariate Time series, to model the sensor network as a dynamical functional graph. This structure improves the unsupervised learning capabilities of Autoencoders by considering individual sensor measurements together with the nonlinear correlations existing among signals. On this basis, we developed a deep anomaly detection framework that was validated on 12 failure events occurred during 20 months of operation of four wind turbines. The results show that the proposed framework successfully detects anomalies and anticipates SCADA alarms by outperforming other two recent neural approaches.

1. Introduction

Wind energy is possibly one of the game-changer in future decarbonization scenarios, in view of a plurality of factors. To mention but a few, an incoming generations of multi-MW wind turbines (WTs) [1], the maturity of technology and infrastructures, and the cost competitiveness even in off-shore applications [2–4].

As reported in Hameed et al. [5], the most critical elements in WT energy converters reside in powertrain components, subject to highly irregular loads driven by wind turbulence and extreme weather conditions. As such, the fatigue loading of major structural components can be remarkably greater and peculiar when compared to other rotating machines.

Therefore, second to CAPEX investments are operation and maintenance (O&M) costs, being the most frequent faults on electric and control systems, followed by blades and hydraulic groups [6,7]. In addition, failures (typically in generators and gearboxes) entail high repair and replacement costs and result in long downtimes with significant loss of production. Remedial approaches to face O&M challenges advocate Condition Monitoring (CM) strategies capable of early detection and isolation of incipient faults. CM is a key ingredient to enable condition-based maintenance, able to outperform the on-schedule state-of-the-art, in a view to identify (at early stages) component degradation and limit unnecessary outage of WTs. In mechanical systems, CM is typically based on the acquisition of high-frequency data (e.g., vibrational analysis), possibly processed through a variety of methods (see [8] for a recent review). However, this strategy suffers from several limitations as it demands the installation of additional sensors on WTs and specific data infrastructure, in fact discouraging the implementation [5,9].

On the other hand, modern WTs are integrated with sensor networks as part of Supervisory Control and Data Acquisition (SCADA) systems for monitoring power-train status (e.g. bearing temperature, lube oil sub-system, etc.) with standard practice to record 10-minute averaged values and other statistics of the sensor time series. CM of wind power generation plants through analysis of routinely collected SCADA data is envisaged as a viable mean of forestalling expensive failures and optimizing maintenance through the identification of faults at the earliest possible stage [10,11]. The challenge to operators is, therefore, in identifying the signature of failures within data streams and disambiguating those from other behavioural factors. The strong heterogeneity of signals, together with the loss of high-frequency temporal dependencies caused by the 10-minute averaging, makes the task very demanding [12].

In view of the lack of a comprehensive physical or mathematical model of WT operations, many data-driven methods based on 10-minute SCADA were recently proposed (see [13] for a systematic review). Probabilistic methods fail in modelling the proper temporal dependencies (and dynamics) in sensor networks [14]. For this reason,

* Corresponding author.
E-mail address: ericstefan.miele@uniroma1.it (E.S. Miele).

https://doi.org/10.1016/j.egyai.2022.100145
Received 29 November 2021; Received in revised form 28 January 2022; Accepted 29 January 2022
Available online 12 February 2022

to take into account signals mutual non-linearity and causal dependencies among WT components, most of the methods appeared to date rely on the use of Artificial Neural Networks (NNs) models [15].

In the field of early fault detection, NNs are often employed to learn the normal operating conditions of the system and detect incipient faults by monitoring the real-time deviations from the standard behaviour. The common assumption is that failure occurrences reflect a change of correlation among signals, causing a high multivariate reconstruction error.

To this end, several neural architectures have been proposed in order to capture anomalous scenarios based on prediction errors, where the neural models regress a target variable on a multivariate input. For example, approaches based on Convolutional NN (CNN) [12] have been developed, together with space–time fusion NNs combining convolutional kernels with recurrent units, such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU), to extract multi-scale spatial and temporal correlations [16–18].

However, as reported in [19,20], the standard convolutional operation of CNNs restricts the model to consider only local spatial structures in the signal time series rather than the general domain of the process. In addition, recurrent NNs for sequenced learning require iterative training, which may suffer from error accumulation, difficult training, and increase in computational costs [19].

As a promising alternative to regression models, Autoencoder (AE) architectures have been recently employed in unsupervised anomaly detection, in view of their ability to extract salient features characteristic of normal operating conditions. Examples of such architectures include deep AEs, denoising AEs, LSTM-based AEs, and CNN-based AEs [21–24].

In this paper, we propose an original unsupervised deep anomaly detection framework which has at its core a neural architecture combining AEs and Graph Convolutional Networks (GCNs). Both AEs and GCNs have recently been employed for traffic forecasting or shape coding of buildings in maps where the graph formulation is intrinsic in the application domain [25–27]. Instead, we propose to adapt the formulation to multivariate time series, by modelling the sensor network as a graph where each node represents a sensor with specific feature vectors extracted from its time series. Owing to its multivariate formulation, we advocate the method to be able to analyse contextual anomalies in sensor networks [24].

In detail, we introduce a Graph Convolutional Autoencoder for Multivariate Time series (MTGCAE), composed by an encoder and a decoder based on GCNs adapted to multivariate time series. By representing the data as graphs, the structural information can be encoded to model the relations among entities and furnish more promising insights underlying the sensor data measurements, outperforming standard CNNs, especially in modelling arbitrarily structured systems like sensor networks [28–30].

To perform anomaly detection, the network is trained to learn the normal behaviour of the system in an unsupervised fashion. By defining local and global indicators based on the model reconstruction errors, the framework triggers warnings after the application of a four-stage threshold method that aims at minimizing false alarms during normal operating conditions. In fact, only significant model errors are considered by filtering individual spikes and transient disturbances, allowing to generate sensor-level warnings that isolate the assembly/sub-assembly mostly involved in the anomaly.

We tested the model on SCADA data gathered from 4 WTs belonging to the same wind farm, with a nominal power of 2 MW each [31]. The results showed that the proposed model can anticipate 10 SCADA log alarms with an average time to failure of about 23 days involving some of the most critical components, without triggering any false alarms. Furthermore, to validate the effectiveness of the model, two recently proposed neural architectures have also been applied to the same dataset, one based on an LSTM AE [23] and one on the combination of CNNs, LSTM cells and attention mechanisms [18]. The comparison confirms that the proposed model outperforms these two approaches in terms of evaluation metrics.

The rest of the paper is organized as follows. In Section 2, we present the proposed MTGCAE neural architecture and in Section 3 we discuss the building blocks of the deep anomaly detection framework. Then, in Section 4, we describe the case study and the obtained results, and, finally, in Section 5 we summarize the present work and draw our conclusions.

2. Neural architecture

In this chapter, we describe Autoencoders and Graph Convolutional Networks, together with the proposed neural architecture, namely a Graph Convolutional Autoencoder for Multivariate Time series, which is formulated as a combination of the first two and adapted for multivariate time series.

2.1. Autoencoder

An Autoencoder (AE) is a neural network trained to learn a compressed representation for a set of data in an unsupervised manner [32]. First, it produces a reduced encoding for the input data. Then, it tries to reconstruct the original input from the reduced encoding. In particular, it aims at learning an identity function under specific constraints, for example with a limited number of neurons in the hidden layers. An AE consists of two parts, namely an Encoder and a Decoder.

The Encoder maps the input $\mathbf{x} \in \mathbb{R}^n$ to a latent space and, by considering a feedforward neural network, the output $\mathbf{h}_{\mathbf{e}}^{(l+1)}$ of the lth layer can be written as:

$$\mathbf{h}_{\mathbf{e}}^{(l+1)} = \sigma(\mathbf{W}_{\mathbf{e}}^{(l)} \cdot \mathbf{h}_{\mathbf{e}}^{(l)}) \tag{1}$$

where $\mathbf{W}_{\mathbf{e}}^{(l)}$ are the trainable weights of the layer and σ is a non-linear activation function. Considering an Encoder with L_e layers, we have that $\mathbf{h}_{\mathbf{e}}^{(0)} = \mathbf{x}$ and that $\mathbf{h}_{\mathbf{e}}^{(L_e)} = \mathbf{h}$, where $\mathbf{h} \in \mathbb{R}^k$ is the compressed version of the input.

The Decoder, instead, maps the compressed representation $\mathbf{h}$ back to its original space and, by considering again a feedforward neural network, we have that:

$$\mathbf{h}_{\mathbf{d}}^{(l+1)} = \sigma(\mathbf{W}_{\mathbf{d}}^{(l)} \cdot \mathbf{h}_{\mathbf{d}}^{(l)}) \tag{2}$$

where $\mathbf{h}_{\mathbf{d}}^{(l+1)}$ is the output of the lth layer, $\mathbf{W}_d$ is the trainable weight matrix and σ is a non-linear activation function. Considering L_d layers, we have that $\mathbf{h}_{\mathbf{d}}^{(0)} = \mathbf{h}$ and that $\mathbf{h}^{(L_d)} = \mathbf{x}'$, where $\mathbf{x}'$ is the reconstruction of the input vector $\mathbf{x}$

Since the goal of an AE is to reconstruct the input as accurately as possible (ideally $\mathbf{x}' = \mathbf{x}$), it is trained by minimizing the reconstruction error $\mathcal{L}(\mathbf{x}', \mathbf{x}) = \|\mathbf{x}' - \mathbf{x}\|$, also referred as loss function, through the backpropagation algorithm [33].

It is important to notice that, based on the specific application, other neural architectures different from feedforward NNs can also be considered as Encoder or Decoder with an arbitrary number of hidden layers.

2.2. Graph convolutional network

A Graph Convolutional Network (GCN) is a neural network designed to work directly on graphs and leverage their structural information [34]. The input of a GCN is a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ and $\mathbf{E}$ are the set of nodes and edges respectively. $\mathbf{V}$ is represented as an $N \times F$ feature matrix $\mathbf{X}$, composed by the feature vectors of length F associated with each of the N nodes. The structural information of the graph enclosed in $\mathbf{E}$ is, instead, represented as an $N \times N$ adjacency matrix $\mathbf{A}$. GCNs produce a node-level output $\mathbf{O}$ in the form of an $N \times K$ matrix, where K is the number of output features computed for each node.

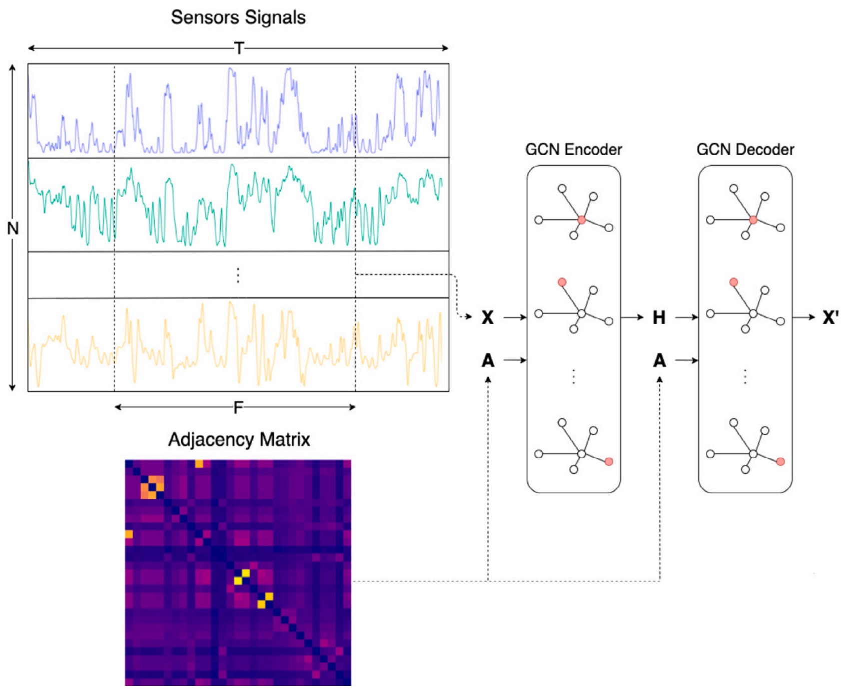


Fig. 1. Detail of the proposed MTGCAE neural architecture. Starting from N sensor signals, the adjacency matrix $\mathbf{A}$ is computed through MI and a sliding window $\mathbf{X}$ of length F is extracted as input for the GCN Encoder. The compressed output representation $\mathbf{H}$, together with the adjacency matrix $\mathbf{A}$, are used by the GCN Decoder to produce a reconstruction of the input signals $\mathbf{X}'$.

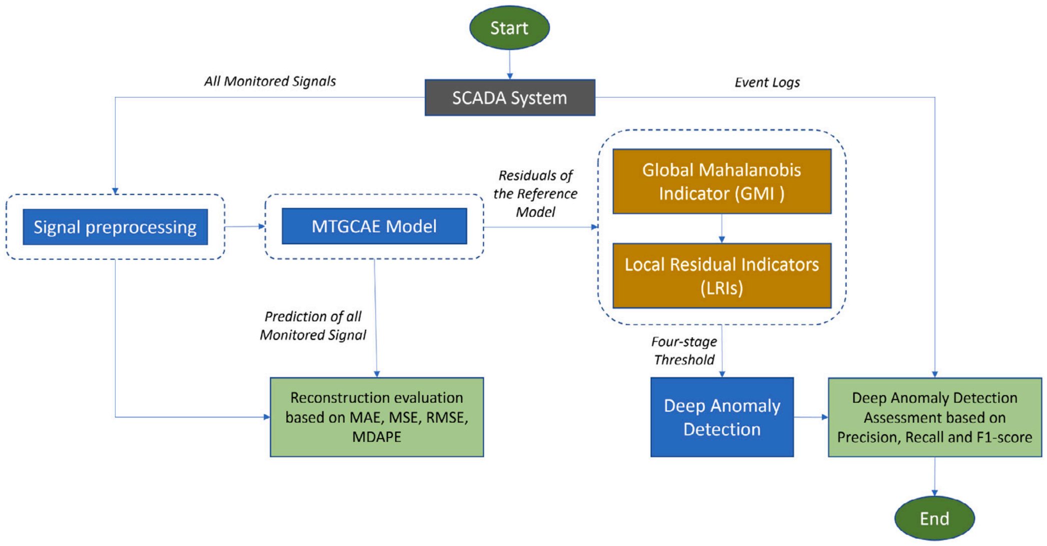

Start
All Monitored Signals
SCADA System
Event Logs
Signal preprocessing
MTGCAE Model
Residuals of the Reference Model
Global Mahalanobis Indicator (GMI)
Local Residual Indicators (LRIs)
Prediction of all Monitored Signal
Four-stage Threshold
Reconstruction evaluation based on MAE, MSE, RMSE, MDAPE
Deep Anomaly Detection
Deep Anomaly Detection Assessment based on Precision, Recall and F1-score
End

Fig. 2. Diagram of the proposed framework for deep anomaly detection.

This matrix can be written as function of the feature matrix $\mathbf{X}$ and the adjacency matrix $\mathbf{A}$ as:

$$\mathbf{O} = f(\mathbf{X}, \mathbf{A}) = \sigma(\mathbf{AXW}) \tag{3}$$

where $\mathbf{W}$ is a $F \times K$ trainable weight matrix.

When multiplying by $\mathbf{A}$, for each node a weighted sum is computed between the feature vectors of all neighbouring nodes excluding itself. For this reason self-loops are added by defining $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, where $\mathbf{I}$ is the $N \times N$ identity matrix. Since $\mathbf{A}$ is typically not normalized, the matrix multiplications defined in Eq. (3) can cause a change of scale in the feature vectors. To prevent numerical instabilities and vanishing/exploding gradients, the adjacency matrix can be normalized by computing $\mathbf{D}^{\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}}$, where $\mathbf{D}$ is the diagonal node degree matrix defined as $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$.

Considering the previous adjustments, the output of a GCN layer can be rewritten as:

$$\mathbf{O} = f(\mathbf{X}, \mathbf{A}) = \sigma(\mathbf{D}^{\frac{1}{2}}\hat{\mathbf{A}}\mathbf{D}^{\frac{1}{2}}\mathbf{XW}) \tag{4}$$

It is possible to define a multi-layer GCN by feeding the output feature matrix of a layer together with the adjacency matrix $\mathbf{A}$ as input for the next layer. Generally, GCNs are trained by minimizing a loss function $\mathcal{L}$ defined between its output $\mathbf{O}$ and the expected output $\bar{\mathbf{O}}$ by means of the backpropagation algorithm.

2.3. Graph convolutional autoencoder for multivariate time series (MTGCAE)

We propose a neural architecture based on the combination of GCNs and AEs, namely a Graph Convolutional Autoencoder for Multivariate Time series (MTGCAE), in order to exploit the extraction of multi-scale spatial and temporal correlations by encoding data as graphs.

In particular, as shown in Fig. 1, the neural architecture consists of a multi-layer GCN which uses as input a graph representation of the sensor network. Each node in the graph represents one of the N signals and the edges quantify the degree of correlation between pairs of time series. More specifically, as input feature matrix $\mathbf{X}$ we consider a sliding window which is a $N \times F$ matrix. In this way, the feature vector $\mathbf{x_i}$ associated with the ith node (i.e. the ith sensor in the SCADA system) is composed by the values of the ith time series in a time window of length F. As for the adjacency matrix $\mathbf{A}$, we define the entry (i, j) as the Mutual Information (MI) between the ith and jth signals or nodes in the graph sensor network [35].

The layers of the GCN are divided into an Encoder and a Decoder, each of which can be composed by multiple layers. Following the typical structure of AEs, the Encoder compresses the input to a latent representation, while the Decoder tries to reconstruct the original input as accurately as possible. Unlike standard AEs, here we adapt the formulation given in Eq. (1) and (2) in order to consider as input a feature matrix instead of a vector.

With reference to the GCN layer formulation in Eq. (4), the output $\mathbf{H_e}^{(l+1)}$ of the lth Encoder layer can be written as function of the previous layer output $\mathbf{H_e}^{(l)}$ and the adjacency matrix $\mathbf{A}$ as:

$$\mathbf{H_e}^{(l+1)} = f(\mathbf{H_e}^{(l)}, \mathbf{A}) = \sigma(\mathbf{D}^{\frac{1}{2}} \hat{\mathbf{A}} \mathbf{D}^{\frac{1}{2}} \mathbf{H_e}^{(l)} \mathbf{W_e}^{(l)}) \tag{5}$$

where $\mathbf{W_e}^{(l)}$ is the trainable weight matrix of the layer and σ is the ReLu activation function [36]. Considering an Encoder with L_e layers, we have that $\mathbf{H_e}^{(0)} = \mathbf{X}$ and that $\mathbf{H_e}^{(L_e)} = \mathbf{H}$, where $\mathbf{H}$ is an $N \times K$ matrix representing the compressed version of the input feature matrix $(K < F)$ after passing through the Encoder.

The Decoder, instead, maps the compressed feature matrix back to its original space and can be formulated in a mirrored way as:

$$\mathbf{H_d}^{(l+1)} = f(\mathbf{H_d}^{(l)}, \mathbf{A}) = \sigma(\mathbf{D}^{\frac{1}{2}} \hat{\mathbf{A}} \mathbf{D}^{\frac{1}{2}} \mathbf{H_d}^{(l)} \mathbf{W_d}^{(l)}) \tag{6}$$

where the subscript d is adopted to discriminate Decoder matrices. Considering L_d layers, we have that $\mathbf{H_d}^{(0)} = \mathbf{H}$ and that $\mathbf{H}^{(L_d)} = \mathbf{X}'$, where $\mathbf{X}'$ is the reconstruction of the input feature matrix $\mathbf{X}$.

Similarly to AEs, the reconstruction error $\mathcal{L} = \|\mathbf{X}' - \mathbf{X}\|_F^2$ is minimized during training using the backpropagation algorithm.

The MTGCAE is trained to reconstruct the sensor signals assuming a reference state. When reapplying it to unseen data during anomalous conditions, we expect the prediction of the trained network to deviate from the actual signals, thus generating residuals.

3. Deep anomaly detection framework

In this chapter, we discuss the main steps of the proposed deep anomaly detection framework shown in Fig. 2. First, we describe the preprocessing of the monitored signals and the proposed MTGCAE model application. Then, we define global and local health indicators together with a four-stage threshold approach for anomaly detection.

3.1. Signal preprocessing

Before the training of the proposed neural architecture, all monitored signals are preprocessed by deleting the records having missing values. In the case of few isolated points, a linear interpolation was applied without introducing distortion in the data [37].

Then, signals presenting high levels of noise were smoothed using the Savitzky–Golay filter [38].

Finally, extreme outliers were filtered using the 5-sigma rule and data was scaled by means of the min–max normalization.

3.2. MTGCAE model

In order to take into account the temporal dependencies in time series, data are explored using sliding windows. Given the data matrix $\mathbf{Z} = (\mathbf{z_1}, \dots, \mathbf{z_i}, \dots, \mathbf{z_T})^T$, where $\mathbf{z_i}$ is the ith N-dimensional multivariate sample, with N the number of signals and T the number of time observations, the ith sliding window is an $N \times F$ matrix defined as:

$$\mathbf{S_i} = (\mathbf{z_{i-F}}, \mathbf{z_{i-F+1}}, \dots, \mathbf{z_i})^T, \quad i = (F, F+1, \dots, N) \tag{7}$$

where F is the length of the window. As a consequence, the dataset $\mathbf{S}$ is structured in successive $w = T - F + 1$ sliding windows:

$$\mathbf{S} = (\mathbf{S}_1, \dots, \mathbf{S}_i, \dots, \mathbf{S}_w)^T \tag{8}$$

To isolate the reference period used to train the proposed MTGCAE model for anomaly detection, as discussed in [24], we employed an unsupervised approach based on the assumption that the hidden layers of deep AEs are capable of capturing intrinsic properties of the majority of the data, representing the normal operation.

In detail, to sample a subset $\mathbf{S_n}$ of normal behaviour windows, we trained the proposed MTGCAE architecture on all windows in $\mathbf{S}$ to learn the most common patterns in the data. Downtimes caused by failures are not captured by the model since rare operating conditions, thus generating high model residuals during their occurrence. In this way, it is possible to isolate outages in an unsupervised manner and exclude them from $\mathbf{S_n}$.

To prevent that fault precursors might be included in the dataset of standard behaviour $\mathbf{S_n}$, we exclude all windows falling in a potentially anomalous time period preceding each downtime consisting of half the Mean Time Between Failure (MTBF), since further away from the last repair.

Once the normal operating conditions are isolated, the MTGCAE model is trained on $\mathbf{S_n}$ and employed for anomaly detection and early fault prediction.

Based on the above, we split $\mathbf{S_n}$ into training $\mathbf{S_{train}}$ and validation $\mathbf{S_{val}}$ both containing only standard behaviour windows, and a testing set $\mathbf{S_{test}}$ which includes both normal and anomalous windows. In particular, the model is trained for E epochs using the Adam optimizer, considering early stopping to avoid overfitting [39].

It is important to notice that the latent representation $\mathbf{H}$ of the MTGCAE directly depends on the parameter K, which should be smaller than the window size F in order to compress the inputs properly, but should also be large enough to capture the most common patterns in the data.

Another crucial parameter is the window size F, which defines the temporal depth of the model. If too small, it will capture only small-scale local patterns and, if too large, it will processes excessively wide time windows and fail to capture their temporal patterns.

3.3. Global Mahalanobis Indicator (GMI) and Local Residual Indicators (LRIs)

The errors of the MTGCAE model are used to specify a rule to warn as early as possible about incipient anomalies. To this end, we define the Global Mahalanobis Indicator (GMI), reflecting the operating status of the whole sensor network, and a Local Residual Indicator (LRI), for each monitored variable.

The GMI is computed as the distance between the model multivariate reconstruction error and the reference multivariate probability distribution of the errors obtained on the validation set $\mathbf{S_{val}}$ by means of Minimum Covariance Determinant [22]. The LRI for each signal is, instead, defined as its specific reconstruction error.

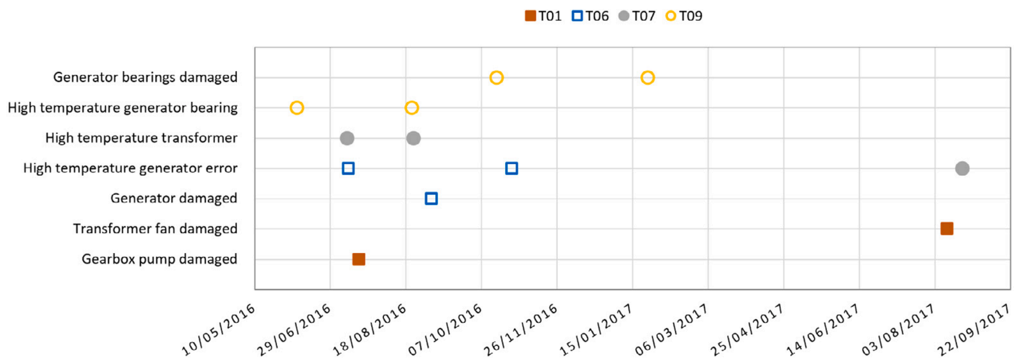


Fig. 3. The time distribution of logs recorded by the SCADA system at sub-assembly and part level for the four WTs.

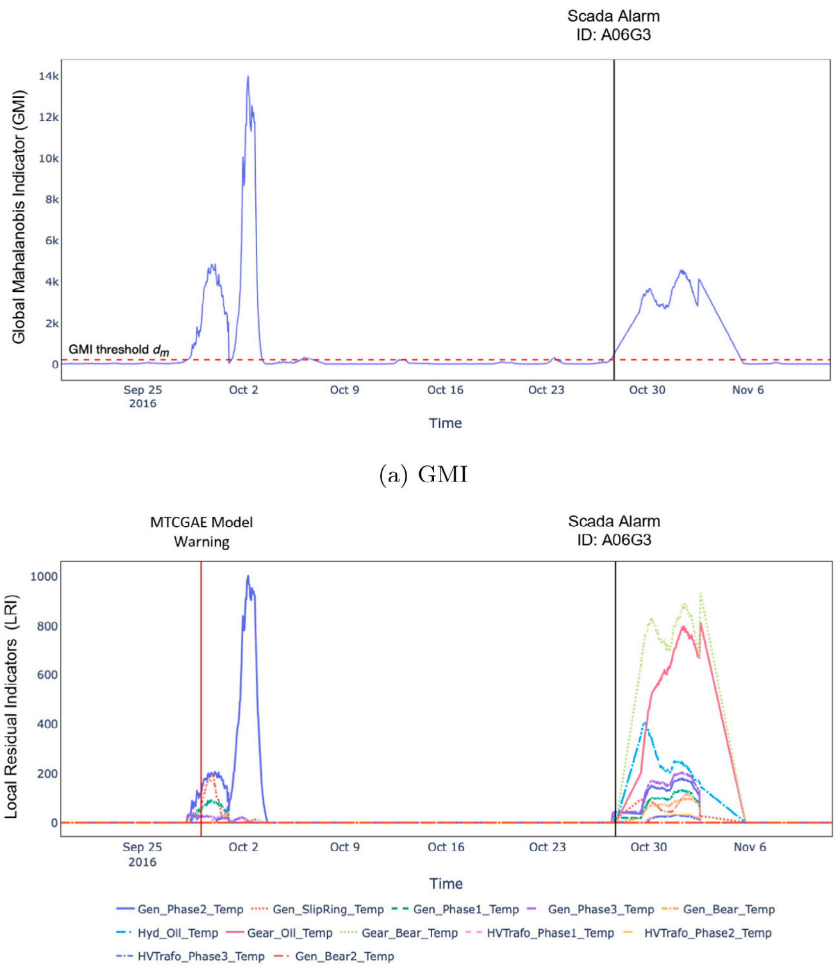


Fig. 4. 4(a) and 4(b) show, respectively, the trend of the Global Mahalanobis Indicator (GMI) and the Local Residual Indicators (LRIs) in correspondence of the alarm ID A06G3. This alarm reports a damage of the T06 generator, detected by SCADA on October 27, 2016 at 16:26. In particular, Fig. 4(a) shows the trend of the global indicator and the threshold d_m applied in the first step of the four-stage filtering method. Fig. 4(b), on the other hand, reports the local residuals that satisfy all the conditions necessary to trigger an alarm based on the four-stage threshold. The vertical red line in the LRI plot represents the MTCGAE model warning associated with possible fault precursors. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

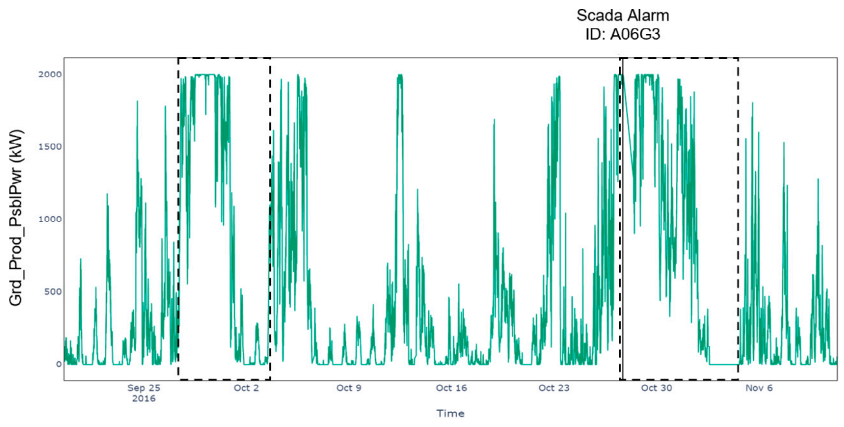
3.4. Four-stage threshold

In order to generate a prompt warning before the occurrence of failures and, at the same time, reduce false alarms during normal operation, a multi-stage threshold is designed for the GMI and LRIs. In particular, as in [14], it evaluates the magnitude of the model errors to detect deviations from standard conditions, but also considers their duration in time to attenuate the effect of individual spikes and transient disturbances. To make the warnings produced by the model more robust to false alarms, we apply a four-stage threshold, considering first the GMI and, then, the LRIs.

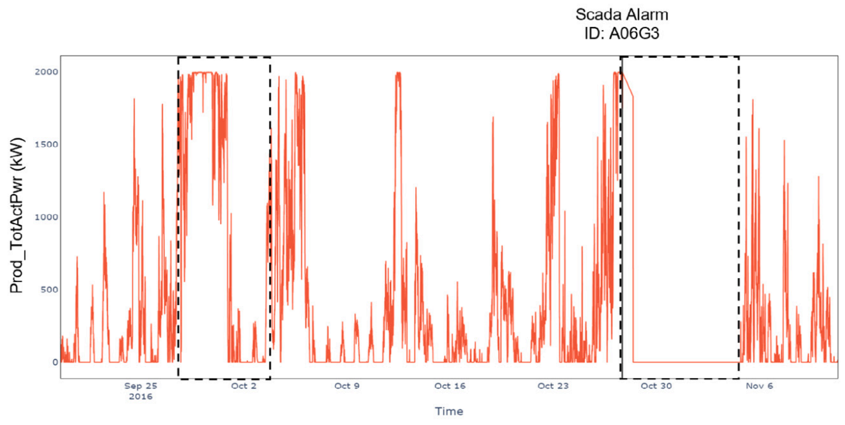
The four sequential filtering steps applied to the local and global indicators are the following:

Two-stage threshold on GMI:

1. filter GMI values below a threshold d_m to consider only significant multivariate reconstruction errors;



(a) Grid Power Request



(b) Total Active Power

Fig. 5. 5(a) and 5(b) show, respectively, the trend of the Grid Power Request and the Total Active Power of the T06 WT in correspondence of the alarm ID A06G3. The dashed boxes detail the periods in which anomalies were detected by the proposed deep anomaly detection framework (see Fig. 4).

2. filter GMI values that have a duration less than F in order to consider only residuals persistent in time for at least the length of the model input sliding window;

Two-stage threshold on LRI:

3. for each signal i, filter LRI values below a threshold d_i to consider only significant reconstruction errors;
4. for each signal i, filter LRI values that have a duration less than F in order to consider only residuals persistent in time for at least the length of the model input sliding window.

When the conditions of all four stages are satisfied in the presented order, the model triggers a warning for the sensor having the highest LRI.

4. Experimental results

In this section, the proposed fault detection framework is validated using the open dataset available at [31]. More details can be found below.

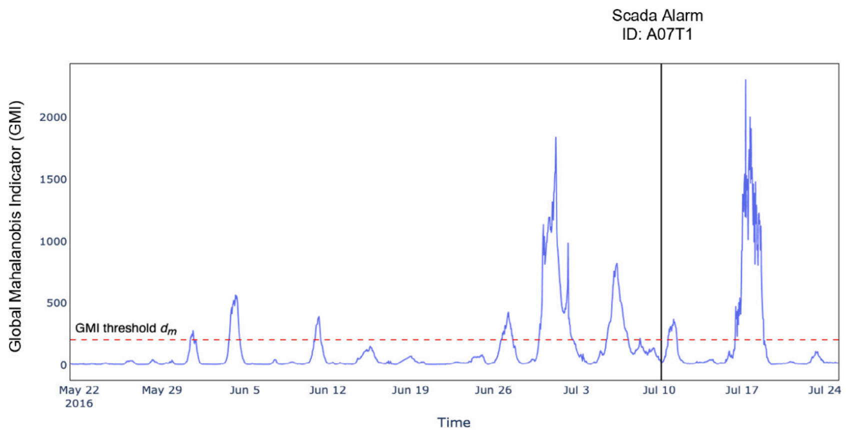
4.1. Dataset description

The data is collected from four WTs belonging to the same wind farm, each having a diameter of 90 m, a maximum rotor speed of 14.9 rpm, and a maximum rated power of 2 MW at a nominal wind speed of 12 m/s. The wind farm is ranked class 2 according to the standard IEC 61400 [40]. The complete description of the technical information of the WTs is given in Table 1.

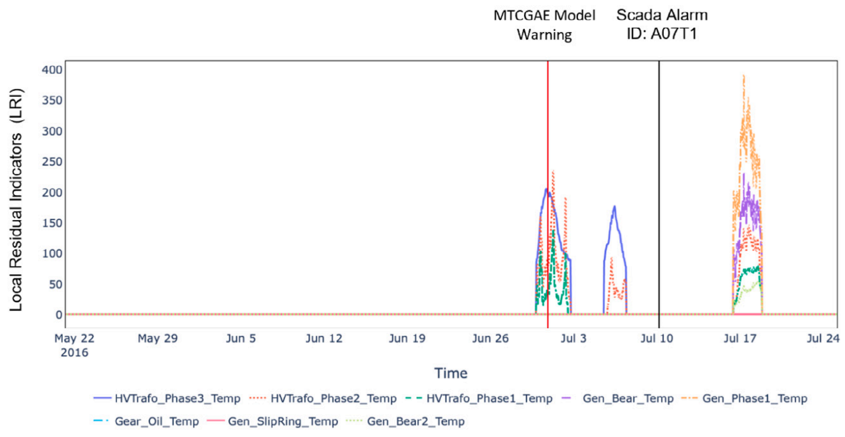
Table 1
Technical information of each turbine.

Rated power (kW)	2000
Cut-in wind speed (m/s)	4
Rated wind speed (m/s)	12
Cut-out wind speed (m/s)	25
Rotor diameter (m)	90
Rotor swept area (m^2)	6362
Number of blades	3
Max rotor speed (rpm)	14.9
Rotor tip speed (m/s)	70
Rotor power density 1 (W/m^2)	314.4
Rotor power density 2 (m^2/kW)	3.2
Gearbox Type	Planetary/spur
Gearbox stages	3
Generator type	Asynchronous
Max generator speed (rpm)	2016
Generator voltage (V)	690
Grid frequency (Hz)	50
Hub height (m)	80

All WTs are equipped with a SCADA system for the monitoring of multiple parameters collected from the main components together with ambient measurements. In particular, for each WT we considered a separate dataset composed by 30 monitored parameters listed in



(a) GMI



(b) LRI

Fig. 6. 6(a) and 6(b) show, respectively, the trend of the Global Mahalanobis Indicator (GMI) and the Local Residual Indicators (LRIs) in correspondence of the alarm ID A07T1. This alarm reports a high temperature anomaly of the T07 transformer detected by the SCADA system on July 10, 2016 at 03:46. In particular, Fig. 6(a) shows the trend of the global indicator and the threshold d_m applied in the first step of the four-stage filtering method. Fig. 6(b), on the other hand, reports the local residuals that satisfy all the conditions necessary to trigger an alarm based on the four-stage threshold. The vertical red line in the LRI plot represents the MTGCAE model warning associated with possible fault precursors.

Table 2. The dataset covers a time span of about 20 months (from January 1, 2016 to September 1, 2017).

In addition to the sensor signals, we considered the event log that includes all alarms recorded by the SCADA system on the four WTs during the reported period. These events include all potential operational risks, which can be seen as anomalies reducing the remaining useful life of components. In fact, the SCADA system supervises the operating status of the wind turbines and protects them from extreme loads. In this way, when a critical signal exceeds predefined operating thresholds, an event is triggered and recorded in the log file.

With reference to the Reliawind turbine taxonomy presented in [41], the event log available in [31] mainly contains the details of the anomalies recorded at the assembly and sub-assembly levels, and only for some alarms at the component/part level. Starting from this, we filtered out all false alarms and minor events that did not lead to repair or replacement actions for the component.

Table 3 lists all the events we considered for this study, detailing the turbine IDs, the assembly/sub-assembly involved, the date and time of the alarm recorded by the SCADA system and the type of action taken by the operators to restore proper operation (repair or replacement). Fig. 3, instead, shows the time distribution of the alarms for each WT over the investigated period.

From the analysis of the logs contained in Table 3 and Fig. 3 it is possible to notice that most of the recorded alarms concern the drive train and power sub-systems. In particular, two repairs were carried out on the T01 turbine, one involving the gearbox and the other the transformer, in response to component level anomalies detected respectively on the gearbox pump (July 18, 2016) and the transformer fan (August 11, 2017).

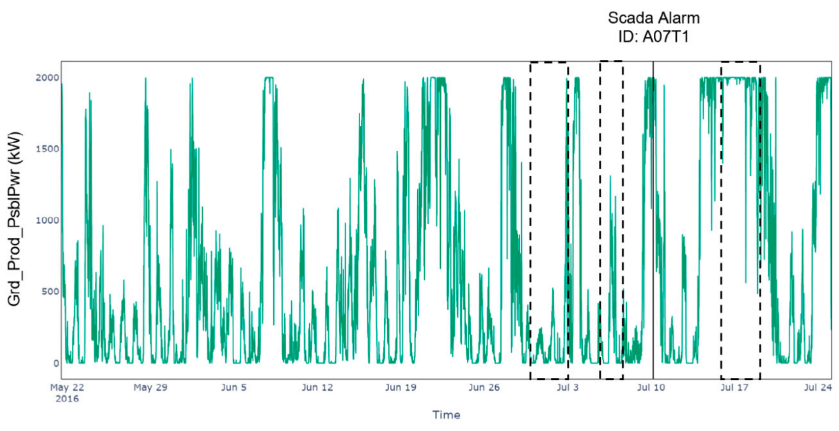
As for the three alarms recorded on the T06 turbine, they refer to generator anomalies that occurred in the period June–November, 2016, and two of these required a full replacement at the assembly level.

The alarms of the T07 turbine, on the other hand, concern two episodes of high temperature recorded in the transformer on July 10, 2016 and August 23, 2016 and an anomaly in the generator that occurred about a year later (August 21, 2017), which required specific repair interventions.

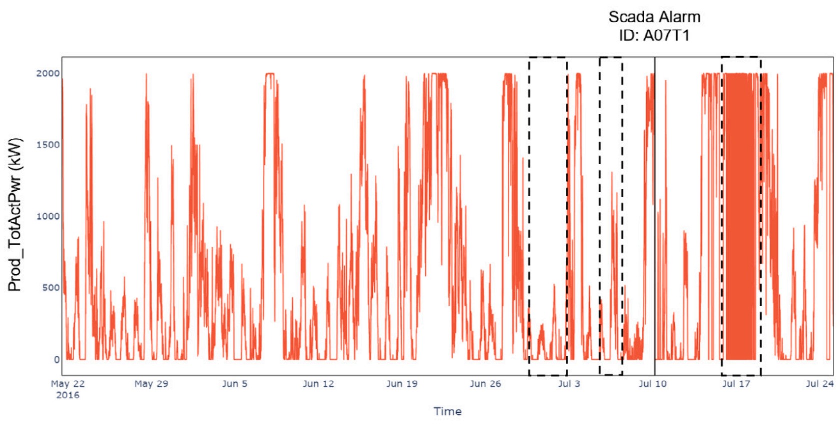
Finally, in turbine T09, four distinct alarms were recorded on the generator bearings, two of which concern high temperature events on June 7, 2016 and August 22, 2016, requiring a repair of the damaged parts. The other two alarms, instead, were triggered on October 17, 2016 and January 25, 2017, and involve major damages that led to the replacement of the components.

4.2. MTGCAE parameter setting

The analysis was carried out for each wind turbine, from the pre-processing and data preparation to the training of the model.



(a) Grid Power Request



(b) Total Active Power

Fig. 7. 7(a) and 7(b) show, respectively, the trend of the Grid Power Request and the Total Active Power of the T07 WT in correspondence of the alarm ID A07T1. The dashed boxes detail the periods in which anomalies were detected by the proposed deep anomaly detection framework (see Fig. 6).

The MTGCAE was trained for $E = 50$ epochs considering early stopping to avoid overfitting. Specifically, we considered one layer for the Encoder having K output features as latent representation and one layer for the Decoder to reconstruct the input data. A single layer was sufficient for the autoencoding process and stacking more layers did not significantly improve the performance of the model.

The parameter K was set to be proportional to F so that the number of neurons scaled with the length of the input sliding window, allowing the model to increase its complexity for larger windows. In particular, we set $K = \frac{F}{6}$ in order to compress the inputs to a lower dimension ($K < F$) and at the same time provide the network with enough neurons to reconstruct the inputs. The sliding window size F was set to 144 (24 h) by means of grid search aimed at minimizing the Mean Absolute Error (MAE) on the validation set. This configuration allowed the model to capture the daily patterns which are also highlighted by the autocorrelation function of signals.

In order to determine the standard operating conditions, we applied the unsupervised approach discussed in Section 3.2 by considering a MTBF of 75 days as reported in [42]. In this way, as confirmed by the event logs, the main failures and time periods preceding outages were excluded from the standard behaviour data which was, then, split into training set $\mathbf{S}_{\mathbf{train}}$ (70%) and validation set $\mathbf{S}_{\mathbf{val}}$ (30%). The test set $\mathbf{S}_{\mathbf{test}}$, instead, was defined by selecting time periods including failure occurrences reported in the log files.

Based on of the reconstruction errors, warnings were triggered by the model according to the four-stage threshold method presented in Section 3.4, allowing to consider only significant residuals both in terms of magnitude and duration. In particular, the GMI threshold d_m was set to the 3rd quantile of the validation set distribution of Mahalanobis distances, and the LRI threshold d_i for the ith signal to the 3rd quantile of its reconstruction error distribution.

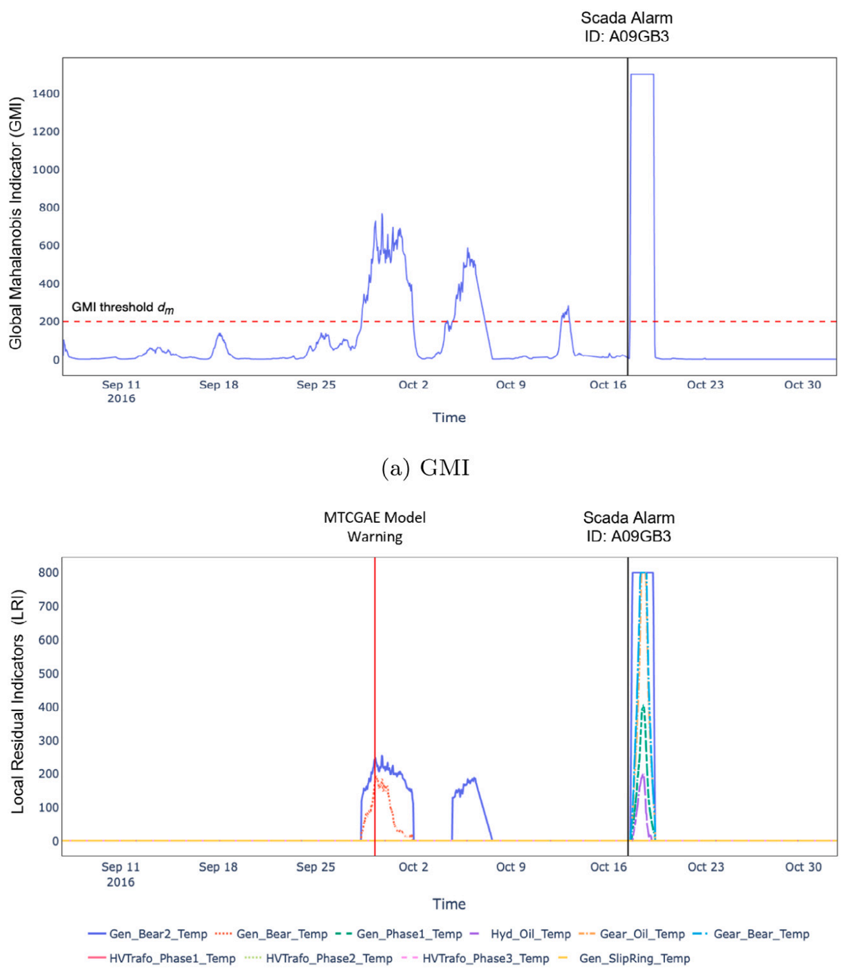
4.3. Reconstruction errors

The MTGCAE model was compared in terms of reconstruction error for all SCADA signals with two other promising neural architectures recently applied for anomaly detection in wind turbines, namely LSTM-AE (introduced in [23]) and CNN–LSTM (introduced in [18]). The first is based on LSTMs and AEs, and the other is a regression model based on the combination of CNNs and LSTMs.

Table 4 shows the scores achieved by the three architectures, by evaluating the Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and the Median Absolute Percentage Error (MDAPE).

Even though MTGCAE performs better, the reconstruction errors of all three models are low, being the highest error 0.085, reached by the RMSE of the LSTM-AE model.

Since the training is performed on standard operating condition data, low errors are expected because all input windows are drawn from the same normal behaviour distribution. High residuals are, instead, expected during anomalous conditions and failures, as confirmed by the next section.



(a) GMI

(b) LRI

Fig. 8. 8(a) and 8(b) show, respectively, the trend of the Global Mahalanobis Indicator (GMI) and the Local Residual Indicators (LRIs) in correspondence of the alarm ID A09GB3. This alarm reports a damage of the T09 generator bearings, detected by the SCADA system on October 17, 2016 at 09:19 (see Table 3). In particular, Fig. 8(a) shows the trend of the global indicator and the threshold d_m applied in the first step of the four-stage filtering method. Fig. 8(b), on the other hand, reports the local residuals that satisfy all the conditions necessary to trigger an alarm based on the four-stage threshold. The vertical red line in the LRI plot represents the MTGCAE model warning associated with possible fault precursors. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.4. Deep anomaly detection results

As a first result of the proposed deep anomaly detection framework, Fig. 4 shows the trend of the GMI and LRIs in the period that goes from September 26 to November 6, 2016, in proximity of the IDA06G3 alarm. This alert notifies a damage on the T06 turbine generator detected by the SCADA system on October 27, 2016 at 16:26 (see Table 3 for details).

From Fig. 4(a), it can be seen that the GMI detects a possible anomaly from September 29 to October 3, 2016, identified by indicator values exceeding the global threshold d_m for at least 24 h (the first two filtering steps of the four-stage threshold discussed in Section 3.4).

Then, looking at Fig. 4(b), we can observe high values of the LRIs in the same period as for the GMI, relative to the three phases of the generator stator windings (*Gen_Phase1_Temp, Gen_Phase2_Temp, Gen_Phase3_Temp*) and also to the temperature measured in the split ring chamber (*Gen_SplitRing_Temp*). Therefore, by applying all the filtering steps provided by the four-stage threshold, the model triggers a warning associated with the sensor having highest LRI, namely the *Gen_Phase1_Temp*. This anomaly, detected on the temperature of the first phase of the generator stator winding, anticipates of about 28 days the SCADA alarm related to the damage of the generator assembly.

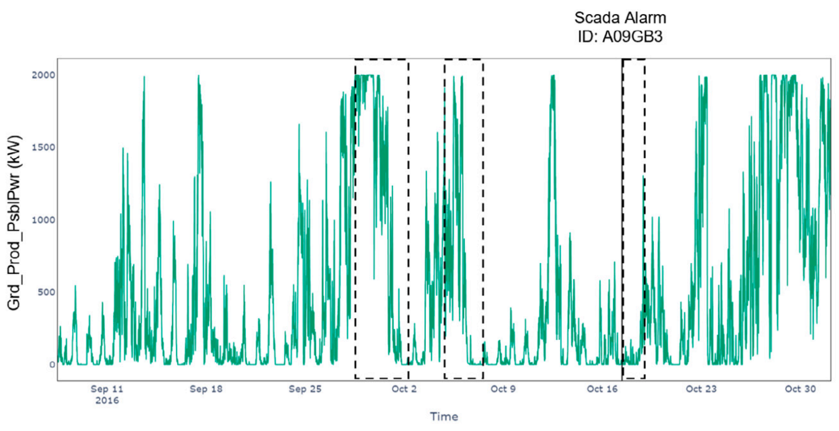
It is interesting to notice that, when comparing the grid power request (Fig. 5) and the total active power produced by the T06 turbine (Fig. 5(b)), no significant mismatch is found in proximity of the precursor (dashed box on the left), thus showing the ability of the model to capture hidden anomalies even when the turbine continues to deliver the power requested by the grid.

On the other hand, looking at the dashed box on the right after the SCADA alarm in Fig. 5(b), a long outage of about six days can be observed, needed to replace the damaged generator. This results in an anomaly in terms of expected power, which also generates significant residuals in the MTGCAE model indicators (see Fig. 4).

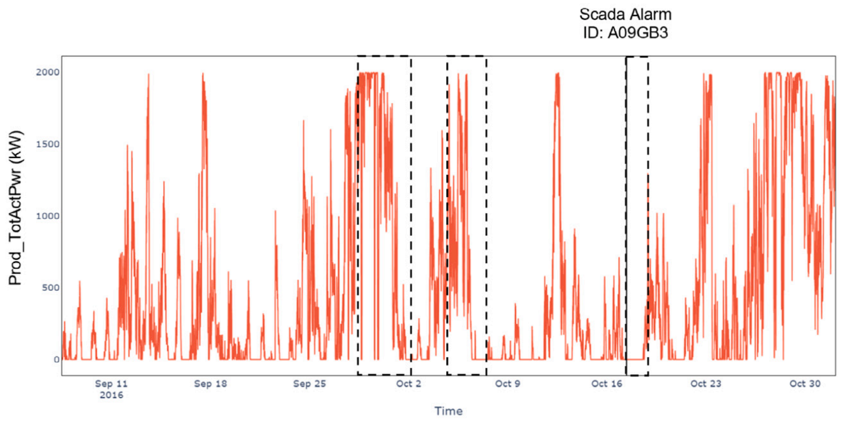
As a second result, Figs. 6(a) and 6(b) show the evolution of the GMI and LRIs during a period of about three months (i.e., from May 22 to July 24, 2016), during which a high temperature anomaly was reported by the SCADA system on July 10, 2016 at 03:46 (alarm ID A07T1 in Table 3).

In this case, after the application of the four-stage threshold to the reconstruction errors of the MTGCAE model, an anomaly on the temperature of the T07 transformer windings (*HVTrafo_Phase1_Temp*, *HVTrafo_Phase2_Temp*, *HVTrafo_Phase3_Temp*) is isolated about 9 days before the SCADA alarm.

As for the previous example, also in this case the model is able to detect an operational anomaly even if the total active power of the T07



(a) Grid Power Request



(b) Total Active Power

Fig. 9. 9(a) and 9(b) show, respectively, the trend of the Grid Power Request and the Total Active Power of the T09 WT in correspondence of the alarm ID A09GB3. The dashed boxes detail the periods in which anomalies were detected by the proposed deep anomaly detection framework (see Fig. 8).

turbine matches the power required by the grid (see the first dashed box on the left in Fig. 7).

About one week after the investigated event, an anomalous behaviour in the power delivered by the turbine T07 is observed for two days (dashed box on the right), during which significant residuals are produced by the deep anomaly detection framework at the assembly level of both the transformer and the generator (see Fig. 6(b)).

As a last application, Fig. 8 details the results in terms of the MTGCAE global and local indicators in the vicinity of a T09 generator bearing damage, reported by the SCADA system on October 17, 2016 at 09:19 and labelled as alarm ID A09GB3 in Table 3.

Looking at the GMI and LRIs shown, respectively, in Fig. 8(a) and Fig. 8(b), the first warning is triggered about 18 days before the SCADA alarm, and is associated with an anomaly isolated by the generator bearing temperature residuals (*Gen_Bear2_Temp*). Also during this anomalous period, turbine T09 seems to be unable to deliver the power required by the grid (see the first dashed box on the left in Fig. 9).

After the SCADA alarm, Fig. 9 shows a period of forced turbine downtime (the rightmost dashed box) required to replace the damaged generator bearings, during which high reconstruction errors are produced by the model.

4.5. Deep anomaly detection assessment

The best performance corresponds to the early detection of the greatest number of anomalies and faults recorded by the SCADA system that required repair or replacement at the assembly/sub-assembly level, with the minimum false alarms and the maximum time advance.

In particular, we considered a discrete event evaluation of the performance where a True Positive (TP) corresponds to triggered model warnings associated to the assembly/sub-assembly that presents a SCADA alarm (with reference to the logs reported in Table 3) in the next time window consisting of $T_f = 4320$ samples (1 month). False positives (FP), on the other hand, are model warnings that are not followed by a SCADA alarm associated with the involved assembly/sub-assembly. Finally, SCADA alarms not anticipated by a model warning within the reference time window are considered as False Negatives (FN).

At this point, we quantified the performance of the model through classification metrics typically used in the field of Machine Learning:

$$\text{Precision } (P) = \frac{\text{TP}}{\text{TP} + FP} \tag{9}$$

$$\text{Recall } (R) = \frac{\text{TP}}{\text{TP} + FN} \tag{10}$$

$$\text{F1-score } (F1) = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + Recall} \tag{11}$$

In addition, we also consider the mean time advance (*Avg Advance*), which represents the time period between the warnings triggered by the model and the reference SCADA alarm.

Based on these metrics, as for the model reconstruction errors discussed in Section 4.5, the deep anomaly detection capabilities of the proposed MTGCAE were validated against the LSTM-AE and the

Table 2
Parameters monitored by WT SCADA system.

Signal ID	Description	Component
Gen_Bear_Temp	Temperature in generator bearing 1 (Non-Drive End)	Generator Bearings
Gen_Bear2_Temp	Temperature in generator bearing 2 (Drive End)	Generator Bearings
Gen_RPM	Generator rpm	Generator
Gen_Phase1_Temp	Temperature inside generator in stator windings phase 1	Generator
Gen_Phase2_Temp	Temperature inside generator in stator windings phase 2	Generator
Gen_Phase3_Temp	Temperature inside generator in stator windings phase 3	Generator
Gen_SlipRing_Temp	Temperature in the split ring chamber	Generator
Hyd_Oil_Temp	Temperature oil in hydraulic group	Hydraulic
Gear_Oil_Temp	Temperature oil in gearbox	Gearbox
Gear_Bear_Temp	Temperature in gearbox bearing on high speed shaft	Gearbox
Nac_Temp	Temperature in nacelle	Nacelle
Nac_Direction	Nacelle direction	Nacelle
Rtr_RPM	Rotor rpm	Rotor
Amb_WindSpeed	Wind speed	Ambient
Amb_WindDir_Relative	Wind relative direction	Ambient
Amb_WindDir_Abs	Wind absolute direction	Ambient
Amb_Temp	Ambient temperature	Ambient
Prod_TotActPwr	Total active power	Production
Prod_TotReactPwr	Total reactive power	Production
Grd_Prod_PsblPwr	Grid Power Request	Grid
HVTrafo_Phase1_Temp	Temperature in HV transformer phase L1	Transformer
HVTrafo_Phase2_Temp	Temperature in HV transformer phase L2	Transformer
HVTrafo_Phase3_Temp	Temperature in HV transformer phase L3	Transformer
Cont_Top_Temp	Temperature in the top nacelle controller	Controller
Cont_Hub_Temp	Temperature in the hub controller	Controller
Cont_VCP_Temp	Temperature on the VCP-board	Controller
Cont_VCP_ChokcoilTemp	Temperature in the choke coils on the VCS-section	Controller
Cont_VCP_WtrTemp	Temperature in the VCS cooling water	Controller
Spin_Temp	Temperature in the nose cone	Spinner
Blds_PitchAngle	Blades pitch angle	Blades

Table 3
Main alarms reported in the maintenance log file available in Ref. [31].

Turbine ID	Alarm ID	Alarm timestamp	Assembly/Sub-assembly	Type of alarm	Type of action	
					Repair	Replacement
T01	A01GX	18/07/2016; 02:10	Gearbox	Gearbox pump damaged	x	
T01	A01T	11/08/2017; 13:14	Transformer	Transformer fan damaged	x	
T06	A06G1	11/07/2016; 19:48	Generator	Generator damaged		x
T06	A06G2	04/09/2016; 08:08	Generator	High temperature generator error	x	
T06	A06G3	27/10/2016; 16:26	Generator	Generator damaged		x
T07	A07T1	10/07/2016; 03:46	Transformer	High temperature transformer	x	
T07	A07T2	23/08/2016; 02:21	Transformer	High temperature transformer	x	
T07	A07G	21/08/2017; 14:47	Generator	Generator damaged	x	
T09	A09GB1	07/06/2016; 16:59	Generator bearings	High temperature generator bearing	x	
T09	A09GB2	22/08/2016; 18:25	Generator bearings	High temperature generator bearing	x	
T09	A09GB3	17/10/2016; 09:19	Generator bearings	Generator bearings damaged		x
T09	A09GB4	25/01/2017; 12:55	Generator bearings	Generator bearings damaged		x

Table 4
The table shows the reconstruction errors in terms of MAE, MSE, RMSE and MDAPE for the proposed MTGCAE model, comparing it with the LSTM-AE and CNN–LSTM architectures.

Model	MAE	MSE	RMSE	MDAPE
MTGCAE	0.038	0.004	0.061	0.058
LSTM-AE	0.053	0.007	0.085	0.08
CNN–LSTM	0.051	0.007	0.082	0.075

Table 5
Results of MTGCAE, LSTM-AE and CNN–LSTM models.

Model	TP	FN	FP	Avg Advance (Time)	P	R	F1
GCAE	10	2	0	23 days, 0:05:42	1.0	0.83	0.91
LSTM-AE	10	2	4	27 days, 12:55:18	0.71	0.83	0.77
CNN–LSTM	9	3	4	28 days, 16:10:20	0.69	0.75	0.72

CNN–LSTM architectures. It is important to notice that the same framework for anomaly detection presented in Section 3 was applied when considering LSTM-AE and CNN–LSTM, in order to make their outputs comparable with MTGCAE.

Table 5 presents the evaluation metrics for the three models on the same test set.

Results shows that MTGCAE is able to detect 10 out of 12 anomalous events without triggering any false alarm. LSTM-AE, while achieving the same TPs, counts 4 FPs, thus reducing the Precision and F1-score. Finally, the CNN–LSTM model only detects 9 events and presents 4 false alarms, producing lower scores also in terms of the Recall metric.

Even though MTGCAE achieves a lower average time advance (23 days) with respect to the other two models, the proposed approach seems to be more reliable in view of the lack of FPs, and robust provided the number of FNs.

5. Conclusions

In this paper, we present an original unsupervised deep anomaly detection framework in the context of horizontal axis wind turbines (WTs) based on SCADA data. The core of the method is the proposed neural architecture, namely a Graph Convolutional Autoencoder for Multivariate Time series (MTGCAE), which models the sensor network as a dynamical functional graph. The main advantage with respect to standard Autoencoders (AE) lies in the capability to simultaneously take into account the information content of the individual sensors measurements (graph node features) and the nonlinear correlations existing between all pairs of sensors (graph edges).

The proposed neural architecture is trained to learn the normal behaviour of the system without providing any kind of data labelling and, based on the model reconstruction errors, multiple monitoring indicators are defined, namely a Global Mahalanobis Indicator (GMI) for the whole sensor network, and a Local Residual Indicator (LRI) for each monitored variable. All indicators are evaluated considering both their magnitude and duration in time by a four-stage threshold method. In this way, only significant model errors are taken into account, allowing to attenuate the effect of individual spikes and transient disturbances, thus reducing false alarms during normal operating conditions. After the four-stage threshold, a warning is triggered for the sensor having the highest reconstruction error, allowing to isolate the assembly/sub-assembly mostly involved in the anomaly for troubleshooting purposes.

The proposed method was validated on 10-minute SCADA data collected from four WTs belonging to the same wind farm, with a rated power of 2 MW each. The dataset counts 12 failures on the most critical components (generator, gearbox, and transformer) occurred during 20 months of operation. The model was trained on normal behaviour data isolated using an unsupervised method and was tested on anomalous periods selected using the maintenance logs.

The presented model was compared with other two promising approaches, namely an architecture based on LSTMs and AEs (LSTM-AE) and one that combines CNNs and LSTMs (CNN–LSTM). To guarantee a reliable and robust analysis, we trained a separate model for each WT and considered all 12 anomalies for the final evaluation metrics.

The results show that the MTGCAE outperforms the other two neural architectures in terms of Precision (1.0), Recall (0.83) and F1-score (0.91). It is important to notice that the MTGCAE demonstrates the ability to capture hidden anomalies even when the turbine continues to deliver the power requested by the grid.

Since the model is unsupervised and completely data-driven, we expect it to be independent of the specific use case and potentially applicable to any WT equipped with a SCADA system.

CRediT authorship contribution statement

Eric Stefan Miele: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Fabrizio Bonacina:** Conceptualization, Methodology, Validation, Resources, Writing – original draft, Writing – review & editing. **Alessandro Corsini:** Writing – original draft, Writing – review & editing, Supervision.

Declaration of competing interest

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

References

[1] Liserre M, Cardenas R, Molinas M, Rodriguez J. Overview of multi-MW wind turbines and wind parks. IEEE Trans Ind Electr 2011;58(4):1081–95.
[2] Fung KT, Scheffler RL, Stolpe J. Wind energy - a utility perspective. IEEE Trans Power Appar Syst 1981;PAS-100(3):1176–82.
[3] Sesto E, Casale C. Exploitation of wind as an energy source to meet the world's electricity demand. J Wind Eng Ind Aerodyn 1998;74:375–87.
[4] Wee H-M, Yang W-H, Chou C-W, Padilan MV. Renewable energy supply chains, performance, application barriers, and strategies for further development. Renew Sustain Energy Rev 2012;16(8):5451–65.
[5] Hameed Z, Hong Y, Cho Y, Ahn S, Song C. Condition monitoring and fault detection of wind turbines and related algorithms: A review. Renew Sustain Energy Rev 2009;13(1):1–39.
[6] Kim K, Parthasarathy G, Uluyol O, Foslien W, Sheng S, Fleming P. Use of SCADA data for failure detection in wind turbines. In: Energy sustainability. 54686, 2011, p. 2071–9.
[7] Dao C, Kazemtabrizi B, Crabtree C. Wind turbine reliability data review and impacts on levelised cost of energy. Wind Energy 2019;22(12):1848–71.
[8] Stetco A, Dinmohammadi F, Zhao X, Robu V, Flynn D, Barnes M, et al. Machine learning methods for wind turbine condition monitoring: A review. Renew Energy 2019;133:620–35.
[9] Zaher A, McArthur S, Infield D, Patel Y. Online wind turbine fault detection through automated SCADA data analysis. Wind Energy Int J Prog Appl Wind Power Convers Technol 2009;12(6):574–93.
[10] Lebranchu A, Charbonnier S, Bérenguer C, Prévost F. A combined mono-and multi-turbine approach for fault indicator synthesis and wind turbine monitoring using SCADA data. ISA Trans 2019;87:272–81.
[11] Menezes D, Mendes M, Almeida JA, Farinha T. Wind farm and resource datasets: A comprehensive survey and overview. Energies 2020;13(18):4702.
[12] Ulmer M, Jarlskog E, Pizza G, Manninen J, Goren Huber L. Early fault detection based on wind turbine scada data using convolutional neural networks. In: 5th European conference of the prognostics and health management society, virtual conference, 27-31 July 2020. Vol. 5. PHM Society; 2020, p. 9.
[13] Maldonado-Correa J, Martín-Martínez S, Artigao E, Gómez-Lázaro E. Using SCADA data for wind turbine condition monitoring: A systematic literature review. Energies 2020;13(12):3132.
[14] Cui Y, Bangalore P, Bertling Tjernberg L. A fault detection framework using recurrent neural networks for condition monitoring of wind turbines. Wind Energy 2021.
[15] Pang G, Shen C, Cao L, Hengel AVD. Deep learning for anomaly detection: A review. ACM Comput Surv 2021;54(2):1–38.
[16] Pang Y, He Q, Jiang G, Xie P. Spatio-temporal fusion neural network for multi-class fault diagnosis of wind turbines based on SCADA data. Renew Energy 2020;161:510–24.
[17] Kong Z, Tang B, Deng L, Liu W, Han Y. Condition monitoring of wind turbines based on spatio-temporal fusion of SCADA data by convolutional neural networks and gated recurrent units. Renew Energy 2020;146:760–8.
[18] Xiang L, Wang P, Yang X, Hu A, Su H. Fault detection of wind turbine based on SCADA data analysis using CNN and LSTM with attention mechanism. Measurement 2021;175:109094.
[19] Yu B, Yin H, Zhu Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: Proceedings of the 27th international joint conference on artificial intelligence. IJCAI'18, AAAI Press; 2018, p. 3634–40.
[20] Wu Z, Pan S, Long G, Jiang J, Chang X, Zhang C. Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 2020, p. 753–63.
[21] Renström N, Bangalore P, Highcock E. System-wide anomaly detection in wind turbines using deep autoencoders. Renew Energy 2020;157:647–59.
[22] Chen J, Li J, Chen W, Wang Y, Jiang T. Anomaly detection for wind turbines based on the reconstruction of condition parameters using stacked denoising autoencoders. Renew Energy 2020;147:1469–80.
[23] Chen H, Liu H, Chu X, Liu Q, Xue D. Anomaly detection and critical SCADA parameters identification for wind turbines based on LSTM-AE neural network. Renew Energy 2021;172:829–40.
[24] Chalapathy R, Chawla S. Deep learning for anomaly detection: A survey. 2019, arXiv preprint arXiv:1901.03407.
[25] Yu JJQ, Gu J. Real-time traffic speed estimation with graph convolutional generative autoencoder. IEEE Trans Intell Transp Syst 2019;20(10):3940–51.
[26] Hu Y, Qu A, Work D. Graph convolutional networks for traffic anomaly. 2020, arXiv preprint arXiv:2012.13637.
[27] Yan X, Ai T, Yang M, Tong X. Graph convolutional autoencoder model for the shape coding and cognition of buildings in maps. Int J Geogr Inf Sci 2020;1–23.
[28] Zhang S, Tong H, Xu J, Maciejewski R. Graph convolutional networks: a comprehensive review. Comput Soc Netw 2019;6(1):1–23.
[29] Li G, Muller M, Thabet A, Ghanem B. Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 9267–76.
[30] Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 2020.
[31] EDP. Wind farm 1 - failures. 2016, Data retrieved from EDP Open Data, https://opendata.edp.com/explore/dataset/htw-failures-2016/information/.
[32] Baldi P. Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning. JMLR Workshop and Conference Proceedings; 2012, p. 37–49.
[33] Kelley HJ. Gradient theory of optimal flight paths. Ars J 1960;30(10):947–54.
[34] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th international conference on learning representations. ICLR '17, 2017, p. 1.

[35] Ross BC. Mutual information between discrete and continuous data sets. PLoS One 2014;9(2):e87357.

[36] Arora R, Basu A, Mianjy P, Mukherjee A. Understanding deep neural networks with rectified linear units. 2016, arXiv preprint arXiv:1611.01491.

[37] Lepot M, Aubin J-B, Clemens FH. Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. Water 2017;9(10):796.

[38] Savitzky A, Golay MJ. Smoothing and differentiation of data by simplified least squares procedures. Anal Chem 1964;36(8):1627–39.

[39] Prechelt L. Early stopping-but when? In: Neural networks: tricks of the trade. Springer; 1998, p. 55–69.

[40] Madsen PH, Risø D. Introduction to the IEC 61400-1 standard. RisøNational Laboratory, Technical University Of Denmark; 2008.

[41] Wilkinson M, Harman K, Hendriks B, Spinato F, van Delft T, Garrad G, et al. Measuring wind turbine reliability-results of the reliawind project. In: EWEA conference. 2011, p. 1–8.

[42] Sahnoun M, Bagui F, Messaadia M. Failure analysis of onshore wind farms based on experimental data. In: Mediterranean conference on information & communication technologies' 2015. 2015.