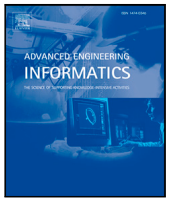
Advanced Engineering Informatics 60 (2024) 102357

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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



Full length article

From anomaly detection to classification with graph attention and transformer for multivariate time series

Chaoyang Wang, Guangyu Liu *

Zhejiang Provincial Laboratory of IoT and Information Fusion Technologies, School of Automation, Hangzhou Dianzi University, China

ARTICLE INFO

Keywords:
Multivariate time series
Anomaly detection
Classification
Graph attention
Transformer

ABSTRACT

Numerous industrial environments and IoT systems in the real world contain a range of sensor devices. These devices, when in operation, produce a large amount of multivariate time series data. These data are interconnected and together reflect the status and changing trends of the devices and environment. To ensure the safety and stability of the working devices, it is of great research value to continuously monitor the generated data and perform accurate and rapid anomaly detection. Moreover, in some emergency situations, after detecting an anomaly, making a preliminary judgment and reporting the type of anomaly can save a lot of precious time and reduce property losses. Therefore, we propose an anomaly detection and classification architecture that can quickly perform sequence modeling. The architecture includes our proposed novel graph learning method and the high-efficiency Transformer model. We use two-stage adversarial training to train the anomaly detection model and apply the model to anomaly classification using a prototype network. Extensive experiments on four public datasets prove that our anomaly detection method outperforms other state-of-the-art methods. We also verify the accuracy of anomaly classification through case studies.

1. Introduction

With the rapid proliferation of devices connected to the Internet, the Internet of Things (IoT) generates a vast amount of sensor data, characterized by its geographical and temporal dependencies. From natural phenomena such as temperature, wind speed, and precipitation, to the substantial data produced across various industry sectors such as business (e.g., sales and market trends), finance (e.g., stock prices), biomedical sciences (e.g., heart activity and glucose level), and manufacturing (e.g., sensor data and yield), these data reflect the high-speed generation of IoT data. Given the widespread application of IoT, it has permeated all aspects of life and industry. It is crucial to focus on improving efficiency, safety, and security in handling the extensive data produced by IoT devices. Additionally, enhancing products, processes, and services associated with this technology is of utmost importance. Faced with increasingly frequent anomalies and network attacks, critical infrastructures also require robust security monitoring. This is especially true for those based on Cyber-Physical Systems (CPS), such as smart grids, water treatment and distribution networks, transportation systems, and autonomous vehicles. This need has spurred interest in powerful and accurate anomaly detection systems that continuously monitor basic controls or indicators and promptly alert potential anomalies [1].

Currently, the detection of data anomalies, or any behaviors that do not conform to expected trends, has become an active research discipline known as multivariate time series (MTS) anomaly detection [2]. In our work, we emphasize this aspect because in many real-world scenarios, a large number of IoT sensors continuously generate substantial time-series data. A time series refers to a sequence of data points organized in chronological order. It is commonly characterized by observations gathered over specific intervals of time [3]. Additionally, anomaly detection is particularly important in industrial environments like wind turbines [4,5] and photovoltaic (PV) systems [6]. For example, PV systems constantly generate data, making them susceptible to various operational and environmental anomalies. These anomalies could significantly reduce system efficiency and even lead to failures. By utilizing advanced anomaly detection technologies, potential issues in PV systems can be promptly identified and resolved, ensuring the reliability and efficiency of energy production. But anomalies are typically rare and obscured by a vast number of normal data points, making the process of data labeling challenging and costly. Therefore, our attention is directed towards unsupervised anomaly detection in time series data. In addition, the dependencies among sensors are initially hidden and are often difficult to obtain in most real-life scenarios. Establishing a system that can quickly and accurately pinpoint anomalies has always been a challenging problem.

* Corresponding author.
E-mail address: g.liu@hdu.edu.cn (G. Liu).

https://doi.org/10.1016/j.aei.2024.102357
Received 22 September 2023; Received in revised form 27 December 2023; Accepted 5 January 2024
Available online 23 January 2024

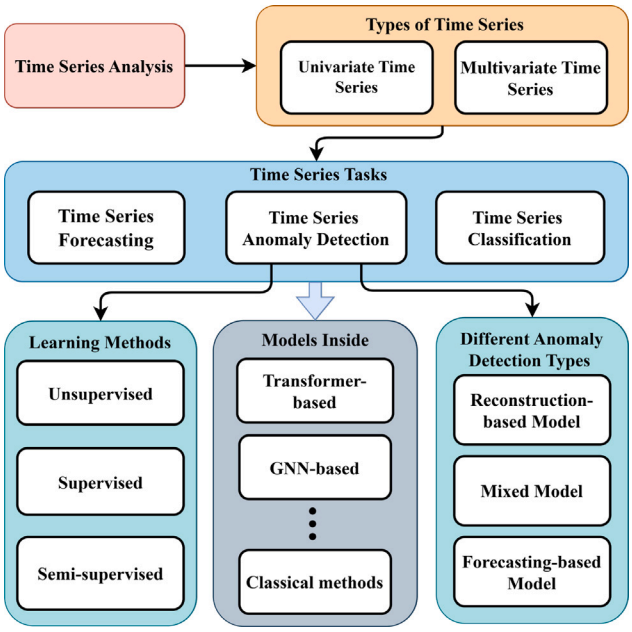


Fig. 1. Architecture of Time Series Analysis.

In recent years, deep learning models have become increasingly prevalent in the field of anomaly detection [7]. Among these, Graph Neural Networks (GNN) [8] have been introduced to better handle high-dimensional data and learn the topological structure between sensors. In addition, in the context of multivariate time series analysis, spatio-temporal modeling stands as a pivotal component. The capability of GNN modules to capture spatial dependencies, specifically their ability to recognize latent relationships between dimensions, has been widely acknowledged [9]. Concurrently, the Transformer architecture [10], with its demonstrated prowess in modeling long-range dependencies in sequence data, has garnered significant attention [11]. With the questioning and verification of predecessors, the advantages of Transformers in time series feature extraction have been widely recognized [12]. The amalgamation of GNNs and Transformers has been empirically proven to excel in areas such as climate forecasting [13], transportation [14,15] and multimodal prediction. This fusion also aids in a more profound understanding of spatio-temporal dynamics and causative relationships. Nevertheless, existing methods are not able to effectively learn the complex topological relationships between sensors while also improving time efficiency, which does not align with the high response speed requirements of IoT environments.

Moreover, in the field of MTS analysis, one existing challenge is the lack of integrated models capable of consecutively performing anomaly detection and subsequent classification. In fault diagnosis within photovoltaic power stations, the role of prototype networks is particularly crucial. Due to the relative difficulty in collecting abnormal data from photovoltaic power stations, there are often only a few samples of faults available, especially for extremely serious faults. This constitutes a typical problem of few-shot learning [16]. Prototype networks can effectively utilize these limited fault samples for learning and establish prototypes of both faulty and normal states. When new operational data emerges, a rapid and accurate fault diagnosis can be performed by comparing the distance between this data and the prototypes of fault and normal states. Furthermore, for newly occurring types of faults, the prototype of the new faults can be computed by collecting only a small number of new samples, thereby incorporating the new faults into the diagnostic scope.

To address these problems, we propose an innovative model GIN. This model first employs a **G**raph Attention Network for feature extraction, followed by an enhanced Transformer model, the **In**former, for long sequence time modeling. We have significantly increased the inference speed of our model while combining these two aspects, making it far superior to other similar models. Finally, the model uses adversarial reconstruction to perform binary classification to detect the presence of anomalies. To demonstrate the efficiency of our model, we conduct experiments on four public datasets, the experimental results show that the proposed method achieves substantial improvement over the baseline methods. Then, through ablation studies, we verified the importance of each component. Moreover, we employ a Prototype Network for anomaly classification. Through a case study, we present our classification results. The findings indicate that our model can achieve high classification accuracy, we also visualized the results of the experiment.

The contributions of this paper are summarized as follows:

- We propose a novel anomaly detection method that employs an enhanced graph attention network and Transformer model to account for temporal and spatial dependencies respectively.
- We propose a new sparse graph attention mechanism, which filters the neighborhood of each node to induce sparsity, and employs layer-independent multi-head graph attention coefficients.
- We propose a two-step time series classification architecture, where an encoder pre-trained in the anomaly detection phase is employed within a prototypical network for the classification task.
- We combined a variant of the Transformer with a two-stage adversarial training to achieve excellent results in both long-term and short-term time series.

The rest of the paper is organized as follows. Section 2 presents the related works. In Section 3, we detail our proposed method. Section 4 demonstrate the efficacy of our method. Finally, conclusions are given in the last section.

2. Related work

Time series analysis encompasses a vast domain [17], covering three primary tasks: forecasting, anomaly detection, and classification [18–20]. Based on the dimensionality of the input data, it can be further categorized into multivariate and univariate time series. As illustrated in Fig. 1, we have consolidated the overarching architecture of time series analysis. For multivariate time series anomaly detection, the overall framework suggests that the existing methods can be broadly classified into three categories: forecasting-based, reconstruction-based, and mixed models. Forecasting-based models employ learning algorithms to predict a subsequence based on a individual point or recent window. To determine the degree of anomaly of the input values, the predicted values are contrasted with the actual ones. Reconstruction models endeavor to reconstruct inputs by encoding normal training data subsequences in a latent space. The reconstruction error is obtained by comparing it with the actual values, and anomalies are flagged when this error exceeds a certain threshold. Beyond these, mixed models merge the virtues of both prediction-based and reconstruction-based models, aspiring to attain a superior representation of the time series. Currently, most models employ semi-supervised or unsupervised learning methods during their training phase, given that the models are trained solely on normal data. In essence, anomaly detection is a binary classification task, the aforementioned three model types specialize for this task, distinguishing them from the conventional multi-class tasks and laying a theoretical foundation for the amalgamation of anomaly detection and classification tasks. Next, we will explore the models inside the realm of anomaly detection, which includes both classical methods and deep learning models such as those based on Transformers and GNNs.

Classical methods. For modeling time-series data, common classical methods include the use of k-means clustering, SVMs, and regression models [21–23]. Other techniques use PCA [24], process

regression, or hidden Markov chains [25], while some leverage wavelet theory or signal transformations like the Hilbert transform [26]. Isolation forests, which employ an ensemble of trees to partition the feature space, are also used [27,28]. Another method applies ARIMA [29] to model linear dependencies in time-series for anomaly detection.

Deep learning based methods. Recently, various deep learning approaches have made significant advancements in anomaly detection for high-dimensional datasets. The AutoEncoder (AE) [30] relies on reconstruction errors to generate anomaly scores. The DAGMM [31] method uses a Deep Autoencoding Gaussian Mixture Model for feature space reduction. This model's Gaussian parameters come from a deep neural network, and a recurrent network handles temporal modeling. The LSTM-NDT [32] approach is based on an LSTM predictive model and introduces a non-parametric dynamic error threshold (NDT) strategy. This strategy employs the moving average of error sequences to set the threshold for anomaly markings. MSCRED [33] transforms a given sequence of input data into a normalized two-dimensional image. This image is then processed through a ConvLSTM layer and utilizes a convolutional decoder to detect anomalies based on reconstruction errors. LSTM-VAE [34] introduces a process-based varying prior, integrates signals, and reconstructs their expected distribution. OmniAnomaly [35] employs a stochastic model simliar to LSTM-VAE to mitigate the potential misleading effects of uncertain instances. By integrating stochastic variable connections and normalizing flow, this model calculates reconstruction probabilities. This method outperformed previous ones, but required longer training times. MAD-GAN [36] innovatively employs an LSTM-based GAN model to model the distribution of time series, it then utilizes discriminator loss in the anomaly scores and prediction error to detect anomalies. USAD [37] method employs an adversarial training framework utilizing an autoencoder that is equipped with two decoders. Also, it is one of the earliest works to focus on improving time efficiency and significantly reduces training time compared to previous models.

With the rise of Transformers and GNNs, the latest works have shifted away from the use of resource-intensive recurrent models, instead utilizing attention-based network structures to enhance training speeds. The Graph Deviation Network (GDN) [38] was designed to understand the pairwise relationships through the use of cosine similarity, allowing it to build a graph as an adjacent matrix. This graph was then used to predict future values using a graph attention-based forecasting approach. The accuracy of these predictions was assessed using an absolute error value. CAE-M [39] constructed a Deep Convolutional Autoencoder to mirror a target low-dimensional distribution, minimizing overfitting from data noise and anomalies. On the other hand, the MTAD-GAT [40] used a combination of feature-oriented and time-oriented graph attention layers to learn the structure of the graph. This method also used the mixed model to compute an integrated loss by joint optimization. To detect anomalies, the MTAD-GAT applied an automatic threshold algorithm.

However, these models above have their limitations. Due to the use of a smaller input window and the absence or simplification of recurrent models, they struggle to effectively recognize long-term dependencies. The TranAD approach, as proposed in [41], introduces an adversarial learning technique to emphasize reconstruction discrepancies that might be overlooked by a standard Transformer network, particularly those stemming from subtle anomalies. It leverages pairs of Transformer encoders and decoders to bolster the stability of the model. Recently, BTAD [42] utilizes the Bi-Transformer structure to extract correlated features. Meanwhile, it adopts an adaptive multi-head attention mechanism similar to TranAD to parallelly infer the trends of various dimensions in multivariate time series data, thereby enhancing the performance. The Anomaly Transformer [43] model integrates the Transformer architecture with a Gaussian prior-Association to enhance the distinction between anomalies. In addition to the traditional reconstruction loss, AnomalyTrans then employs a minimax strategy to make anomalies even more discernible. While these three methods employ Transformer architecture to capture the long-term dependencies in time series, the models still incur significant memory and computational overhead due to the shortcomings of the traditional self-attention mechanism. GTA [44] integrates the Transformer with a graph-based learning architecture for multivariate time series anomaly detection. It also employs a multi-branch attention mechanism to reduce memory overhead and enhance inference speed. However, as the number of time points increases, the forecasting-based model it uses may increase the prediction error, at the same time, its time efficiency still needs to be improved.

3. Methodology

3.1. Problem statement

A multivariate time series is a sequence of observations or data points associated with timestamps, denoted as

$$T = \{x_1, \ldots, x_T\} \tag{1}$$

Each data point x_t is collected at a specific timestamp t, and each x_t belongs to $\mathbb{R}^m$, i.e., it is an m-dimensional real vector. The univariate setting, where $m = 1$, is a specific case of this. In the model, the time series window W and the time slice C play pivotal roles, each with its distinct characteristics and interrelationship. To capture the dependency of a data point x_t at a timestamp t, a local contextual window of length K is introduced, termed as W_t. This window encompasses x_t and its preceding $K-1$ data points,

$$W_t = \{x_{t-K+1}, \ldots, x_t\} \tag{2}$$

when the timestamp t is less than K, a technique called replication padding is employed. In such cases, to maintain a window length of K, a constant vector of length $K-t$ is appended to the window W_t. Through this approach, the original long time series T gets transformed into a sliding window sequence W. The model predominantly employs W as its training input, with $\hat{W}$ designated for the test series. Simultaneously, C_t, acting as the time slice, signifies the subsequence from the start of the time series to the current timestamp t, providing the model with a more global context. Thus, while W offers the model a local context of a particular data point, C_t delivers a complete subsequence from the inception of the time series up to t, serving as the global context.

Anomaly Detection: Given a training input time series T, for any unseen test time-series $\hat{T}$ of length $\hat{T}$ and the same modality as the training series, we aim to predict $Y = \{y_1, \ldots, y_{\hat{T}}\}$, where $y_t \in \{0, 1\}$ denotes whether the data point at the tth timestamp of the test set is anomalous, with 1 signifying an anomaly.

Anomaly Classification: Given the same training and test time-series, we aim to predict $Y = \{y_1, \ldots, y_{\hat{T}}\}$ after the anomaly has been detected. For a dataset with K classes, the one-hot label vector Y_t has a length of K. Each element j in the range [1, K] is set to 1 if the class of the input time series corresponds to j, and 0 if not.

In essence, the output of anomaly detection is a binary label indicating whether each data point is anomalous. In contrast, for anomaly classification, the output is a categorical label denoting the type of anomaly.

3.2. Topology construction

We employ a variant of Graph Neural Networks (GNNs) [8,45] to discern the topological structures and relationships among various dimensions, specifically between different sensors. In this network, the adjacency matrix and the node feature matrix (consisting of feature vectors for each node) are utilized to refine node representations. To achieve this objective, GNNs adopt an iterative aggregate-update strategy. In this approach, during each iteration, every node receives information from its neighboring nodes and subsequently updates its own states or features. This mechanism allows GNNs to capture intricate

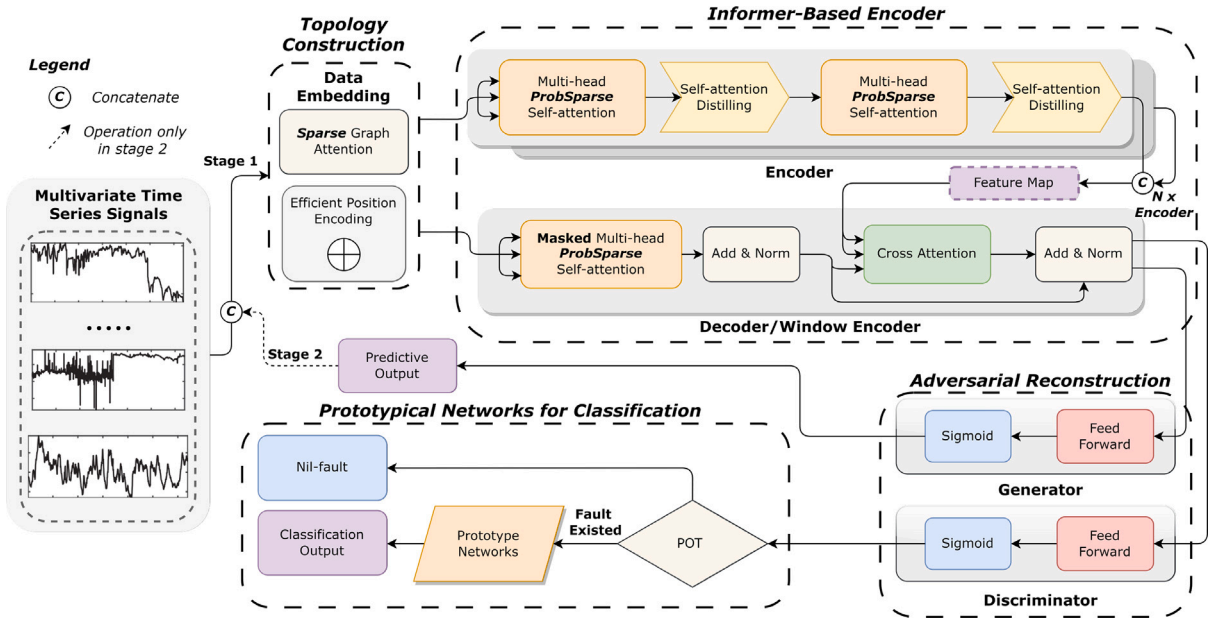


Fig. 2. Architecture of our proposed model with 4 main parts for time series anomaly detection and classification. (1) Time series embedding using SP-GAT and position encoding. (2) Informer-based Encoding. (3) GAN-based two-stage training. (4) Prototypical networks for classification.

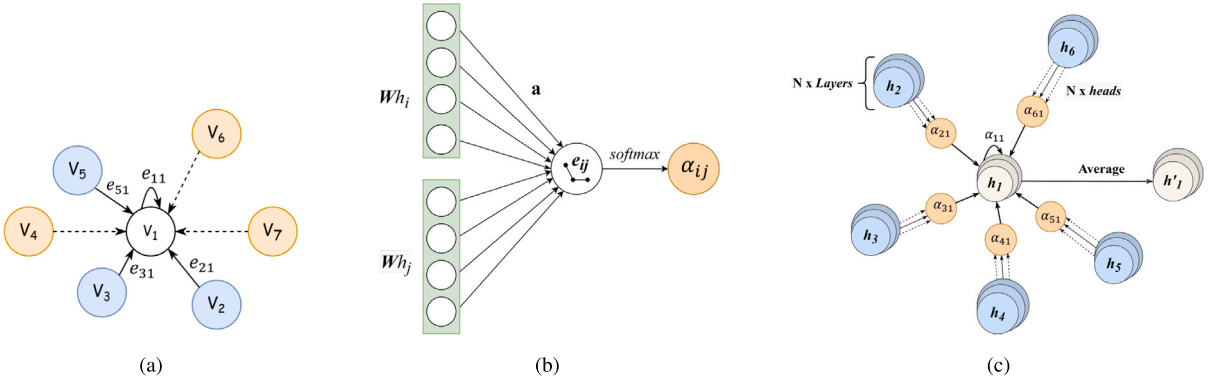


Fig. 3. The overview of SP-GATs: (a) an illustration of basic structure where the blue vertices represent the vertices being filtered through Pearson correlation coefficient and can generate an edge pointing to v_1 that belong to N_i and N_i^{raw}, the yellow vertices belong only to N_i^{raw} and the edges are represented by dashed lines; (b) an illustration of the calculation process of the edge e_{ij}; (c) an illustration of complete structure with attention coefficients independent of layers and heads.

structures and patterns within the graph. A specific variant of GNN, the Graph Attention Networks (GATs) [46], employs an attention mechanism to calculate the edge weights of each layer based on node features. This enables the model to adaptively focus on all neighbors of a node for representation learning, thereby enhancing the model's expressive power. This design implies that the influence exerted by different neighboring nodes on the current node can vary, reflecting real-world scenarios. For instance, in industrial contexts, the importance of data from different sensors also changes. GATs make further use of multi-head attentions to amass features: every head operates independently to accumulate information, and all the results from the multi-heads are subsequently joined together to generate a new feature depiction for the subsequent layer. Fundamentally, the attention coefficient that is learned can be regarded as a significance score for an edge e_{ij}.

In GAT modeling, redundancy not only adds to the computational overhead and memory usage, but also poses an overfitting challenge. This is particularly the case when the graphs, constructed from real-world temporal data, often contain noise. Without proper normalization, GATs can easily overfit to such noise. To further minimize redundancy between edges or to eliminate noisy edges, we use a sparsity constraint in the GAT's attention mechanism. This provides greater robustness for downstream classification tasks.

Now, we propose a new graph learning method named Sparse-Pearson Graph Attention Networks (SP-GAT) for the topology construction. Fig. 3 visualizes the main steps of this method. The input multivariate time series W_i is viewed as a set of vertex features, $h = \{h_1, h_2, \ldots, h_N\}$ where $h_i \in \mathbb{R}^F$ and N is the number of vertices and F is the number of features for each vertex. This layer will produce a new set of vertex features, where $h_i' \in \mathbb{R}^{F'}$ as the output. The output dimension needs to be the same as the input dimension, and position embeddings are added before feeding it into the encoder.

For the topological structure of the input data, we constructed a graph G. It can be represented as a combination of a set of vertices V and a set of edges E, i.e., $G = (V, E)$. Each vertex v_i has an associated time series feature vector h_i, and the relationship between each pair of vertices is indicated by the weight of the edge a_{ij}. We use the Pearson correlation coefficient [47] to quantify the relationships between vertices (i.e., the edges). Then, by selecting the top-K edges with the

highest values and removing the connections from other vertices to the chosen vertex, we determine how many edges are ultimately connected to each vertex. This process is the first step in sparsification. Now, we calculate the correlation between all nodes and denote it with e_{ik}^{raw}:

$$e_{ik}^{raw} = \text{Pearson}(h_i, h_j) = \frac{\sum_k (h_{ik} - \bar{h}_i)(h_{jk} - \bar{h}_j)}{\sqrt{\sum_k (h_{ik} - \bar{h}_i)^2 \sum_k (h_{jk} - \bar{h}_j)^2}} \tag{3}$$

where h_{ik} and h_{jk} are the kth elements of vectors h_i and h_j respectively, and $\bar{h}_i$ and $\bar{h}_j$ are the means of vectors h_i and h_j.

Next, we will use the top-K method to filter out the nodes that are most closely connected to node i, serving as its true neighbors. Prior to this, the entire graph is a fully connected directed graph, and each node also has an additional edge pointing to itself. The initial neighbor set of node i is N_i^{raw}, and the set after filtering is N_i.

$$N_i = \{j | j \in \text{TopK}(\{e_{ik}^{raw} | k \in N_i^{raw}\})\} \tag{4}$$

We then compute e_{ij} only for vertices $j \in N_i$, where N_i is the set of neighbors of vertex i that remain after Pearson coefficient filtering.

$$e_{ij} = a(W h_i, W h_j) \tag{5}$$

where $a(.)$ is a shared attention mechanism that maps $\mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$. In order to obtain sufficient expressiveness when transforming the input time series to high-dimensional features, each vertex uses a linear transformation with shared parameters, where the parameter is $W \in \mathbb{R}^{F' \times F}$. To allow comparison of attention coefficients across different vertices, we normalize them using softmax:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \tag{6}$$

In additon, in a single layer of GATs, each edge will receive multiple attention coefficients. On the other hand, the same edge at different layers and heads has a different set of attention coefficients. Although the purpose of multi-head attention mechanism is to allow the model to learn from multiple perspectives and focus points, thereby improving its performance and stability. However, according to research [48], in GATs, all heads usually learn attention coefficients with similar distributions, indicating that there is significant redundancy in the GAT model. Therefore, we use the following formula:

$$h_i^{(l+1)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in N_i} \alpha_{ij} h_i^{(l)} W_k^{(l)}\right) \tag{7}$$

where K is the number of heads, with attention coefficients a_{ij} shared among all heads and layers, resulting in F' features for each node. We use multi-head attention in the last layer of the network. Afterward, we average the outputs from all the heads and defer the use of the final non-linear layer, sigmoid.

3.3. Informer-based encoder

After spatial modeling of the input data, we use a model based on a variant of the Transformer called Informer [49] to extract temporal features. Transformer architectures, originally proposed by [10], have garnered significant attention in various fields due to their unparalleled capacity in modeling long-range dependencies. Central to the design of Transformer is its self-attention mechanism and its extension, the multi-head attention. Given a sequence of token representations $X \in \mathbb{R}^{n \times d}$, where n signifies the sequence length and d the dimensionality, the self-attention function projects these tokens into queries $Q \in \mathbb{R}^{n \times d_k}$, keys $K \in \mathbb{R}^{n \times d_k}$, and values $V \in \mathbb{R}^{n \times d_v}$. The scaled dot-product attention mechanism then computes the attention weights as:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{8}$$

This mechanism extends to multi-head attention, allowing the model to attend to various representation subspaces at different positions. The cumulative output of the multi-head attention is:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h) W_O \tag{9}$$

with each $head_i$ being the output of the self-attention function with its respective projections,

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{10}$$

where the projection matrices are defined as $W_i^Q \in \mathbb{R}^{d \times d_k}$, $W_i^K \in \mathbb{R}^{d \times d_k}$, $W_i^V \in \mathbb{R}^{d \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d}$. Transitioning to the Informer model, a notable enhancement is the ProbSparse self-attention. It employs a mechanism to identify and prioritize important queries based on their attention distribution. The most prominent feature is that Informer can reduce the $O(L^2)$ time complexity caused by the quadratic dot product in the transformer to $O(L \log L)$, thereby improving work efficiency. If a query's attention distribution is closer to a uniform distribution, the resulting self-attention is viewed as redundant, as it merely produces a trivial sum of values. To measure this redundancy, Informer uses the Kullback–Leibler (KL) divergence, comparing the attention probability distribution with a uniform distribution.

An efficient method proposed by Informer to acquire the sparsity of the query is through an empirical approximation:

$$\bar{M}(q_i, K) = \max_j \left\{ q_i k_j^T p \right\} - \frac{1}{L_K} \sum_j q_i k_j^T p \tag{11}$$

This measurement, $\bar{M}(q_i, K)$, encapsulates the likeness or similarity between the query's attention distribution and a uniform distribution. Based on this measurement, Informer refines the attention mechanism to focus only on the top u dominant queries, resulting in the ProbSparse self-attention:

$$\text{ProbAttention}(Q, K, V) = \text{Softmax}\left(\frac{\bar{Q}K^T}{\sqrt{d_k}}\right)V \tag{12}$$

where $\bar{Q}$ is a sparse matrix analogous to Q but contains only the top u queries based on the sparsity measure. This approach ensures minimal information loss, especially in a multi-head perspective, making Informer an efficient and effective model for time-series data processing. Moreover, Informer innovatively introduces self-attention distilling. This approach, incorporates convolution, activation, and maximum pooling operations between encoder and decoder layers. This technique effectively reduces the preceding layer's input sequence length by half, significantly addressing memory constraints. The process can be mathematically represented as:

$$X_{j+1}^t = \text{MaxPool}\left(\text{ELU}\left(\text{Conv1d}([X_j^t]_{AB})\right)\right) \tag{13}$$

Here, $[.]_{AB}$ denotes the attention block, which encompasses the Multi-head ProbSparse self-attention and other essential operations. The 1-D convolutional filter, with a kernel width of 3, operates on the time dimension. Following the convolution, an Exponential Linear Unit (ELU) activation function is applied to introduce non-linearity. Subsequently, a max-pooling layer with a stride of two is applied, downsampling X^t to half its initial length, optimizing the model's efficiency and mitigating potential memory constraints.

Traditionally, an encoder is responsible for transforming input data into a fixed-size vector, capturing its essential features. The decoder, on the other hand, is tasked with converting the context vector produced by the encoder into output data. For instance, in the original architecture of Informer, the decoder is utilized to generate predicted time series data. However, in this context, we refer to the decoder of Informer as a Window Encoder, as illustrated in Fig. 2. This is because it also ingests the W sequence and encodes it.

Algorithm 1 Anomaly Detection Model Training

Require:
Informer Encoder and Window Encoder E
Generator and Discriminator Gen and $Disc$
Datasets used for training $\mathcal{W}$
Iteration limit L

Initialize weights in E, Gen and $Disc$. Set $l \leftarrow 0$
do
  for ($t = 1$ to T) **do**
    $Z \leftarrow E(W_t, C, \vec{0})$
    $O_D, O_G \leftarrow Gen(Z), Disc(Z)$
    $\hat{Z} \leftarrow E(W_t, C, ||O_G - W_t||_2)$
    $\hat{O}_D \leftarrow Disc(\hat{Z})$
    Calculate L_G, L_D using Eq. (17), Eq. (18)
    Update weights of E by L_G and L_D
    $l \leftarrow l + 1$
  end for
while $l < L$

3.4. Adversarial reconstruction for detection

We now describe the anomaly detection training process in our model, summarized in Algorithm 1.

The Informer-based model allows us to accurately predict the reconstruction of each input time series window. It is very effective when the length of the time series is long. However, there are still some areas for improvement in the attention mechanism of the informer model. It still cannot accurately capture short-term trends, and if the bias is too small, it often misses anomalies [36]. Inspired by TranAD [41], we predict the reconstructed window in two stages to address this issue, and then we employed an adversarial training approach to enhance the model's adaptability and strengthen its resistance to a wide range of input sequences [37].

Stage 1 – Reconstruction. In Stage 1, our primary objective is to employ the previously mentioned Informer-based model as an encoder–decoder network to approximate the reconstruction of the input time series window W. Unlike the traditional Transformer architecture which is autoregressive in nature, the Informer decoder predicts the output through a forward process, eschewing the time-consuming dynamic decoding inherent in the conventional encoder–decoder structure. In an autoregressive decoding setup, the original Transformer model relies on previous steps at each stage of decoding. For time series tasks, when producing the timestamp at the t-th position, it takes into account the preceding $t-1$ timestamps that have already been generated. Following the token embedding and encoding module, the time series window W_t is mapped to a latent space vector representation Z. Subsequently, Z is independently fed into both the generator and discriminator. At this juncture, the roles of the generator and discriminator converge, with both aiming to ensure that the generated time series closely mirrors the input time series window. We employ the L_2-norm to define the reconstruction loss for the generator and discriminator in this stage:

$$L_G = \|O_G - W_t\|_2 \tag{14}$$

$$L_D = \|O_D - W_t\|_2 \tag{15}$$

Stage 2 – GAN-Based Training. In the GAN (Generative Adversarial Network) [50] framework, the goal of the generator is to produce data that is very similar to real data, while the discriminator's goal is to distinguish between real data and generated data as much as possible. During the training process, these two networks are adversarial to each other: the generator tries to deceive the discriminator into believing that the generated data is real, while the discriminator aims to become smart enough to recognize the counterfeit data produced by the generator. Now we use $\hat{O}_D$, outputs of the second stage, to define adversarial loss:

$$\min_{Gen} \max_{Disc} \left\| \hat{O}_D - W_t \right\|_2 \tag{16}$$

It should be underscored that in the first stage, both the generator and the decoder essentially act as decoders, as they are not adversarial. Their distinct roles emerge in the second stage. The two-stage training is adopted to utilize the first stage's reconstruction error $\|O_G - W_t\|_2$, which, when combined with the C sequence, is input into the Encoder. This enhances the error detection in the second stage, aiding in the extraction of short-term trends. This method counterbalances the Informer's tendency to prioritize long-term trends and inadvertently miss minor short-term shifts due to its sparse attention mechanism. To sum it up, here we develop a loss function that integrates both stages:

$$L_G = \frac{1}{n}\|O_G - W_t\|_2 + (1 - \frac{1}{n})\|\hat{O}_D - W_t\|_2 \tag{17}$$

$$L_D = \frac{1}{n}\|O_D - W_t\|_2 - (1 - \frac{1}{n})\|\hat{O}_D - W_t\|_2 \tag{18}$$

where n represents the training epoch. During the initial stages of training, the reconstruction loss is given a higher priority. This is essential to maintain training stability, especially when the generator and discriminator are not effectively reconstructing the input time series window. If the reconstruction is not accurate in the early stages, the input to the subsequent stage can be compromised. Therefore, the adversarial loss is assigned a lower weight initially. However, as training progresses and the output more accurately mirrors the input time series window, the weight of the adversarial loss is gradually increased. This approach ensures a balance between reconstruction accuracy and adversarial robustness as training continues [37]. Now, we define the anomaly score s used to determine whether the input sequence is anomalous.

$$s = \frac{1}{2}\left\|O_G - \hat{W}_t\right\|_2 + \frac{1}{2}\left\|\hat{O}_D - \hat{W}_t\right\|_2 \tag{19}$$

In line with established practices found in previous studies [32,35], we employ the Peak Over Threshold (POT) method [51] for a consistent and dynamic threshold selection to ensure equitable comparisons. In the detection results, if the anomaly score s_i of the anomaly label y_i for dimension i exceeds the threshold, then the entire multivariate time series of m dimensions is marked as anomalous.

Algorithm 2 Anomaly Classification Model Training

Require:
Pretrained Encoder E_{Pre}, Iteration limit N
Dataset for classification $\mathcal{W}^{\mathcal{P}}$

Initialize E_{Pre} with weights trained from anomaly detection, Set $n \leftarrow 0$
do
  for ($t = 1$ to T) **do**
    $P_c = \frac{1}{N_c}\sum_{i=1}^{N_c} E_{Pre}(W_{t_i}^P)$
    $p_c((W_t^P) = \text{softmax}(d(W_t^P, P_c))$
    Calculate L_P using Eq. (23)
    Update weights of E_{Pre} by L_P
  end for
while $n < N$

3.5. Prototypical networks for classification

Prototype Networks (PN) [16] emerge as a deep learning model specifically crafted to address the challenges of few-shot learning. In the realm of deep learning, many models mandate vast quantities of data for training. However, as observed, real-world applications often

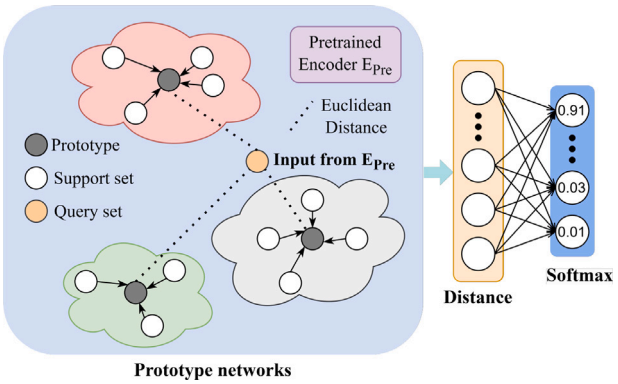


Fig. 4. Details of the Proposed Classification Module.

offer limited labeled data. Addressing this conundrum, researchers have unfurled a plethora of few-shot learning methodologies, among which stands the Prototype Networks. Serving as an efficient classification model, its core philosophy revolves around the notion that for every class, a prototype can be defined as the mean of the feature vectors of all samples within that class. This prototype can be perceived as the centroid or representative of that class.

The conceptual framework of PN incorporates the ideas of a Support Set and a Query Set. The former is a collection of samples used to delineate prototypes for each class, while the latter, employed during the training phase, serves to compute loss and subsequently update the network weights. The Query Set encompasses other labeled samples, distinct from those in the Support Set. For every sample within the Query Set, its distance to each class prototype is computed, serving as a foundation for classification. A complete training iteration materializes by using the samples in the Query Set and their predictions to calculate the loss and update network parameters. During testing, the feature vector of a novel sample is juxtaposed against all prototypes, consequently classifying it into the category represented by the nearest prototype.

The dataset used for classification is specifically partitioned and denoted as W^P, and this dataset solely comprises labeled anomalous multivariate time series. Within each category of anomaly, N samples are selected as the support set, and M samples are chosen as the query set. W_t^P is a single sample extracted from W^P. Each W_t^P has an associated class label. In the proposed architecture, there is flexibility to either solely deploy the anomaly detection module, which pinpoints anomalies, or to utilize a comprehensive suite of detection and classification models. The specific classification process is delineated as follows.

Post the training of anomaly detection, the weights of the Informer Encoder and Window Encoder (collectively denominated as E) are fixed, yielding a pre-trained encoder, E_{Pre}. Harnessing the principles of transfer learning, E_{Pre} is then employed as the starting point for the classification task. This approach capitalizes on the latent features previously discerned by the encoder during the anomaly detection phase. As a result, the classification task demands fewer labeled samples, simultaneously expediting the training process. Feature extraction via E_{Pre} produces $E_{Pre}(W_t^P)$ (whereas during the anomaly detection stage, Z is procured).

Firstly, to understand the core characteristics of each class, we compute the prototype for each class. For each class c, its prototype P_c is obtained by averaging the feature vectors of all samples in that class:

$$P_c = \frac{1}{N_c} \sum_{i=1}^{N_c} E_{Pre}(W_{t_i}^P) \tag{20}$$

where N_c represents the number of samples in class c. With these prototypes, when given a new sample x, the Euclidean distance between

Algorithm 3 Anomaly Classification Model Testing

Require:
Trained Encoder E_{Pre}, Gen and $Disc$
Encoder's stage 2 output $\hat{Z}$
Prototype Networks
Test Dataset $\hat{\mathcal{W}}$
Anomaly score s, Time series dimension i

for ($t = 1$ to T) **do**
  $O_D, O_G \leftarrow Gen(\hat{Z}), Disc(\hat{Z})$
  $\hat{O}_D \leftarrow Disc(\hat{Z})$
  $y = \bigvee_i \mathbb{1}(s_i \geq \mathrm{POT}(s_i))$
  if ($y > 0$) **then**
    $P_c = \frac{1}{N_c} \sum_{i=1}^{N_c} E_{Pre}(\hat{W}_{t_i})$
    $p_c((\hat{W}_t) = \mathrm{softmax}(d(\hat{W}_t, P_c))$
    $\hat{c}(\hat{W}_t) = \arg\max_c p_c(\hat{W}_t)$
    Return $\hat{c}$
  end if
  Return $\vec{0}$
end for

the sample and each prototype is computed to ascertain the class to which it most closely aligns,

$$d(W_t^P, P_c) = \|E_{Pre}(W_t^P) - P_c\|^2 \tag{21}$$

Directly utilizing raw distances for classification may not be optimal. Therefore, these distances are transformed into a probability distribution, offering a more distinct view of the sample's likelihood of belonging to a particular class. This conversion can be accomplished using the softmax function:

$$p_c(W_t^P) = \frac{\exp(-d(W_t^P, P_c))}{\sum_j \exp(-d(W_t^P, P_j))} \tag{22}$$

where $p_c(W_t^P)$ is the probability that sample W_t^P belongs to class c. During the training process, we want the model's predicted probabilities to be as close as possible to the true labels. Therefore, we use the cross-entropy loss to measure this discrepancy:

$$L_P = -\sum_c y_c \log(p_c(W_t^P)) \tag{23}$$

where y_c is the true label of sample W_t^P. It is a one-hot encoded vector, where only the position corresponding to the class of the sample W_t^P is set to 1, and all other positions are set to 0. Based on this loss, we can update the model parameters, making it predict classes more accurately in subsequent iterations. Finally, in testing or actual application, for a given sample W_t^P, we classify it by selecting the class with the highest probability:

$$\hat{c}(W_t^P) = \arg\max_c p_c(W_t^P) \tag{24}$$

Thus, $\hat{c}(W_t^P)$ becomes the model's predicted class label. The training algorithm for anomaly classification is summarized in Algorithm 2, while the comprehensive testing algorithm for both anomaly detection and classification is detailed in Algorithm 3. We also intuitively demonstrated the classification process in Fig. 4.

3.6. Algorithm summary

The following summarizes the operation process of the propose model. Our model is divided into two main steps: anomaly detection and anomaly classification.

Step 1 - Anomaly Detection (Binary Classification). The reconstruction-based model is trained using normal samples (up to the GAN module as illustrated in Fig. 2), and the reconstruction error of the input data is calculated to determine an appropriate threshold. If

Table 1
Dataset characteristics and anomaly ratios.

Dataset	Train size	Test size	Dimensions	Anomalies (%)
SMAP	135 183	427 617	25 (55)	13.13
MSL	58 317	73 729	55 (3)	10.72
SWaT	496 800	449 919	51 (1)	11.98
WADI	1 048 571	172 801	123 (1)	5.99

the difference exceeds this threshold, the time series will be marked as an anomaly.

Step 2 - Anomaly Classification (Multiclass Classification). The feature vectors of the anomalous samples are extracted using the encoder trained in Step 1. For each category, the mean of the feature vectors of all samples belonging to that category is calculated, resulting in the acquisition of the prototype vector for each category. Following this, the Euclidean distance for multiclass classification is calculated using both the prototype vector and the feature vector of the new sample. The nearest prototype vector's corresponding category will be the predicted category.

As we discussed earlier, in real industrial environments, the majority of data is from normal sensors. Therefore, when training a direct multi-classification model, the following potential risks may arise:

(1) **Model Bias Towards Normal Categories.** When normal samples dominate, the model may tend to predict normal categories, as this can achieve higher accuracy on the training data. This will lead to a decline in the model's ability to identify abnormal categories.
(2) **Reduced Differentiation Between Abnormal Categories.** The model will focus more on distinguishing between normal and abnormal, rather than on different abnormal categories, making it insensitive to subtle differences between abnormal categories.
(3) **Overfitting Abnormal Categories.** Due to the scarcity of abnormal samples, the model may overfit them, learning their noise rather than their true underlying patterns, which may result in a decline in generalization performance.
(4) **Difficulty in Evaluation.** When evaluating the model, common evaluation metrics may be dominated by the majority class, making the assessment of abnormal classification performance less convincing and reliable.

Therefore, we propose this two-step anomaly classification method. The advantage of this approach is that once the anomaly detection model is trained and fixed, the anomaly classification model can be updated at any time without the need to redo anomaly detection. This is very useful in practical applications, as new types of anomalies may emerge over time. in addition, by dividing the problem into normal and abnormal categories. In the first step, the model will focus more on distinguishing between these two main categories, capturing the features of abnormal samples even if they make up a small proportion of the entire dataset. Another benefit is that this can reduce the imbalance problem between multiple minority categories and one majority category. Once the abnormal samples are identified, the model can focus on classifying different abnormal types in the second step.

4. Experiments

4.1. Dataset description

In our experiment, we utilized four multivariate datasets that are publicly accessible. A summary of their attributes can be found in the Table 1, with the numbers in brackets denoting the count of sequences within the dataset repository. These standard benchmark datasets are employed to make a direct comparison between our approach and other methods.

- SMAP Dataset: This dataset comprises soil samples and telemetry data obtained using NASA's Mars rover [32].
- MSL Dataset: This dataset is akin to the SMAP dataset, but it focuses on the sensor and actuator data from the Mars rover. Notably, it has many sequences that are not particularly informative. Thus, we focus on the three significant sequences named A4, C2, and T1, as highlighted in [52].
- SWaT Dataset: Originating from a real-world water treatment plant, this dataset captures 7 days of standard operations and 4 days of anomalies. It encompasses sensor readings such as water levels and flow rates, along with actuator actions like the operation of valves and pumps [53].
- The WADI [54] Water Distribution dataset is sourced from a water distribution testbed, serving as an extension of the SWaT testbed. Spanning 16 days of continuous operations, it includes 14 days of standard operation and 2 days featuring attack scenarios. In total, the testbed comprises 123 sensors and actuators.

4.2. Data preprocessing

Before initiating the model training, we standardize the data to bolster the model's resilience. This data preprocessing method is applied to both the training and testing datasets. Given the maximum value $\max(X)$ and minimum value $\min(X)$ in the training set, the data point x is transformed as follows:

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)} \tag{25}$$

In this equation, x' represents the standardized data point. In this way, the original data is linearly scaled to a range of [0,1]. Because the ranges of different features in the original data vary greatly, we use this method to normalize the data, so that each feature has equal weight during model training.

4.3. Experimental details

Evaluation Metrics. To measure the effectiveness of our methodology, we utilize established metrics commonly employed in anomaly detection studies. These include Precision, Recall, and the F1 score, each contributing to a different perspective of evaluation as follows:

$$\text{Precision (Pre)} = \frac{TP}{TP + FP} \tag{26}$$

$$\text{Recall (Rec)} = \frac{TP}{TP + FN} \tag{27}$$

$$\text{F1 Score (F1)} = 2 \times \frac{Pre \times Rec}{Pre + Rec} \tag{28}$$

Anomaly detection outcomes fall into four categories. True Positives (TP) denote correct anomaly identifications. False Positives (FP) are normal instances mislabeled as anomalies. True Negatives (TN) accurately classify normal instances, while False Negatives (FN) misidentify anomalies as normal.

In real-world applications, the priority is capturing all true anomalies, accepting some false alarms. Hence, the focus is often on Recall and the overall F1 score, both emphasizing anomaly detection over precision.

Baseline Methods. We compare our model GIN with state-of-the-art models for multivariate time-series anomaly detection, including (1) forecasting-based models: LSTM-NDT [32], DAGMM [31], MSCRED [33], GDN [38], GTA [44]; (2) mixed-models: MTAD-GAT [40], and CAE-M [39]; (3) reconstruction-based models: PCA [24], AE [30], OmniAnomaly [35], MAD-GAN [36], LSTM-VAE [34], USAD [37] and TranAD [41]. The results in Table 2 are based on [41]. We retested using the same environment and updated results showing notable improvements over the article. Results for models not in this article are acquired from their original papers.

Table 2
Experimental results on SMAP, MSL, SWaT and WADI.

Method	SMAP			MSL			SWaT			WADI		
	Pre (%)	Rec (%)	F1	Pre (%)	Rec (%)	F1	Pre (%)	Rec (%)	F1	Pre (%)	Rec (%)	F1
Forecasting-based models												
LSTM-NDT	85.23	73.26	0.7879	62.88	100.0	0.7721	77.78	51.09	0.6167	1.38	78.23	0.0271
DAGMM	58.45	90.58	0.7105	49.11	55.62	0.5216	27.46	69.52	0.3937	54.44	26.99	0.3609
MSCRED	81.75	92.16	0.8664	89.12	98.62	0.9363	99.92	67.70	0.8072	25.13	73.19	0.3741
GDN	74.80	98.91	0.8518	93.08	98.92	0.9591	96.97	69.57	0.8101	97.50	40.19	0.5692
GTA	89.11	91.76	0.9041	91.04	91.17	0.9111	74.91	96.41	0.8379	74.56	90.50	**0.8176**
Mixed models												
MTAD-GAT	75.17	99.91	0.8583	79.17	98.24	0.8768	97.18	69.57	0.8109	28.18	80.12	0.4169
CAE-M	81.93	95.67	0.8827	77.51	100.0	0.8733	96.97	69.57	0.8101	27.82	79.18	0.4117
Reconstruction-based models												
PCA	29.37	24.14	0.2650	28.84	19.93	0.2357	24.92	21.63	0.2316	39.53	5.63	0.0986
AE	72.16	79.95	0.7586	71.66	50.08	0.5896	72.63	52.63	0.6103	34.35	34.35	0.3435
OmniAnomaly	74.16	97.76	0.8434	88.67	91.17	0.8989	97.82	69.57	0.8131	31.58	65.41	0.4260
MAD-GAN	81.57	92.16	0.8654	85.16	99.30	0.9169	95.93	69.57	0.8065	22.33	91.24	0.3588
LSTM-VAE	85.51	63.66	0.7298	52.57	95.46	0.6780	96.24	59.91	0.7385	87.79	14.45	0.2482
USAD	74.80	96.27	0.8419	79.49	99.12	0.8822	99.77	68.79	0.8143	64.51	32.20	0.4296
TranAD	80.43	99.99	0.8915	90.38	99.99	0.9494	97.6	69.97	0.8151	35.29	82.96	0.4951
Ours	84.99	98.53	**0.9126**	92.97	99.32	**0.9604**	98.91	76.84	**0.8649**	73.92	86.48	0.7971

Table 3
Analysis of the duration (in seconds) for each training epoch.

Method	SMAP	MSL	SWaT	WADI
Forecasting-based models				
LSTM-NDT	30.38	28.86	29.07	326.83
DAGMM	20.96	**18.05**	20.36	195.99
MSCRED	24.19	36.82	162.04	1483.96
GDN	65.46	95.71	48.87	3596.25
GTA	58.75	73.63	50.48	2044.03
Mixed models				
CAE-M	205.94	633.56	45.38	4457.16
Reconstruction-based models				
OmniAnomaly	29.76	23.44	31.23	234.29
MAD-GAN	32.53	28.98	30.65	323.84
USAD	28.36	25.46	27.26	291.43
TranAD	**17.74**	18.46	**1.93**	**165.80**
Ours	22.59	29.47	21.03	695.32

Hyper-parameter tuning. Our methods are implemented with PyTorch and trained with 13th Gen Intel(R) Core(TM) i9-13900HX and an NVIDIA 4070 GPU. The entire model is trained using a batch size of 32. Early-stopping criteria are used for avoiding over-fitting. For all datasets, the model input embedding dimension is set to 128. Through empirical testing, we have set the sliding window size to 32. We set the parameter K for graph learning to 30, 15, 30, and 30 for SMAP, MSL, SWaT, and WADI, respectively. The encoder and decoder layers of the Informer model we use are 2. For the multi-head attention mechanism of SP-GAT and Informer, the number of heads is set to 8. We employ the AdamW [55] optimizer to refine our proposed methods, initiating with a learning rate of 0.01 which diminishes by a factor of 0.5 each epoch. We utilize the POT algorithm to set the anomaly threshold over the testing dataset. To train our model, we divide the training time-series into 80% training data and 20% testing data.

4.4. Experimental results

In Table 2, we present the performance of anomaly detection, measured by precision, recall, and F1-score. Our method GIN achieved the highest F1 scores: 0.9126 for SMAP, 0.9604 for MSL, 0.8649 for SWaT, and secured the second highest F1 score for WADI, which was 0.7971. On the WADI dataset, our performance was only slightly lower compared to the best-performing GTA model, while GIN showed a 22.79% improvement in performance compared to the third-ranked GDN model. On the other datasets, the performance of our model also surpasses the state-of-the-art methods. From the results and methods, several insights emerge: Classical unsupervised techniques like KNN and PCA are outperformed by basic deep learning techniques such as AE, LSTM-VAE, and MAD-GAN. The deep learning models, especially those with recurrent mechanisms (like RNN, GRU, and LSTM), excel because they can model long sequences and grasp temporal context dependencies better than traditional methods. DAGMM is designed for multivariate data that lacks temporal information. This means it could only consider single timestamp rather than a sliding time series window. This design choice makes it less effective for modeling the temporal dependencies vital for time series anomaly detection. Many of the discussed methods rely on RNNs to capture temporal dependencies. Some, like LSTM-NDT, are deterministic, missing out on the stochastic nature of time series. Others, like LSTM-VAE, blend LSTM and VAE for sequence modeling but overlook temporal dependencies among latent variables. OmniAnomaly addressed this gap. The OmniAnomaly, CAE-M and MSCRED models utilize sequential observations, ensuring the preservation of temporal information. However, they struggle to identify anomalies that are very similar to normal patterns. MAD-GAN uses adversarial training for time series reconstruction. However, RNNs inherently process data sequentially. This means they rely on past hidden states to retain past information, which can restrict their ability to model long sequences. In contrast, the Transformer model uses a non-sequential approach, and its self-attention mechanism reduces the context distance between any part of a time series to one. This is vital as more historical data can provide richer pattern information. Approaches like USAD and MTAD-GAT employ attention mechanisms to concentrate on particular data features. Additionally, these models aim to capture long-term trends by fine-tuning their neural network weights. They rely on a localized window of data for the reconstruction task, rather than using the entire sequence. GDN, a graph learning-based anomaly detection method, according to the experimental results from the WADI dataset, which has many dimensions, it can be seen that due to its excellent graph structure modeling, it performs better than most baseline methods. However, due to its lack of efficient temporal modeling models like Transformers, GDN has its limitations. On the other hand, MTAD-GAT assumes all sensors are interconnected, leading to a complete graph, which is not always representative of real-world scenarios. Although TranAD resolves many of the issues, its lack of investigation into the correlations of multi-dimensional data leads to mediocre performance on datasets with a larger number of dimensions. The graph convolutional structure proposed in GTA effectively learns the relationships among the 123 dimensions in the WADI dataset.

Table 4
Ablation results of our model and its variants.

Method	SMAP				WADI			
	Pre (%)	Rec (%)	F1-score	Training time (s)	Pre (%)	Rec (%)	F1-score	Training time (s)
GIN	84.99	98.53	0.9126	21.03	73.92	86.48	0.7971	695.32
w/o SP-GAT	75.01	98.36	0.8511	**0.85**	65.17	69.44	0.6724	**122.14**
w/o informer	81.81	95.75	0.8823	70.37	66.52	77.12	0.7143	1720.28
w/o GAN-based training	73.12	97.31	0.8350	19.59	62.47	70.15	0.6609	626.81

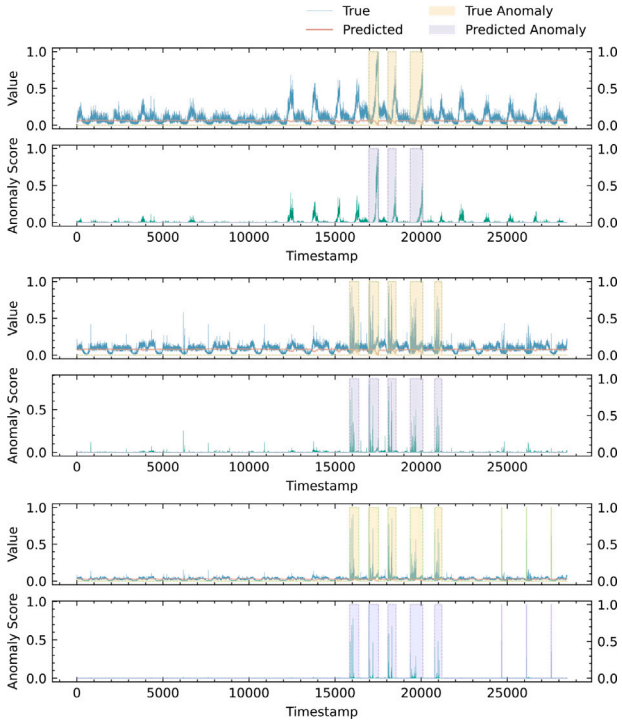


Fig. 5. Visualization of the anomaly detection process.

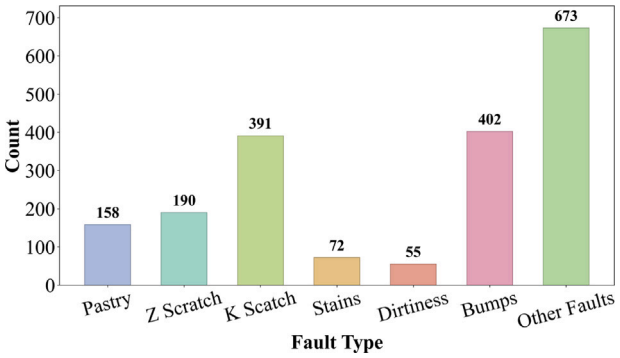


Fig. 6. Distribution of fault classes.

Our proposed SP-GAT module significantly reduces the number of edges in the graph, greatly improving efficiency but inevitably falling marginally short in learning the relationships between sensors. Therefore, GTA's F1 score on the WADI dataset is slightly higher than that of GIN. In summary, while various methods offer different approaches to anomaly detection in time series data, our proposed method proves to be notably superior on the studied datasets.

We also visualized the anomaly scores s from a subset of dimensions in the SMAP dataset, as shown in Fig. 5. In this representation, the larger the peak of the anomaly score, the more likely an anomaly is present. Our model uses the POT method to predict anomalies based on anomaly scores, and the anomalies predicted and the true anomalies that occur are marked in the figure, demonstrating the excellent anomaly detection capability of GIN. Additionally, time steps with significant changes (noise) often have higher anomaly scores. Therefore, the two-stage adversarial training approach we employed proved effective: by assigning varying weights to different dimensions, it enhanced adaptability and robustness to noise. This is evident in the figure, where the fluctuation of anomaly scores varies across different dimensions. Concurrently, the timestamps of anomalies across various dimensions are highly consistent, indicating strong inter-dimensional correlations. This observation underscores the significance of the graph learning method we proposed.

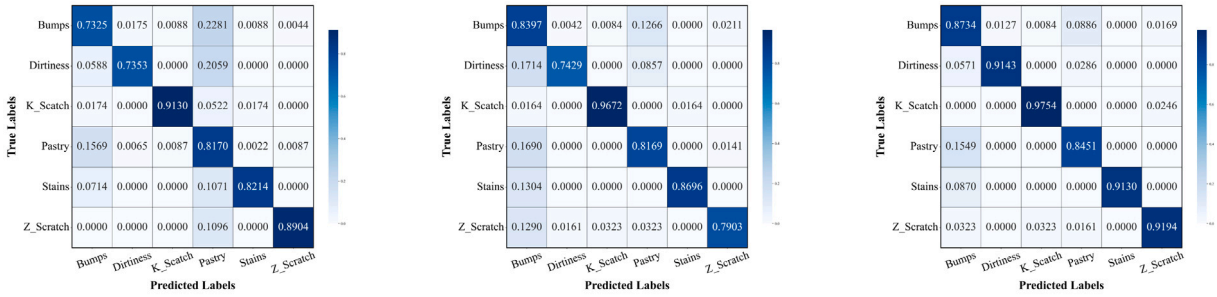
In Table 3, we have tabulated the training time for each model over a single epoch and made comparisons. It is evident that, because TranAD uses Transformer as its core component, it achieved the shortest training time on three datasets, especially standing out on the SWaT and WADI datasets. Our model, due to the incorporation of graph learning, has a slightly longer training duration. However, through sparse topological modeling and the ProbSparse attention mechanism in the Informer encoder, training speed of our model has significantly improved compared to GDN and GTA, which also use graph learning. While the excellent graph learning methods of GDN and GTA accurately capture the relationships between sensors, their overly intricate topological structures reduce time efficiency. In GTA, the focus on reducing time complexity mainly lies in the Transformer module used for extracting temporal features, but the overall time consumed by its model is still roughly three times more than our model. In the subsequent ablation studies, we will discuss this issue by components.

4.5. Ablation studies

To better understand the efficacy of each component in our approach, we have conducted a step-by-step analysis by sequentially excluding different elements and observing the impact on model performance using the SMAP and WADI datasets. Firstly, we examined the importance of SP-GAT in modeling sensor dependencies. For this, we used the original time series data directly as input for the Informer-based encoder. Secondly, We investigate the enhancement of Informer in time series feature extraction by replacing it with a vanilla Transformer model. Lastly, We modified the training process to a single stage, training solely with reconstruction loss, to compare with the two-stage GAN-based training. These results are encapsulated in Table 4 and lead us to several observations. (1) There is a notable difference between GIN and the variant that excludes our proposed graph attention mechanism. This disparity underscores the importance of topological structure modeling. However, even though the detection performance has decreased, the training time on multi-dimensional datasets like WADI has been significantly reduced compared to GIN, due to the reduction in modeling across input data dimensions. Therefore, in real-world scenarios, choices can be made based on needs. (2) The Informer-based enocoder outperforms its counterparts in sequence modeling, highlighting the critical role of the improved self-attention mechanism. In addition, this attention mechanism has proven effective, greatly reducing the training time. (3) The impact of GAN-based training is particularly crucial. The results on the WADI dataset exhibited significant variations primarily because the features of its anomalous

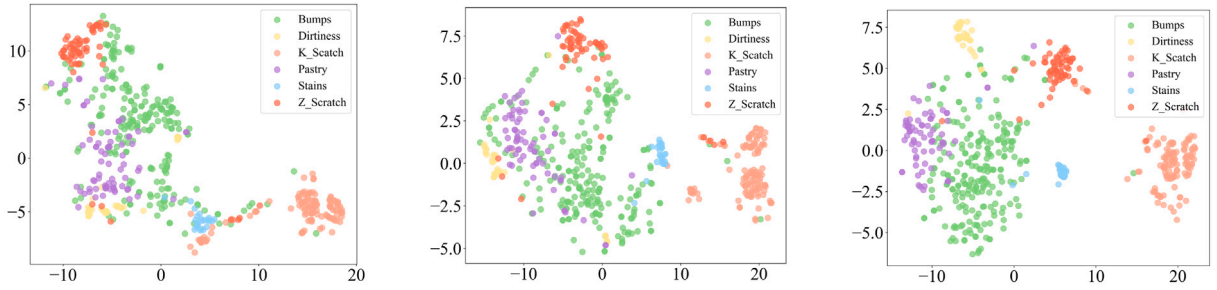
Table 5
Performance comparison between GIN and other models in classification tasks.

Method	DTW			Resnet			Our model		
Fault types	Pre (%)	Rec (%)	F1-score	Pre (%)	Rec (%)	F1-score	Pre (%)	Rec (%)	F1-score
Bumps	68.16	73.25	0.7061	86.52	83.97	0.8522	89.33	95.36	**0.9224**
Dirtiness	78.12	73.53	0.7576	92.86	74.29	0.8254	96.97	91.43	**0.9412**
K scatch	94.59	91.30	0.9292	96.58	96.83	0.9670	97.52	96.72	**0.9712**
Pastry	83.15	81.70	0.8242	62.37	81.69	0.7073	88.71	77.46	**0.8271**
Stains	82.09	82.32	0.8220	90.91	86.96	0.8889	100.0	86.96	**0.9302**
Z scratch	92.86	89.04	**0.9091**	89.09	79.03	0.8376	90.16	88.71	0.8943
Accuracy	0.8113			0.8545			**0.9200**		



(a) DTW (b) Resnet (c) Our model (GIN)

Fig. 7. Results of the confusion matrices.



(a) DTW (b) Resnet (c) Our model (GIN)

Fig. 8. Visualized Classification Results by t-SNE.

data closely resemble those of the normal data. Incorporating adversarial loss combined with a single-stage reconstruction error helps amplify the anomaly scores and detect such subtle anomalies. It is only natural that the training time for a two-stage process is longer than a single-stage reconstruction. However, this difference is insignificant compared to the importance of detecting subtle anomalies. Furthermore, these findings reaffirm that each component of our method is critical and collectively makes the framework potent for multivariate time series anomaly detection.

4.6. Case study for anomaly classification

In the anomaly classification module, our goal is to further identify the type of anomaly once it has been detected. This offers a more tangible relevance, making the model particularly applicable in industrial scenarios such as photovoltaic fault diagnosis and factory equipment malfunctions. We employ an industrial defect dataset [56] comprising 27 variables, thereby constituting 27 dimensions. This dataset encompasses seven types of anomalies: Pastry, Z Scratch, K Scratch, Stains, Dirtiness, Bumps and Other Faults, as illustrated in Fig. 6. Although the dataset lacks timestamps, preventing the formation of a comprehensive multivariate time series, our anomaly detection models have previously demonstrated robust capability in extracting multivariate time series features. Hence, we opted for this dataset to validate classification performance. It mirrors real-world scenarios more closely, and given its modest size, it is well-suited for investigating small sample learning performance. We preprocess the data for each anomaly type following our prior methodologies, segmenting them into sliding windows. We compare the classical time series classification method DTW [57] and the recent deep learning approach ResNet [58] as baseline methods against the GIN baseline. Both of these methods directly perform multiclass classification on the input data. Specifically, the DTW method initially measures the similarity between two time series and then clusters them using KNN. ResNet utilizes its deep residual learning framework to extract temporal patterns from one-dimensional time series data through its convolutional layers, ultimately achieving effective classification. In Table 5, We provide the precision, recall, and F1 scores for each method across six types of faults, excluding 'Other Faults'. Additionally, we have incorporated classification accuracy as an enhanced evaluation metric to offer a more comprehensive assessment. To facilitate a more thorough comparison of the three methods, we employ confusion matrices and 2D t-SNE visualizations. These are depicted respectively in Figs. 7 and 8. From the results, it is evident that our classification method significantly outperforms the previous baseline methods. However, the detection accuracy of our model for the 'Pastry' and 'Bumps' anomaly types is comparatively lower than the other categories. This can be attributed to the fact that the 27-feature input sequence does not manifest pronounced consistency when these faults occur, making detection challenging. Their respective data points

in the t-SNE visualization also appear dispersed. Utilizing a prototypical network, even with limited samples, we still achieved remarkable classification performance. In summary, the transfer anomaly detection model we employed has attained a classification accuracy of 92%. We believe that future research will introduce more advanced and efficient methodologies addressing this issue.

5. Conclusion

We proposed GIN, an anomaly detection and classification model based on the Transformer. This model employs the SP-GAT mechanism, which we propose, to autonomously learn the topological structures and spatial dependencies of various sensors. To capture the long-term temporal dependencies in time series, we incorporated the time series prediction model Informer as a substitute for the vanilla Transformer. This change enhances inference speed without compromising the performance of the model. Extensive experiments conducted on four real-world datasets have shown that our approach outperforms other state-of-the-art methods in terms of prediction accuracy and time efficiency. Furthermore, we provide a case study to demonstrate how our proposed model can be used for anomaly classification following anomaly detection. For future work, our objective is to enable online learning for this method and to ensure its versatile application across industrial fault detection domains and mobile IoT environments.

CRediT authorship contribution statement

Chaoyang Wang: Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft. **Guangyu Liu:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Project administration, Resources, Supervision, Visualization, Writing – review & editing.

Declaration of competing interest

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

Data availability

The data that has been used is confidential.

Acknowledgments

This work was funded by National Natural Science Foundation of China (Grant No. 62273124 and Grant No. 62002315) and Zhejiang Provincial Science and Technology Plan Project of China (Grant No. 2023C01117)

References

[1] H. Xu, Z. Sun, Y. Cao, H. Bilal, A data-driven approach for intrusion and anomaly detection using automated machine learning for the Internet of Things, Soft Comput. 27 (19) (2023) 14469–14481.
[2] T. Reiss, Y. Hoshen, Mean-shifted contrastive loss for anomaly detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, No. 2, 2023, pp. 2155–2162.
[3] J.D. Hamilton, Time Series Analysis, Princeton University Press, 2020.
[4] M. Zheng, J. Man, D. Wang, Y. Chen, Q. Li, Y. Liu, Semi-supervised multivariate time series anomaly detection for wind turbines using generator SCADA data, Reliab. Eng. Syst. Saf. 235 (2023) 109235.
[5] H.S. Dhiman, D. Deb, S.M. Muyeen, I. Kamwa, Wind turbine gearbox anomaly detection based on adaptive threshold and twin support vector machines, IEEE Trans. Energy Convers. 36 (4) (2021) 3462–3469, http://dx.doi.org/10.1109/TEC.2021.3075897.
[6] A. Mellit, S. Kalogirou, Assessment of machine learning and ensemble methods for fault diagnosis of photovoltaic systems, Renew. Energy 184 (2022) 1074–1090.
[7] G. Li, J.J. Jung, Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges, Inf. Fusion 91 (2023) 93–102.
[8] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[9] M. Jin, H.Y. Koh, Q. Wen, D. Zambon, C. Alippi, G.I. Webb, I. King, S. Pan, A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection, 2023, arXiv preprint arXiv:2307.03759.
[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[11] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, L. Sun, Transformers in time series: A survey, 2022, arXiv preprint arXiv:2202.07125.
[12] C. Yun, S. Bhojanapalli, A.S. Rawat, S.J. Reddi, S. Kumar, Are transformers universal approximators of sequence-to-sequence functions? 2019, arXiv preprint arXiv:1912.10077.
[13] Z. Gao, X. Shi, H. Wang, Y. Zhu, Y.B. Wang, M. Li, D.-Y. Yeung, Earthformer: Exploring space-time transformers for earth system forecasting, Adv. Neural Inf. Process. Syst. 35 (2022) 25390–25403.
[14] L. Cai, K. Janowicz, G. Mai, B. Yan, R. Zhu, Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting, Trans. GIS 24 (3) (2020) 736–755.
[15] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G.-J. Qi, H. Xiong, Spatial-temporal transformer networks for traffic flow forecasting, 2020, arXiv preprint arXiv:2001.02908.
[16] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[17] Z.Z. Darban, G.I. Webb, S. Pan, C.C. Aggarwal, M. Salehi, Deep learning for time series anomaly detection: A survey, 2022, arXiv preprint arXiv:2211.05244.
[18] M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, W. Song, Gated transformer networks for multivariate time series classification, 2021, arXiv preprint arXiv:2103.14438.
[19] N.M. Foumani, L. Miller, C.W. Tan, G.I. Webb, G. Forestier, M. Salehi, Deep learning for time series classification and extrinsic regression: A current survey, 2023, arXiv preprint arXiv:2302.02515.
[20] M. Rußwurm, M. Körner, Self-attention for raw optical satellite time series classification, ISPRS J. Photogramm. Remote Sens. 169 (2020) 421–435.
[21] K. Kingsbury, P. Alvaro, Elle: Inferring isolation anomalies from experimental observations, 2020, arXiv preprint arXiv:2003.10554.
[22] P. Boniol, J. Paparrizos, T. Palpanas, M.J. Franklin, SAND: streaming subsequence anomaly detection, Proc. VLDB Endow. 14 (10) (2021) 1717–1729.
[23] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus, B. Furht, Anomaly detection in medical wireless sensor networks using SVM and linear regression models, Int. J. E-Health Med. Commun. (IJEHMC) 5 (1) (2014) 20–45.
[24] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang, A novel anomaly detection scheme based on principal component classifier, in: Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, IEEE Press, 2003, pp. 172–179.
[25] A. Patcha, J.-M. Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, Comput. Netw. 51 (12) (2007) 3448–3470.
[26] S. Kanarachos, J. Mathew, A. Chroneos, M. Fitzpatrick, Anomaly detection in time series data using a combination of wavelets, neural networks and Hilbert transform, in: 2015 6th International Conference on Information, Intelligence, Systems and Applications, IISA, IEEE, 2015, pp. 1–6.
[27] T.R. Bandaragoda, K.M. Ting, D. Albrecht, F.T. Liu, J.R. Wells, Efficient anomaly detection by isolation using nearest neighbour ensemble, in: 2014 IEEE International Conference on Data Mining Workshop, IEEE, 2014, pp. 698–705.
[28] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, pp. 413–422.
[29] A.H. Yaacob, I.K. Tan, S.F. Chien, H.K. Tan, Arima based network anomaly detection, in: 2010 Second International Conference on Communication Software and Networks, IEEE, 2010, pp. 205–209.
[30] C.C. Aggarwal, C.C. Aggarwal, Probabilistic and statistical models for outlier detection, Outlier Anal. (2013) 41–74.
[31] B. Zong, Q. Song, M.R. Min, W. Cheng, C. Lumezanu, D. Cho, H. Chen, Deep autoencoding gaussian mixture model for unsupervised anomaly detection, in: International Conference on Learning Representations, 2018.
[32] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 387–395.
[33] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, N.V. Chawla, A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, 2019, pp. 1409–1416.
[34] D. Park, Y. Hoshi, C.C. Kemp, A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder, IEEE Robot. Autom. Lett. 3 (3) (2018) 1544–1551.
[35] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, 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, 2019, pp. 2828–2837.

[36] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, S.-K. Ng, MAD-gan: Multivariate anomaly detection for time series data with generative adversarial networks, in: International Conference on Artificial Neural Networks, Springer, 2019, pp. 703–716.
[37] J. Audibert, P. Michiardi, F. Guyard, S. Marti, M.A. Zuluaga, Usad: Unsupervised anomaly detection on multivariate time series, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3395–3404.
[38] A. Deng, B. Hooi, Graph neural network-based anomaly detection in multivariate time series, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 5, 2021, pp. 4027–4035.
[39] Y. Zhang, Y. Chen, J. Wang, Z. Pan, Unsupervised deep anomaly detection for multi-sensor time-series signals, IEEE Trans. Knowl. Data Eng. (2021).
[40] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, Q. Zhang, Multivariate time-series anomaly detection via graph attention network, in: 2020 IEEE International Conference on Data Mining, ICDM, IEEE, 2020, pp. 841–850.
[41] S. Tuli, G. Casale, N.R. Jennings, Tranad: Deep transformer networks for anomaly detection in multivariate time series data, 2022, arXiv preprint arXiv:2201.07284.
[42] M. Ma, L. Han, C. Zhou, BTAD: A binary transformer deep neural network model for anomaly detection in multivariate time series data, Adv. Eng. Inform. 56 (2023) 101949.
[43] J. Xu, H. Wu, J. Wang, M. Long, Anomaly transformer: Time series anomaly detection with association discrepancy, 2021, arXiv preprint arXiv:2110.02642.
[44] Z. Chen, D. Chen, X. Zhang, Z. Yuan, X. Cheng, Learning graph structures with transformer for multivariate time-series anomaly detection in IoT, IEEE Internet Things J. 9 (12) (2021) 9179–9189.
[45] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.
[46] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, et al., Graph attention networks, Stat 1050 (20) (2017) 10–48550.
[47] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, I. Cohen, Pearson correlation coefficient, in: Noise Reduction in Speech Processing, Springer, 2009, pp. 1–4.
[48] Y. Ye, S. Ji, Sparse graph attention networks, IEEE Trans. Knowl. Data Eng. 35 (1) (2021) 905–916.
[49] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 12, 2021, pp. 11106–11115.
[50] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014).
[51] A. Siffer, P.-A. Fouque, A. Termier, C. Largouet, Anomaly detection in streams with extreme value theory, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1067–1075.
[52] T. Nakamura, M. Imamura, R. Mercer, E. Keogh, Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives, in: 2020 IEEE International Conference on Data Mining, ICDM, IEEE, 2020, pp. 1190–1195.
[53] A.P. Mathur, N.O. Tippenhauer, Swat: A water treatment testbed for research and training on ics security, in: 2016 International Workshop on Cyber-Physical Systems for Smart Water Networks, CySWater, IEEE, 2016, pp. 31–36.
[54] C.M. Ahmed, V.R. Palleti, A.P. Mathur, WADI: a water distribution testbed for research in the design of secure cyber physical systems, in: Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks, 2017, pp. 25–28.
[55] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[56] M. Buscema, S. Terzi, W. Tastle, Steel plates faults, 2010, http://dx.doi.org/10.24432/C5J88N, UCI Machine Learning Repository.
[57] V. Niennattrakul, C.A. Ratanamahatana, On clustering multimedia time series data using k-means and dynamic time warping, in: 2007 International Conference on Multimedia and Ubiquitous Engineering, MUE'07, IEEE, 2007, pp. 733–738.
[58] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (4) (2019) 917–963.