

Enhancing Industrial Time Series Anomaly Detection Through Graph Modeling and a Hybrid VAE-GAN Approach

Giovanni Zurlo[1(✉)], Roberto Cornali[1], and Elisabetta Ronchieri[1,2]

[1] INFN CNAF, Viale Carlo Berti Pichat 6, 40127 Bologna, BO, Italy
{gzurlo,elironc,eronchie}@cnaf.infn.it
[2] Department of Statistical Sciences, University of Bologna, Via Belle Arti 41, 40126 Bologna, BO, Italy

Abstract. In mission-critical industries, early anomaly detection can make a difference between seamless operations and costly downtime. Traditional statistical methods often struggle with the complexity and high dimensionality of temporal dependencies, whereas Graph Neural Networks (GNNs) are emerging as a powerful tool for predictive maintenance, offering deeper insights into data from interconnected and distributed systems.

Building upon our previously proposed detection model integrating Generative Adversarial Networks and GNNs, we improve and adapt it to a new semi-supervised setting. By transforming multivariate time series into static graph structures, our method separately captures spatial and temporal dependencies. Within an adversarial framework, we train an autoencoder on these graph representations to learn the system's normal behavior. Anomalies are identified by analyzing the graph reconstruction errors and the discriminator scores.

We test our approach through a real-life case study on industrial data from a gas turbine generator. A preliminary comparison with the baseline production model shows that our approach achieves higher precision, reducing both the frequency and duration of false alarms. Additionally, we conduct an ablation study by introducing synthetic anomalies through systematic connectivity alterations, assessing both the model's robustness and the suitability of the graph representation algorithm.

By emphasizing interpretability alongside detection accuracy, this study investigates the potential of GNNs for advancing predictive maintenance in complex industrial systems.

Keywords: Graph Neural Networks · Generative Adversarial Networks · Anomaly Detection · Predictive Maintenance · Industrial Machinery

1 Introduction

In mission-critical industries, continuous operation exposes equipment to wear that can escalate into costly failures and downtimes. Preventive or reactive maintenance strategies have traditionally been used to mitigate these risks, relying on

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15886, pp. 106–123, 2026.
https://doi.org/10.1007/978-3-031-97576-9_7

scheduled inspections and part replacements. However, these approaches are often inefficient, leading to unnecessary maintenance costs or unexpected failures. This has driven a shift toward more advanced, data-driven strategies that leverage real-time monitoring and predictive analytics to optimize maintenance operations.

The energy sector, in particular, relies on high-performance rotating machinery-such as turbines, compressors, and fans-having rotating mechanical parts that are exposed to enormous stress on a daily basis. The term "turbomachines" is conventionally adopted to refer to a specific class of these devices that extracts energy or imports energy from a continuously moving stream of fluid, either liquid or gas [32]. A turbomachine is a power or head-generating machine that employs the dynamic action of a rotating element, the rotor. The action of the rotor changes the energy level of the continuously flowing fluid through the machine. Damage to these types of systems not only causes economic loss but also a loss of reputation, therefore detecting and repairing them as early as possible is of utmost importance.

In modern industrial plants, almost all the installed equipment include IoT sensors for monitoring their behavior and remote-controlled actuators to regulate the operational profile and avoid undesired events. These IoT sensors provide real-time data on parameters such as temperature, vibration, pressure, and rotational speed. When paired with advanced analytics, machine learning algorithms, and historical data, these systems enable predictive maintenance-a proactive strategy that identifies potential failures before they occur. Predictive maintenance performs as a kind of trade-off between preventive and reactive. It leverages condition monitoring and anomaly detection to forecast wear patterns and detect irregularities. By analyzing trends and deviations from expected performance, maintenance teams can schedule repairs and replacements at optimal times, reducing unplanned outages and extending the life of the equipment. This not only minimizes costs but also enhances safety, reliability, and operational efficiency, making it a vital practice in modern industrial settings.

With the rise of deep learning and generative models, reconstruction-based anomaly detection methods have gained significant attention due to their ability to model complex data distributions. Unlike traditional statistical approaches, which often struggle with high-dimensional and nonlinear data, these methods leverage deep neural architectures to learn underlying patterns in time series data. Autoencoders, Variational Autoencoders (VAEs), and Generative Adversarial Networks (GANs) are among the most commonly used models in this category [4]. Their ability to learn a compressed representation of normal system behavior makes them particularly suitable for identifying deviations indicative of potential faults [27].

In this work, we build upon our previously proposed anomaly detection model, adapting it to a new semi-supervised setting [10]. While the original method operated in an unsupervised fashion due to the annotation cost and rarity of anomalies, we now leverage labeled instances to assess detection performance. Our model requires representing multivariate time series data as a static

graph, highlighting relevant information while filtering noise. With Graph Neural Networks (GNNs) [38] we can model space and time relationships separately and enhance our results' interpretability by detecting anomalous edges explicitly. We apply this adapted approach to a new industrial domain, specifically to data from an oil and gas extraction plant, demonstrating its effectiveness and explainability in real-world anomaly detection tasks.

This paper is organized as follows: Sect. 2 summarizes some works related to this study. Sections 3.1, 3.2, 3.3 provide basic concepts on GANs, GNNs, and Graph Autoencoders. Section 4 introduces the anomaly detection model in use. Section 5 details our case study, with results and further speculations and finally Sect. 6 gives conclusions and future works.

2 Related Works

Traditionally, distance [5] and density-based techniques [39] have been widely used for detecting irregularities in time series data. With the advent of deep learning, neural network architectures including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers have demonstrated significant advantages in modeling time series data. Between 2020–2023, nearly 50% of newly introduced anomaly detection solutions were prediction-based methods leveraging these neural building blocks [4]. Prediction-based methods detect anomalies by measuring forecast and reconstruction errors, relying on the assumption that normal data are easier to predict, whereas unexpected anomalies lead to higher errors. These methods are most effective in semi-supervised settings, where the training set contains little to no anomalous data. Forecasting-based models predict future values based on past sequences, while reconstruction-based models learn normal behavior by encoding and reconstructing time series sub-sequences, using reconstruction errors as an anomaly score.

Autoencoders have been widely adopted for reconstruction-based anomaly detection [34,41], but they are prone to overfitting and struggle to regularize the latent space effectively, even with probabilistic algorithm variants. Generative Adversarial Networks (GANs) [17] have been introduced to address these issues by learning implicit distributions that generalize better. While a naive GAN formulation relies solely on discriminator loss, effective adaptations such as the Multivariate Anomaly Detection GAN (MAD-GAN) [27] and TadGAN [14] incorporate generator reconstruction loss into anomaly scoring. However, remapping data to the latent space during inference remains computationally expensive, representing a bottleneck for real-time applicability. A practical solution involves designing the generator and/or discriminator as an autoencoder to jointly leverage both reconstruction and discrimination capabilities [1,3].

Despite their success, models based on CNNs and RNNs struggle with explicitly capturing pairwise variables dependencies, such as spatial relationships in non-Euclidean spaces, which limits their expressiveness [22] and effectiveness in detecting complex anomalies [45].

Graph Neural Networks (GNNs) have recently gained traction for anomaly detection by modeling both inter-variable (relationships among different variables in a multivariate series) and inter-temporal (dependencies across time points) associations [7,12]. The idea of using graphs for time series anomaly detection dates back to the early 2000s; early approaches represented time points as graph vertices and detected anomalies using random walks [8,9]. However, these classical methods often struggle to learn informative representations and model fine-grained dependencies.

Within the industrial domain of application, [28] employs a cross-correlation graph-based encoder-decoder GAN (CCG-EDGAN) for anomaly detection in wind and steam turbines. Spatio-temporal GNNs have also been applied to condition monitoring of distributed systems of wind turbines [30] and photovoltaic panels [42], demonstrating the potential of graph-based models for the predictive maintenance of complex systems.

3 Preliminaries

3.1 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of deep learning models designed for generative tasks by employing a game-theoretic framework. Introduced by Goodfellow et al. [17], a GAN consists of two neural networks: a generator G_ϕ parametrized by $\boldsymbol{\phi}$, and a discriminator D_ψ which compete in a zero-sum game. The main motivation for the design of GANs is that the learning process requires neither approximate inference nor approximation of a partition function gradient [16]. The generator G_ϕ learns to map random noise $\mathbf{z}$ from a prior distribution $p_{prior}(\mathbf{z})$ to a data distribution $p_{data}(\mathbf{x})$, producing synthetic samples $G(\mathbf{z}, \boldsymbol{\phi}) = \tilde{\mathbf{x}}$. The discriminator D_ψ, a binary classifier, distinguishes real samples $\mathbf{x}$ from generated ones $\tilde{\mathbf{x}}$ by estimating the probability $D(\tilde{\mathbf{x}}, \boldsymbol{\psi}) \in [0, 1]$ that a given input is real. During training, G_ϕ is optimized to generate increasingly realistic samples to fool D_ψ, while D_ψ improves its ability to distinguish real from fake data. This process is governed by an adversarial loss $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\psi})$, where G_ϕ seeks to minimize it while D_ψ aims to maximize it. At equilibrium, G_ϕ generates samples that are indistinguishable from real data, thus D_ψ can be discarded.

GAN training is notoriously challenging, often suffering from non-convergence, instability, and mode collapse, where G_ϕ generates only a limited subset of the target distribution, leading to poor sample diversity [16]. Despite these challenges, GANs have been successfully applied to tasks such as image synthesis, style transfer, and data augmentation, showcasing their ability to produce highly realistic samples.

3.2 Graph Neural Networks

The success of Convolutional Neural Networks (CNNs) in computer vision has inspired the generalization of convolution operations to graph-structured data

[6]. This has led to the development of Graph Neural Networks (GNNs), with two primary approaches for defining convolutions on graphs: spatial-based and spectral-based methods [46]. **Spatial-based approaches** define convolution directly in the graph's vertex domain, mirroring the operation of traditional CNNs. The core idea is to compute the convolution for each vertex as a weighted average of its neighboring nodes, where the weights reflect the influence of these neighbors. A significant challenge in this approach lies in designing convolution operations that can handle neighborhoods of varying sizes while maintaining the weight sharing property characteristic of CNNs [20,43].

Spectral-based approaches, on the other hand, define graph convolutions as filters rooted in graph signal processing methodologies. From this perspective, a graph convolution can be seen as a technique to reduce noise in graph signals. This involves transforming the graph signals into the frequency domain using the eigen-decomposition of the graph Laplacian matrix, where eigenvalues represent frequencies and eigenvectors correspond to harmonics [6,35]. The graph Fourier transform decomposes the signal into its harmonic coefficients, and filters are defined as diagonal matrices that scale each frequency component.

However, the computational cost associated with calculating eigenvectors, especially for large graphs, prompted the development of more efficient methods. ChebNet [11] addressed this by approximating the spectral filter g_θ using Chebyshev polynomials $T_k(L)$ of the diagonal matrix of eigenvalues, up to the Kth order. The convolution of a graph signal x with this filter can be approximated as:

$$g_\theta \star \mathbf{x} \approx \sum_{k=0}^{K} \boldsymbol{\theta}'_k T_k(L)\mathbf{x} \tag{1}$$

where $\star$ denotes the convolution operator, and $\boldsymbol{\theta}' \in \mathbb{R}^K$ are the Chebyshev coefficients.

The Graph Convolutional Network (GCN) module employed here simplifies Eq. (1) by limiting the approximation to the first order ($K = 1$), arguing that stacking multiple localized graph convolutional layers can create a deeper, yet parameter-efficient, architecture capable of capturing graph structure [24]. Approximating the largest eigenvalue $\lambda_{max} \approx 2$, the convolution operation in Eq. (1) becomes: $g_\theta \star \mathbf{x} = \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}$ where θ is a scalar Chebyshev coefficient. To address potential issues with exploding or vanishing gradients, a symmetric renormalization strategy is also introduced:

$$\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \rightarrow \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2} \tag{2}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ (adding self-loops) and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$.

From a spatial perspective, the GCN can be reformulated as a form of neural message passing [15]. In this view, the hidden state of each node at the next layer is generated by aggregating and updating vector messages from its neighbors and itself using neural networks. Specifically, representations from the previous layer are aggregated across the neighborhood (including the node itself), a weighted average is taken, and then transformed by a weight matrix and an activation function to obtain the new representation of the node:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i\cup\{i\}} \frac{\tilde{A}_{ij}}{\sqrt{\tilde{D}_{ii}\tilde{D}_{jj}}}\mathbf{h}_j^{(l)}\mathbf{W}^{(l)}\right) \tag{3}$$

where $\mathcal{N}_i$ denotes the set of neighbors of node i.

3.3 Graph Autoencoders

An autoencoder is an artificial neural network designed to learn a compressed representation (encoding) of input data while minimizing reconstruction loss. It consists of two components: an encoder and a decoder. The encoder maps input data to a lower-dimensional latent space, forcing it to retain only the most relevant information, while the decoder reconstructs the original data from this encoding. In the case of graph-structured data, this framework is referred to as a graph autoencoder (GAE). The entire network is trained jointly to optimize reconstruction accuracy.

A fundamental limitation of standard autoencoders in generative modeling is the structure of their latent space. The encoded representations may be discontinuous or highly clustered, making random sampling and smooth interpolation challenging. If a decoder is presented with an out-of-distribution latent vector (e.g., one sampled from a region where no training encodings exist), it is likely to generate unrealistic outputs.

Variational Graph Autoencoders (VGAEs) [25] address this issue by enforcing a continuous, well-structured latent space. Instead of outputting a single deterministic encoding, the encoder in a VAE produces two vectors: the mean vector and the standard deviation vector, which define a Gaussian distribution for each latent dimension. The actual encoding is then sampled from this learned distribution:

$$q(\mathbf{H} \mid \mathbf{X}, \mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{h}_i \mid \mathbf{X}, \mathbf{A}), \quad q(\mathbf{h}_i \mid \mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{h}_i \mid \boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2)) \tag{4}$$

Here, $\boldsymbol{\mu} = \mathrm{GCN}_\mu(\mathbf{X}, \mathbf{A})$ and $\log \boldsymbol{\sigma} = \mathrm{GCN}_\sigma(\mathbf{X}, \mathbf{A})$ denote the outputs of GCN modules parameterizing the mean and standard deviation of the latent distribution. This so-called "reparameterization trick" [23] allows gradient-based optimization by re-expressing the stochastic sampling operation as a differentiable transformation.

In VAEs, the decoder reconstructs the graph by predicting its structure $\mathbf{A}$ from the latent embeddings $\mathbf{H}$:

$$p(\mathbf{A} \mid \mathbf{H}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p(A_{ij} \mid \mathbf{h}_i, \mathbf{h}_j), \quad \text{where} \quad p(A_{ij} = 1 \mid \mathbf{h}_i, \mathbf{h}_j) = \sigma(\mathbf{h}_i^\top \mathbf{h}_j) \tag{5}$$

where A_{ij} are the elements of $\mathbf{A}$ and $\sigma(\cdot)$ is the sigmoid function. For attributed graphs, node features X can also be reconstructed using additional GCN layers.

This stochastic generation ensures that even for the same input, the sampled encoding varies slightly, exposing the decoder to different representations of the same data point and improving generalization.

However, unregularized autoencoders primarily focus on minimizing reconstruction loss and often ignore the global structure of the latent space, leading to degenerate identity mappings [33]. To address this, latent space regularization techniques enforce a structured prior distribution. One effective approach is adversarial regularization (AR), where an additional discriminator is trained to distinguish between true prior samples and latent encodings, encouraging the learned distribution to align with the prior [36].

4 Methodology

4.1 Graph Design

Representing a time series as a graph is a non-trivial task. In this work, we adopt a single-layer, observations-based nodes approach known as correlation network [44]. According to the taxonomy presented in [21], this method falls under the similarity-based heuristic approaches for time series mapping. Prior research highlights the importance of correlation structures in multivariate time series, as they provide critical insights into system behavior and anomaly detection [19] [29]. Changes in inter-variable correlations often indicate different types of anomalies, including point, contextual, and collective anomalies. Several alternative methods for mapping time series to graphs exist, as detailed in the comprehensive review of multivariate time series network representations [40].

In our approach, we construct graphs using correlation matrices, where each entry-representing a pair (time, signal)-is assigned as a node attribute. Two nodes are connected by an undirected edge if the correlation between their signals over the last $w = 100$ timestamps exceeds a threshold τ_c. Additionally, nodes corresponding to the same signal are connected by a directed edge if the autocorrelation between their timestamps surpasses a threshold τ_a (Fig. 1). The thresholds τ_c and τ_a are selected within stable ranges to prevent excessive sensitivity to minor fluctuations [44], with final values set to $\tau_c = 0.4$ and $\tau_a = 0.3$. If an edge is present, its weight is assigned based on the computed auto- or cross-correlation value. This design enables us to model temporal and inter-signal relationships separately, focusing on the most significant dependencies in the data.

The resulting graph is inherently signed, as negative edges naturally arise from negatively correlated signal windows. However, many graph neural network (GNN) architectures assume unsigned graphs, necessitating an additional preprocessing step. A few naive strategies include removing negative edges, ignoring edge signs, or binarizing the adjacency matrix A such that $A \in \{0,1\}^{N\times N}$ [13]. Instead, to retain correlation information while ensuring compatibility with the GCN module, we normalize the correlation values to the range [0,1] using:

$$\bar{\rho}_{ij} = \frac{\rho_{ij} + 1}{2} \tag{6}$$

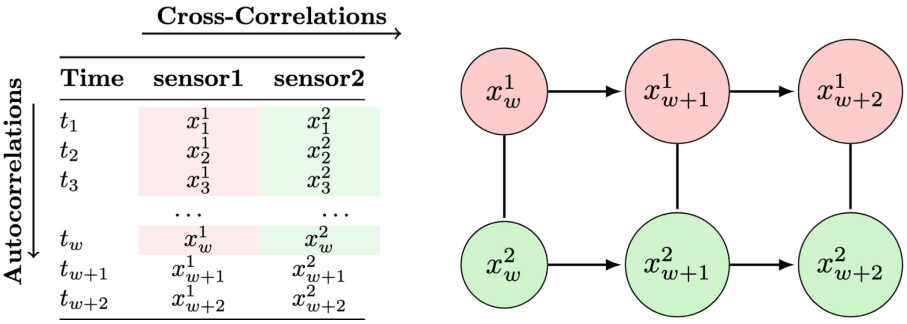

Fig. 1. The Correlation Network Approach

4.2 Model Architecture

The model introduced in [10] extends the AR-VGAE (described in Sect. 3.3) by incorporating an additional graph discriminator and refining the reconstruction process. Specifically, it adopts a Gaussian prior on the latent representations to enforce smoothness and structure in the learned embeddings. The variational Graph Encoder $Enc : \mathbf{x}_i|\mathbf{A} \rightarrow [\mu_i, \sigma_i]$ follows the inference model of Eq. (4) and is constructed using three shared GCN modules along with separate linear layers for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$.

The Graph Decoder $Dec : \mathbf{h} \rightarrow \{\tilde{A}, \tilde{X}\}$ reconstructs both the topological structure $\mathbf{A}$ and node features $\mathbf{X}$. The attributed adjacency matrix is reconstructed using the standard VGAE approach via a sigmoid inner product (Eq. 5). For node feature reconstruction, the decoder incorporates three additional GCN layers to capture local node interactions effectively:

$$\tilde{\mathbf{x}}_i = GCN(\mathbf{h_i}|\tilde{\mathbf{A}}; \theta_\phi). \tag{7}$$

Together, these components form the VGAE part of the model, acting as a dual generator. Differently from the original implementation of [10], our autoencoder incorporates dropout layers with a 0.5 rate and use ReLU activations in all layers except the final ones, which use tanh and sigmoid functions to match the range of the output data.

The Regularizing Discriminator $D_r : \mathbf{h_i} \rightarrow r \in [0, 1]$, introduced in [36] and discussed in Sect. 3.3, is implemented as a multi-layer perceptron (MLP) with a single output neuron and a sigmoid activation function. It serves as an adversarial classifier, distinguishing between latent codes sampled from the Gaussian prior and those produced by the Graph Encoder. This adversarial regularization ensures that the learned latent space remains structured and aligned with the prior, preventing degenerate solutions.

To further refine the generation process, [10] introduced a Graph Discriminator, $D_g : \tilde{\mathbf{x}}_i|\mathbf{A} \rightarrow \ r \in [0, 1]$, to assess the authenticity of reconstructed nodes. This discriminator is a Graph Convolutional Network (GCN) with message-passing modules, a linear layer, and a sigmoid activation function,

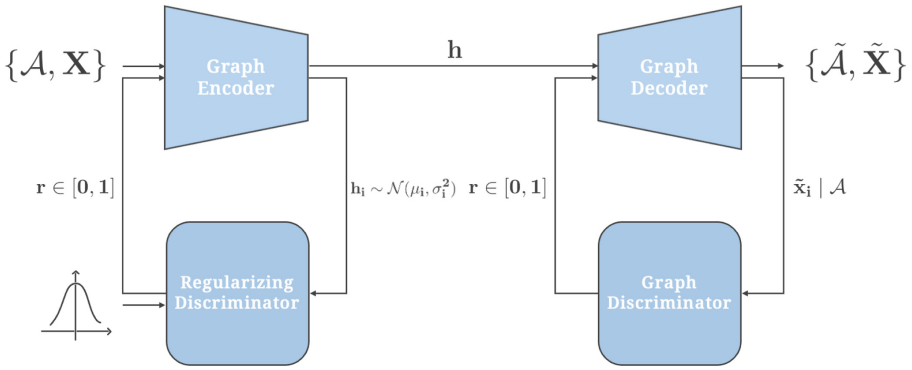

Fig. 2. Network Architecture of the Employed Model [10]

assigning a realness score between 0 and 1 to each decoded node. The complete architecture is illustrated in Fig. 2.

Previous studies indicate that dropout plays an important role in discriminator networks [16,37]. In particular, units should be randomly dropped while computing the gradient for the generator network to be followed. For our message-passing module of choice, we employ dropout as the primary source of stochasticity in the training process [24].

Overall, this AR-VGAE extension leverages both reconstruction-based anomaly detection and adversarially regularized embeddings, improving both interpretability and robustness in semi-supervised settings.

4.3 Model Training

The training of our model follows the adversarial learning framework of [10]. The objective is to simultaneously optimize the Graph Autoencoder for accurate reconstruction and the two discriminators for adversarial regularization and anomaly detection. Given the nature of our task, Wasserstein losses [2] are employed for both adversarial components, as they provide a more stable optimization process compared to standard GAN objectives. Specifically, one Wasserstein loss ensures a structured latent space by aligning it with a Gaussian prior, while the other improves the generation quality of reconstructed node features by encouraging realism in their distribution.

Unlike traditional adversarial training, where instability and mode collapse are major concerns, Wasserstein GAN with Gradient Penalty (WGAN-GP) [18] is adopted to enforce a local Lipschitz constraint on the discriminators. This regularization stabilizes training by penalizing large gradients, preventing the discriminators from becoming overly confident too quickly, which could otherwise disrupt the training dynamics of the encoder-decoder pair.

In addition to adversarial objectives, the Graph Autoencoder is optimized using an ℓ_2 reconstruction loss to directly minimize the discrepancy between

original and reconstructed node features. This serves as an additional constraint, ensuring that even if the adversarial training introduces instabilities, the model retains a strong signal from the reconstruction task. By jointly optimizing adversarial and reconstruction losses, our approach balances reconstruction fidelity, latent space regularization, and adversarial discrimination.

While previous GAN work has used momentum to accelerate training, we adopt the Adam optimizer with tuned hyperparameters as in DCGAN [37]. We found the suggested learning rate of 0.001 to be too low, using 0.02 and 0.01 instead. Additionally, leaving the momentum term β_1 at the suggested value of 0.9 resulted in training oscillation and instability while reducing it to 0.5 helped stabilize training. In this way, we decrease the smoothness of the momentum, the exponentially weighted moving average of past gradients, to take care of the rapidly changing gradients because the generator and the discriminator compete with each other.

4.4 Anomaly Scores

Since a discriminator D_g is trained to distinguish fake data (i.e. anomalies) from real data with high sensitivity, a naive GAN-based anomaly detection approach might use the discriminator loss as the anomaly score. However, this method has limitations because the discriminator primarily serves to support the generator's learning rather than directly detect anomalies. To improve accuracy, it is now common practice to combine the generator's reconstruction errors with the discriminator loss when calculating the anomaly scores [14].

Our model generates both a critic score and a reconstructed attribute for each node in the input graph. This allows for a fine-grained examination of the model's predictions. The Graph Discriminator's scores on the real graph are low only if it cannot classify nodes as belonging to the real graph itself; we exploit this fact to identify nodes as anomalous. Node reconstruction errors are measured as the distance or dissimilarity between real and reconstructed attributes. Some methods calculate point-wise reconstruction errors, while others use time-aware interval-based errors, such as area difference [14].

In our case study, we compute the mean and standard deviation of sensor-wise reconstruction errors $RE(\mathbf{x})$, along with robust indices for the critic output $C(\mathbf{x})$. We then transform the z-scores $Z_{RE}(\mathbf{x})$ and $Z_C(\mathbf{x})$ to be non-negative by clipping or taking absolute values and shift them to have unit minimum. We explored two different methods to combine these into a single dataframe of anomaly scores. First, we take a convex combination $S(\mathbf{x}) = \alpha Z_C(\mathbf{x}) + (1 - \alpha) Z_{RE}(\mathbf{x})$ where α controls the relative importance of both terms. Alternatively, we tried to multiply the scores to emphasize high values $S(\mathbf{x}) = \alpha Z_C(\mathbf{x}) \odot Z_{RE}(\mathbf{x})$.

The scores obtained through this approach enable point-wise anomaly scoring by comparing values at each timestamp for each sensor. By applying an aggregation operation, such as a row-wise average or maximum, we can assess the operating conditions at the system level.

4.5 Two Phase Semi-supervised Approach

Our workflow consists of two-phases: (1) training a model exclusively on normal operating conditions data and (2) optimizing a decision threshold using labeled anomalies from the same training set:

1. During training, labeled anomalies are excluded to ensure that the model learns only normal operating conditions. This prevents the model from inadvertently adapting to anomalous patterns.
2. During inference, the decision threshold is adjusted to maximize precision or the generalized F_β score on the labeled anomalies.

While the F_1 score is commonly used for balancing precision and recall, prioritizing precision is more effective in scenarios where anomalies are rare or occur over short periods. The F_β score introduces a tunable parameter β to emphasize recall when necessary. By optimizing the threshold for high precision, we aim to minimize false positives while ensuring timely and reliable anomaly detection, thereby improving operational alerting systems [26].

Additionally to applying a single system-wide anomaly threshold, we experiment with a per-sensor approach, where each sensor is assigned an individual threshold τ_i. These thresholds are determined based on statistical dispersion of the anomaly scores, controlled by a tunable parameter σ (Eq. 8). Specifically, the anomaly threshold for each sensor is set as a function of its normal behavior scores $\mathbf{S}_{\text{NOC}}$, ensuring adaptability across different sensors with varying signal characteristics. Once anomalies are detected at the sensor level, we aggregate the individual detections using an ensemble-like strategy, leveraging agreement among sensors to improve robustness.

$$\boldsymbol{\tau} = \boldsymbol{\mu}(\mathbf{S}_{\text{NOC}}) + \sigma \cdot \boldsymbol{\sigma}(\mathbf{S}_{\text{NOC}}) \tag{8}$$

5 Case Study

We trained our model using sensor data from a 10.5 MWe gas turbine generator, which provides self-generated electricity for an entire industrial plant. The training dataset consists of 55 covariates that monitor both performance and operating conditions. These include turbine and rotor speeds, multiple temperature readings (related to combustion, oil, bearings), fuel pressure and flow rates, and vibration levels in the bearings and gearbox. Additionally, electrical parameters such as generator amperage, voltage, frequency, and power output are available. The dataset spans 390 days of continuous multivariate time series data. Raw data were originally recorded at 30-second intervals, but for practical processing, first differences were taken and downsampled to 5-minute intervals. To account for operational status, we used the generator's power output as a discrete on-off parameter. Transient phases-including shutdown and ramp-up periods-were handled in the preprocessing pipeline. Specifically, for model training, we excluded data from 12 h before shutdown and 24 h post-ramp-up. However, during anomaly detection, we only removed the first 3 h after ramp-up to retain more operational data.

The trained model was validated using a separate test set from the same machine, covering 447 days of operation. This dataset includes a single anomaly event lasting 6 days, allowing us to assess the model's real-world performance.

5.1 Results

Experiments with various combination strategies revealed that the most effective approach involved computing a weighted sum of node reconstruction errors and critic scores with a very low weight for the latter. In the system-wise approach, these errors were aggregated at the system level using a mean. Overall, node attribute reconstruction errors emerged as the most critical anomaly indicator, highlighting the contextual nature of anomalies in the input graphs. Correlation heatmap analysis further confirmed that node reconstruction errors closely align with correlation patterns observed in the input data.

After fine-tuning the threshold and score combination strategy on the training set, our model repeatedly flags anomalies throughout the 7-day anomalous interval (Fig. 3). To enhance contextual relevance, we also apply a temporal aggregation logic, grouping consecutive point anomalies into anomaly segments. Specifically, point anomalies occurring within a 12-hour window are merged into a single anomalous event, ensuring that isolated detections do not overly fragment the results. With this aggregation strategy, the longest detected contextual anomaly falls within the actual confirmed anomaly. False positives are primarily associated with sudden large variations or transient periods; however, these are generally short-lived and could be effectively filtered through post-processing.

An individual analysis of scores revealed that only a few sensor-specific columns were effective in detecting the anomalous interval. The sensor-wise approach described in Sect. 4.4 mitigates the averaging effect of system-level aggregation, allowing key sensor contributions to be more distinctly highlighted when treated as an ensemble. However, this approach also increased the false positive rate (see Fig. 4).

We compared our model's scores to those generated by the existing anomaly detection system deployed at the plant (Fig. 5), using data available for a 175-day sub-interval. The baseline model relies on a more traditional reconstruction-based approach using principal component analysis (PCA) and kernel density estimation (KDE). For the performance comparison on raw scores, we used the baseline model's "Warning" threshold as a reference, since the "Alert" threshold did not provide valid metrics. To ensure consistency in the normalization, we set an arbitrary "Warning" threshold for our model while using the fine-tuned one as our reference "Alert". From a qualitative perspective, the baseline model offers broader anomaly detection coverage but also exhibits higher rate of false positives with varying durations, potentially leading to unnecessary alerts. Quantitatively, the baseline achieved a higher F_1 score of 0.249 ($F_{0.2} = 0.292$), but at the expense of lower precision (0.296 vs. 0.366 for our model) due to extended false positive intervals. This preliminary comparison suggests that differences between the two models may stem from operational preferences. Additionally, we are currently investigating whether certain discord patterns remain

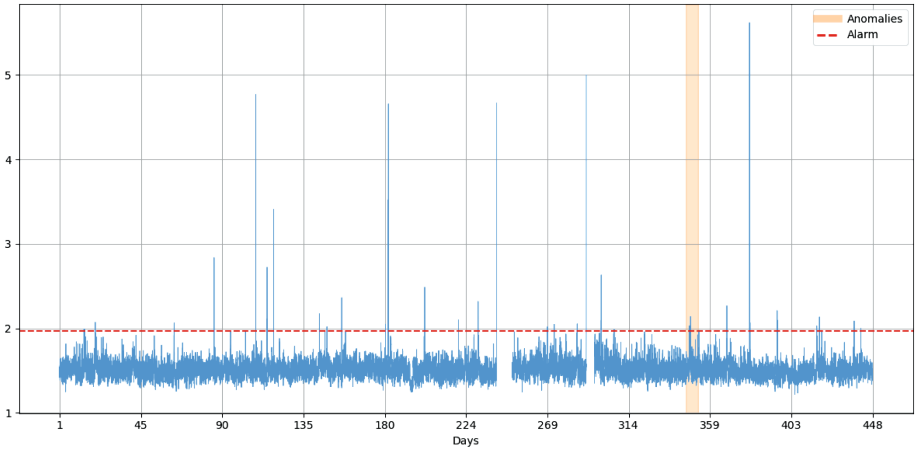

Fig. 3. Aggregated Anomaly Scores Across 55 Signals from the Examined Gas Turbine $Scores = (Z_{RE}, Z_C)$, $\alpha = 0.3$, $\tau = 1.995$, $smooth = 30\ min$, $gap = 12H$

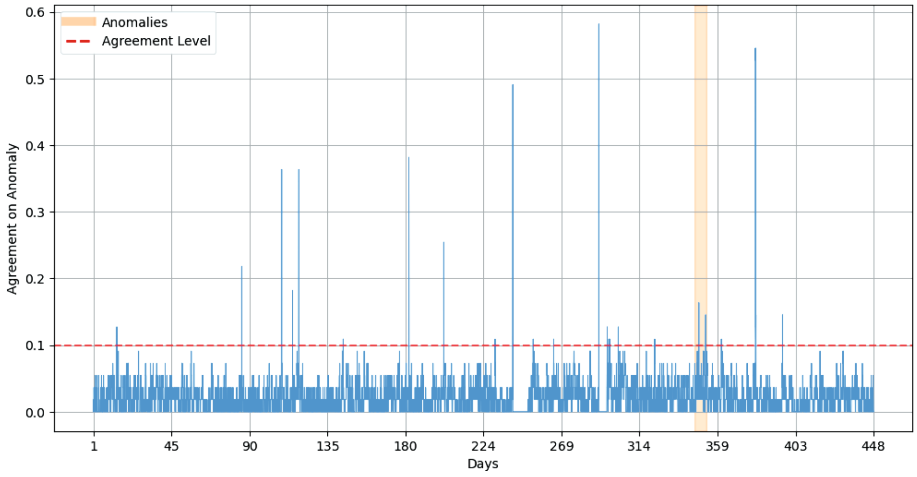

Fig. 4. Signal-wise Anomaly Agreement among Test Set Scores $Scores = (Z_{RE}, Z_C)$, $\alpha = 0.2$, $\sigma = 3.56$, $smooth = 30\ min$, $gap = 12H$

undetected by the baseline model (days 145–150 in Fig. 5). As an additional benchmark, we implemented a model that is architecturally similar to our proposed approach but operates directly on first-order differenced time series data rather than graph-structured inputs. This variant, inspired by [14], replaces the GCN modules in both the generators and critic with Bidirectional Long Short-Term Memory (BiLSTM) layers, making it more suited to sequential rather than relational modeling. While this baseline exhibited slightly broader coverage of

the true anomaly interval, it achieved a lower $F_{0.2}$ score of 0.177 and an F_1 score of 0.14, primarily due to a higher false positive rate (precision = 0.216).

5.2 Ablation Study on Known Anomalies

In graph anomaly detection, we typically encounter two main types of anomalies: **Structural anomalies** manifest as nodes with unusual connectivity patterns. They often appear as densely connected clusters that deviate from the generally sparse connectivity of regular nodes in the graph. In **contextual anomalies**, node attributes significantly differ from their neighbors, regardless of their structural properties. Unlike structural anomalies, these could also be detected using traditional tabular outlier detection methods [31].

In real-world datasets, naturally occurring outlier nodes do not always fit neatly into the categories of structural or contextual anomalies. Their characteristics may not be explicitly defined, meaning they might not belong to either category or could exhibit characteristics of both. This ambiguity makes identifying organic outliers more challenging than detecting synthetic ones, which typically follow a predefined pattern [31].

We designed a benchmark to evaluate the robustness of our graph anomaly detection approach when faced with increasingly sparse structural anomalies. The experiment involves injecting synthetic structural patterns into a network-

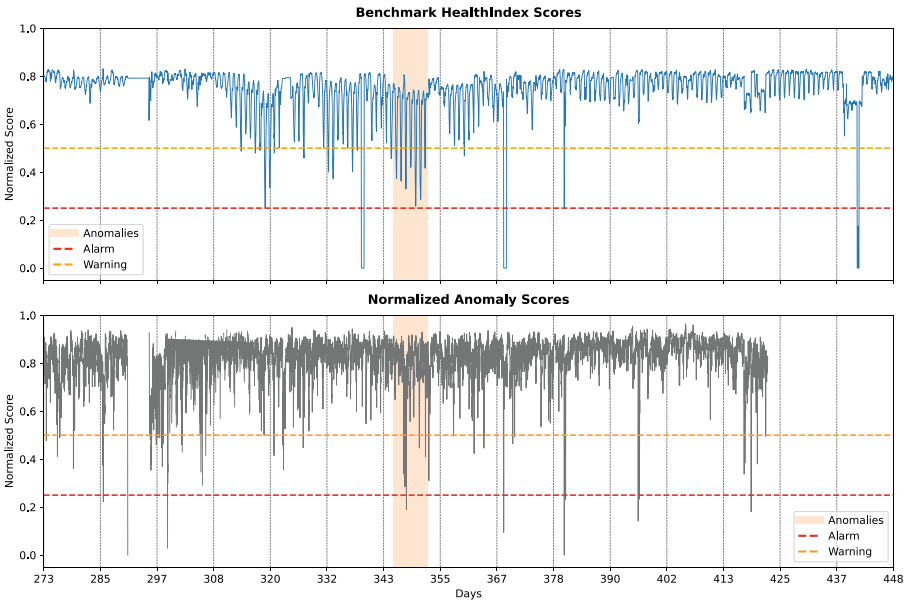

Fig. 5. Visual Comparison of the Test Set Anomaly Scores (Benchmark vs. Ours) Raw Benchmark achieved a precision of 0.296, recall = 0.214, $F_1 = 0.249$, $F_\beta = 0.292$. Our model resulted in a precision of 0.366, recall = 0.078, $F_1 = 0.128$, $F_\beta = 0.210$.

Table 1. Performance evaluation of the AD model on synthetically augmented anomalies. Results show detection metrics across different sparsification levels of injected cliques, controlled by edge dropout probability p. The F_β score and weight of the critic α are fine-tuned on the same set and reported alongside graph statistics. Added Edges represents the number of uncoalesced synthetic connections retained after dropout, while Total Edges indicates the final size of the coalesced graph structure.

p	.998	.9985	.999	.9995	1
$F_{0.2}$	.89	.88	.86	.77	.27
α	1	1	1	1	.4
Total Edges (Millions)	63	62.7	62.4	62.1	62
Added Edges (Millions)	1.24	.93	.62	.31	0

mapped test set and systematically reducing their connectivity to assess detection performance degradation. Starting from a baseline graph constructed following the procedure in Sect. 4.1, we introduce synthetic structural anomalies by embedding fully connected subgraphs (cliques) within a known anomaly period. To maintain consistency with the temporal windowing originally used, we generate 41 distinct cliques, each spanning up to 100 timestamps (same as w used for graph mapping). The edge attributes for these synthetic connections are sampled from the existing attribute distribution to ensure realistic features. The key experimental manipulation involves gradually degrading these anomalous patterns through a controlled edge dropout procedure. By randomly removing edges with increasing probability p, we systematically reduce the density of the injected patterns. This process creates a spectrum of anomalies that become progressively harder to distinguish from the normal graph structure mapping. Each resulting graph configuration is then evaluated using our trained model, which combines node reconstruction errors and critic scores to generate system-level anomaly scores. Detection performance is assessed through a grid search over different threshold values and critic scores weight (α), with results quantified using the F_β score (Table 1).

This methodology allows us to quantify exactly how sparse an anomalous pattern can become before it becomes less detectable, providing insights into the practical limitations of our graph-based anomaly detection approach. The findings have direct implications for real-world applications, where anomalous behavior often manifests not as perfectly dense structures, but as partial or shallow reconstructions of suspicious temporal patterns.

6 Conclusions

In this paper, we further investigated the approach proposed by [10], which integrates GANs, a reconstruction-based framework, and GNNs for effective and interpretable anomaly detection. We introduced several innovations: A sensor-wise anomaly detection approach shows promise, particularly when only a subset

of signals exhibits anomalies. This method avoids system-wide aggregation by treating sensors as an ensemble of detectors. By setting individual thresholds, we can better preserve the role of key sensors in identifying anomalous intervals.

However, our findings also reveal certain limitations of the current model. Specifically, the model appears to over-rely on node attribute reconstruction errors as the primary anomaly indicator. The anomalies in our datasets are largely contextual, meaning that nodes exhibit deviations in their attributes compared to their neighbors, rather than unusual structural properties in the graph. This suggests that an effective model should prioritize attributes deviations over structural connectivity patterns when identifying anomalies.

We believe the full potential of this approach could be realized with a more advanced graph mapping algorithm—one that better captures contextual anomalies as structural deviations. Enhancing the representation of multivariate time series as graphs in a way that more effectively maps temporal contextual anomalies to structural ones could improve both anomaly detection performance and model interpretability. While our correlation-based graph construction allows for separate treatment of temporal and spatial relationships, it may not fully embed the structural intricacies of contextual anomalies.

In light of these observations, we are currently considering variants of our model using newer GNN modules that support signed graphs [13] or spatial-temporal graph mapping [21]. Until now, we have relied on conventional methods such as GCN, which are designed for unsigned graphs. However, disregarding the distinction between positive and negative links may introduce noise or lead to the loss of critical insights specific to signed graph data. By leveraging models explicitly designed for signed graphs, we aim to improve the model's ability to differentiate between various types of relationships, ultimately enhancing its anomaly detection capabilities.

Acknowledgments. The work presented in this paper is performed in the framework of the ICSC project – *Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing*, funded by the NextGenerationEU European initiative through the Italian Ministry of University and Research, PNRR Mission 4, Component 2: Investment 1.4, Project code CN00000013 - CUP I53C21000340006.

References

1. Akcay, S., et al.: GANomaly: Semi-supervised anomaly detection via adversarial training (2018). https://doi.org/10.48550/arxiv.1805.06725
2. Arjovsky, M., et al.: Wasserstein GAN (2017). https://doi.org/10.48550/ARXIV.1701.07875
3. Audibert, J., et al.: USAD: unsupervised anomaly detection on multivariate time series. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3395–3404. KDD '20, Association for Computing Machinery, New York, NY, USA (2020)
4. Boniol, P., et al.: Dive into time-series anomaly detection: A decade review (2024). https://arxiv.org/abs/2412.20512

5. Breunig, M.M., et al.: LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 93–104. SIGMOD '00, Association for Computing Machinery, NY, USA (2000)
6. Bruna, J., et al.: Spectral networks and locally connected networks on graphs (2014). https://arxiv.org/abs/1312.6203
7. Chen, Z., et al.: Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. IEEE Internet Things J. **9**(12), 9179–9189 (2022). https://doi.org/10.1109/jiot.2021.3100509
8. Cheng, H., et al.: Detection and Characterization of Anomalies in Multivariate Time Series, pp. 413–424. https://doi.org/10.1137/1.9781611972795.36
9. Cheng, H., et al.: A robust graph-based algorithm for detection and characterization of anomalies in noisy multivariate time series, pp. 349–358 (2008). https://doi.org/10.1109/ICDMW.2008.48
10. Cornali, R., et al.: Graph neural network solutions for anomaly detection in time series. In: Proceedings of the 58th Hawaii International Conference on System Sciences, p. 997. HICSS'58 (2025). https://doi.org/10.24251/HICSS.2025.119
11. Defferrard, M., et al.: Convolutional neural networks on graphs with fast localized spectral filtering (2017). https://arxiv.org/abs/1606.09375
12. Deng, A., et al.: Graph neural network-based anomaly detection in multivariate time series. Proc. AAAI Conf. Artif. Intell. **35**(5), 4027–4035 (2021). https://doi.org/10.1609/aaai.v35i5.16523
13. Dinh, T.T., et al.: On the modelling and impact of negative edges in graph convolutional networks for node classification. In: NeurIPS 2023 Workshop: New Frontiers in Graph Learning (2023). https://openreview.net/forum?id=U0EPHUDrot
14. Geiger, A., et al.: TadGAN: Time series anomaly detection using generative adversarial networks (2020). https://doi.org/10.48550/arxiv.2009.07769
15. Gilmer, J., et al.: Neural message passing for quantum chemistry (2017). https://arxiv.org/abs/1704.01212
16. Goodfellow, I., et al.: Deep Learning. MIT Press (2016)
17. Goodfellow, I.J., et al.: Generative adversarial networks (2014). https://doi.org/10.48550/arxiv.1406.2661
18. Gulrajani, I., et al.: Improved training of wasserstein GANs (2017). https://doi.org/10.48550/arxiv.1704.00028
19. Hallac, D., et al.: Toeplitz inverse covariance-based clustering of multivariate time series data (2018). https://arxiv.org/abs/1706.03161
20. Hamilton, W.L., et al.: Inductive representation learning on large graphs (2018). https://arxiv.org/abs/1706.02216
21. Jin, M., et al.: A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection (2024). https://arxiv.org/abs/2307.03759
22. Jin, M., et al.: Towards expressive spectral-temporal graph neural networks for time series forecasting (2025). https://arxiv.org/abs/2305.06587
23. Kingma, D.P., et al.: Auto-encoding variational bayes (2013). https://arxiv.org/abs/1312.6114
24. Kipf, T.N., et al.: Semi-supervised classification with graph convolutional networks. CoRR arxiv:abs/1609.02907 (2016). http://arxiv.org/abs/1609.02907
25. Kipf, T.N., et al.: Variational graph auto-encoders (2016). https://doi.org/10.48550/arxiv.1611.07308
26. Lavin, A., et al.: Evaluating real-time anomaly detection algorithms - the Numenta anomaly benchmark. CoRR arxiv:abs/1510.03336 (2015)

27. Li, D. et al.: MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks, pp. 703–716. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-30490-4_56
28. Liang, H., et al.: Consistent anomaly detection and localization of multivariate time series via cross-correlation graph-based encoder decoder GAN. IEEE Trans. Instrum. Meas. **71**, 1–10 (2022). https://doi.org/10.1109/TIM.2021.3139696
29. Liang, H., et al.: Robust unsupervised anomaly detection via multi-time scale DCGANs with forgetting mechanism for industrial multivariate time series. Neurocomputing **423**, 444–462 (2021). https://doi.org/10.1016/j.neucom.2020.10.084
30. Liu, J., et al.: Condition monitoring of wind turbines with the implementation of spatio-temporal graph neural network. Eng. Appl. Artif. Intell. **121**(C) (2023). https://doi.org/10.1016/j.engappai.2023.106000
31. Liu, K., et al.: Bond: Benchmarking unsupervised outlier node detection on static attributed graphs (2022). https://arxiv.org/abs/2206.10071
32. Logan et al.: Handbook of Turbomachinery. Mechanical Engineering, CRC Press (2003). https://books.google.it/books?id=CNrKBQAAQBAJ
33. Makhzani, A., et al.: Adversarial autoencoders (2016). https://arxiv.org/abs/1511.05644
34. Malhotra, P., et al.: LSTM-based encoder-decoder for multi-sensor anomaly detection (2016). https://arxiv.org/abs/1607.00148
35. Ortega, A., et al.: Graph signal processing: Overview, challenges and applications (2018). https://arxiv.org/abs/1712.00468
36. Pan, S., et al.: Adversarially regularized graph autoencoder for graph embedding (2019). https://arxiv.org/abs/1802.04407
37. Radford, A., et al.: Unsupervised representation learning with deep convolutional generative adversarial networks (2016). https://arxiv.org/abs/1511.06434
38. Scarselli, F., et al.: The graph neural network model. IEEE Trans. Neural Networks **20**(1), 61–80 (2009). https://doi.org/10.1109/tnn.2008.2005605
39. Schölkopf, B., et al.: Support vector method for novelty detection. In: Proceedings of the 13th International Conference on Neural Information Processing Systems, pp. 582–588. NIPS'99, MIT Press, Cambridge, MA, USA (1999)
40. Silva, V.F., et al.: Time series analysis via network science: Concepts and algorithms. WIREs Data Mining and Knowledge Discovery **11**(3) (2021)
41. Su, Y., et al.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2828–2837. KDD '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330672
42. Van Gompel, J., et al.: Cost-effective fault diagnosis of nearby photovoltaic systems using graph neural networks. Energy **266**, 126444 (2023). https://doi.org/10.1016/j.energy.2022.126444
43. Velickovic, P., et al.: Graph attention networks (2018). https://arxiv.org/abs/1710.10903
44. Yang, Y., et al.: Complex network-based time series analysis. Phys. A: Stat. Mech. Appl. **387**(5 6), 1381–1386 (2008)
45. Zhao, H., et al.: Multivariate time-series anomaly detection via graph attention network (2020). https://arxiv.org/abs/2009.02040
46. Zhou, J., et al.: Graph neural networks: a review of methods and applications. AI Open **1**, 57–81 (2020). https://doi.org/10.1016/j.aiopen.2021.01.001