International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-024-02212-5

ORIGINAL ARTICLE

Uni-directional graph structure learning-based multivariate time series anomaly detection with dynamic prior knowledge

Shiming He[1] · **Genxin Li**[1] · **Jin Wang**[1] · **Kun Xie**[2] · **Pradip Kumar Sharma**[3]

Received: 29 November 2023 / Accepted: 8 May 2024

Abstract
In the Internet of Things (IoT) system, sensors generate a vast amount of multivariate time series data and transmit it to the data center for aggregation and analysis. However, due to equipment failure or attacks, the collected data may contain anomalies, which in turn affect the overall performance and reliability of IoT services. Therefore, an effective multivariate time series anomaly detection (MTSAD) method is a crucial issue to ensure the quality of service. Graph structure learning (GSL)-based methods become a promising technology in MTSAD, which learns an optimal graph structure joint with the anomaly detection task. However, most existing methods disregard the causal and dynamic relationships between sensors during the processing of IoT and assume that the data is devoid of any missing values. Therefore, we propose a uni-direction graph structure learning-based multivariate time-series anomaly detection with dynamic prior knowledge (DPGLAD), which learns the uni-directional relationships between sensors under the constraint of the dynamic prior graph and utilizes diffusion convolutional recurrent neural networks (DCRNN) based on timestamp mask to extract temporal and spatial features. Extensive experiments show that our method has better detection performance and shorter training times than state-of-the-art techniques on four real-world datasets. Compared with the best GSL-based method GTA, DPGLAD achieves 4.16–7.29% more F1-score.

Keywords Multivariate time series · Anomaly detection · Graph structure learning · Graph neural network · Dynamic prior graph · Uni-directional graph structure

This work is supported by the Open Fund of Key Laboratory of Safety Control of Bridge Engineering, Ministry of Education (Changsha University of Science Technology) under Grant 21KB07.

✉ Jin Wang
jinwang@csust.edu.cn

Shiming He
smhe_cs@csust.edu.cn

Genxin Li
mapleleavesli@stu.csust.edu.cn

Kun Xie
xiekun@hnu.edu.cn

Pradip Kumar Sharma
pradips@ieee.org

[1] Key Laboratory of Safety Control of Bridge Engineering, Ministry of Education, Changsha University of Science and Technology, Changsha 410114, China

[2] College of Computer Science and Electronics Engineering, Hunan University, Changsha 410082, China

[3] Computing Science, University of Aberdeen, Aberdeen AB24 3FX, UK

1 Introduction

In the Internet of Things (IoT) system, sensors generate large amounts of time series data [8]. Anomalies in the data may indicate device malfunctioning or system attacks. If the anomalies are not detected in time, they may result in economic losses [9, 26]. Thus, anomaly detection plays an important role in IoT systems. In real applications, multivariate indicators.[1] are collected to reflect the overall status of a system [7, 16] A multivariate time series (MTS) example from an industrial IoT system is presented in Fig. 1, which includes data from seven different sensors installed in a tap water treatment system. These sensors are Flow meter $FIT101$, level transmitter $LIT101$, motorized valve $MV101$, sump pump $P101$, backup sump pump $P102$, conductivity analyzer $AIT201$, and pH analyzer $AIT202$.

Various indicators affiliated with the same system interrelate with each other, making it so that a sudden shift in several indicators may not be indicative of system failure.

[1] Here, "indicator" refers to the time series of a particular variable.

Published online: 24 May 2024

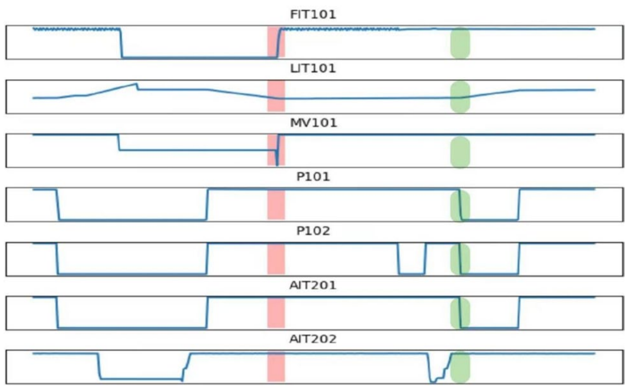

Fig. 1 A multivariate time series example of an industrial IoT system that showcases both normal (represented by green) and abnormal (represented by red) values

As shown in Fig. 1, despite the occurrence of a sudden decline in $P101$, $P102$, and $AIT201$ during the green segment, the system remains in a healthy status. This is due to the consistent trend of these three indicators. In contrast, the red segment shows an inconsistent pattern in $MV101$ compared to all other indicators, indicating system malfunction. Therefore, in the realm of multivariate time series anomaly detection (MTSAD), it is crucial to discover the presence of potential dependencies among the indicators [8].

Recently, graph neural networks (GNNs) [22, 23], especially spatio-temporal graph neural networks have garnered significant attention due to their ability to model inter-indicator relationships with satisfactory results. However, most GNNs [6, 28] presuppose an explicit graph structure or treat the graph structure as a complete graph. In real-world settings, the graph structure may be unavailable, rendering the inter-dependencies between sensors unknown.

To handle unknown graph structure, MTSAD grounded in Graph Structure Learning (GSL) [2, 3, 5]. It is a promising method that effectively acquires knowledge about the concealed graph structure in conjunction with the downstream GNNs task. Figure 2 visually illustrates the typical GSL-based anomaly detection method process. The graph structure learner generates a graph structure. This graph structure is then fed into the detection model, leading to a joint update of parameters in both the detection model and the graph structure learner. In this way, the graph structure is iteratively refined. This iterative parameter update scheme can obtain a more optimal graph structure that aligns with the requirements of the downstream anomaly detection task.

Nonetheless, the graph structure learning-based anomaly detection method still confronts numerous challenges.

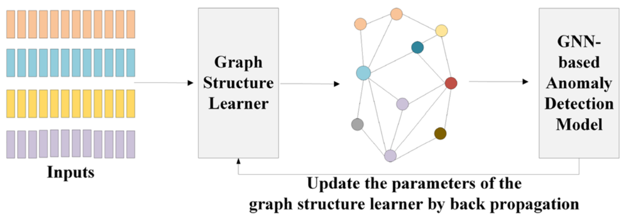
Inputs
Graph Structure Learner
GNN-based Anomaly Detection Model
Update the parameters of the graph structure learner by back propagation

Fig. 2 A general framework of Graph Structure Learning-based anomaly detection

- **Most of the present graph structure learning-based MTSAD** [3, 5] **method only captures undirected dependencies, leaving uni-directional dependencies unaccounted for. The existing uni-directional graph structure** [2] **learning-based MTSAD method has considerable overhead.** Several anomaly detection based on graph structure learning calculate node similarity to construct an undirected graph. However, there is a special processing of IoT. The upstream status decides the downstream operation. The downstream sensor depends on the upstream sensor. The relationship between sensors is uni-directional. Although Graph Learning with Transformer for Anomaly detection (GTA) [2] employs a fully parameterized graph learning method to obtain uni-directional graphs, the training time required is too lengthy and inefficient. Therefore, an efficient graph structure learning method is needed to extract the one-way dependence between sensors.
- **The static nature of the prior graph limits its ability to represent the dynamic relationship between sensors.** To improve the quality of learned graph structure, prior information is often provided by a prior graph. Existing methods utilize all raw data as input to obtain the K nearest neighbor (KNN) graph as prior knowledge, which is a static graph. However, the relationship between sensors changes over time. Using only a static prior graph can not adapt to varying data features and task requirements.
- **The existing methods are susceptible to data with missing values.** In current works, it is commonly assumed that training data is entirely normal and devoid of any missing values [11]. However, in the real world, collected data often contain missing values, especially when dealing with large volumes of data.

To address the three challenges, we explore a uni-direction graph structure learning-based multivariate time series anomaly detection with dynamic prior knowledge (DPGLAD), which utilizes uni-directional graph structure learning to model relationships between sensors under the constraint of the dynamic prior graph, and diffusion convolutional recurrent neural networks (DCRNN) based on timestamp mask to extract temporal and spatial features. Our major contributions can be summarized as follows:

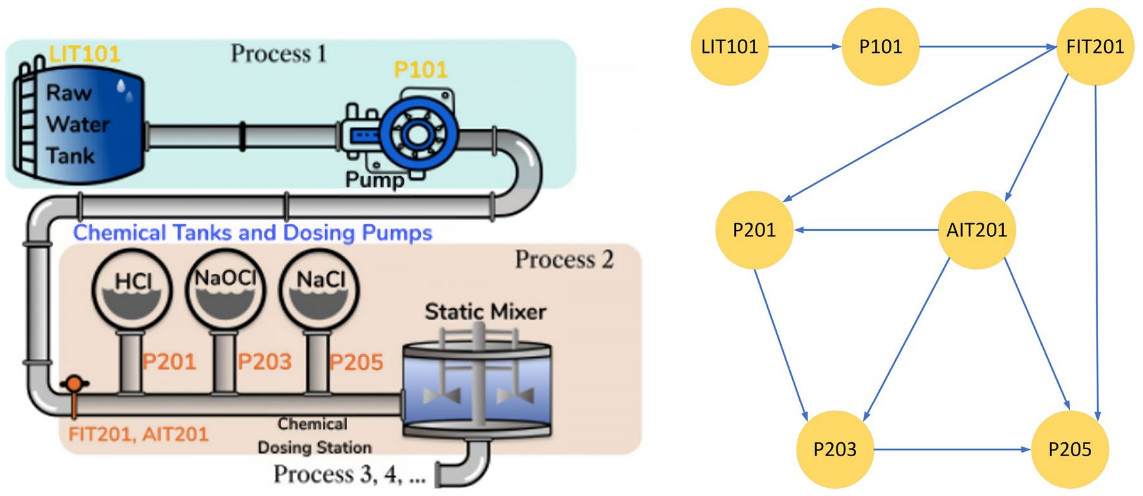

Fig. 3 **Left**: The first two processes of Secure Water Treatment (SWAT) [4] testbed. **Right**: The abstract relationships between sensors

- To mine one-way dependencies between sensors at a low cost, we utilize a lightweight uni-directional graph structure (UGL) learning method. UGL learns two embedding vectors for each sensor to represent the source node and destination node of the edge, respectively, instead of one embedding vector or one feature vector.
- To improve the quality of the learning graph, we propose a dynamic prior graph generation to obtain a dynamic prior graph changing over time instead of a static prior graph, which provides accurate prior information and adapts to varying data features and task requirements.
- To effectively handle data with missing values, we propose a timestamp mask-based diffusion convolutional recurrent neural network (DCRNN) to actively mask some values and robustly predict the next value of the time series. The prediction error is exploited to detect anomalies.
- Extensive experiments conducted on four public and real-world datasets show that our method has better detection performance and shorter training times than state-of-the-art techniques. Compared with the best GSL-based method GTA, DPGLAD achieves 4.16–7.29% more F1-score.

The subsequent sections of this paper are organized as follows. Section 2 explains the background for our work. Section 3 presents an overview of the related work. Section 4 provides a presentation of the problem description and the preliminaries. Section 5 provides a comprehensive explanation of DPGLAD, including both a general overview and a step-by-step guide. In Sect. 6, we conduct experiments to evaluate the performance and efficiency of the model. Finally, our work is concluded in Sect. 7.

2 Background

In this section, we take an example to display the one-way dependence between sensors. An illustration of the first two processes in the Safe Water Treatment (SWAT) testbed is shown in the left of Fig. 3. In process 1, the water level sensor (LIT101) in the tank has a causal effect on the pump control system (P101). When the water level in the tank drops below a certain threshold, it triggers the pump control system to activate and start pumping water for further treatment. Similarly, in process 2, the chemical dosing system (P201, P203, P205) is influenced by the data from the water quality sensor (FIT201, AIT201). Once the water quality meets the required standards, the addition of chemicals will be ceased.

In this simplified illustration, it is evident that the processes have a uni-directional influence on each other, indicating the presence of a uni-directional relationship among the sensors. We can obtain a uni-directional graph for these sensors as shown in the right of Fig. 3.

3 Related work

In this section, we review anomaly detection methods based on temporal feature, graph neural networks, and graph structure learning.

3.1 Temporal feature-based anomaly detection methods

The recurrent neural networks (RNNs) and their variants are prevalent in MTSAD due to their natural aptitude for handling time series data. LSTM-NDT [10] utilizes Long

Short-Term Memory (LSTM) to learn the temporal features and generate predictions. It collects the error between predicted and actual values to form an error vector, which is exponentially averaged and weighted to compute a threshold for anomaly detection. In addition, various combined models of recurrent neural networks and generative models are employed in MTSAD for reconstructing time series. For instance, Omnianomaly [20] employs a stochastic recurrent neural network to effectively capture and represent normal patterns, thereby facilitating the reconstruction of the observed data. DAGMM [29] trains both deep autoencoder and Gaussian mixture models to produce reconstruction errors for detecting anomalies. MAD-GAN [12] employs LSTM to establish a Generative Adversarial Network (GAN) framework that effectively captures the temporal correlation of time series distributions, and identifies anomalies through discrimination and reconstruction processes. USAD [1] adversarially trains an encoder-decoder framework to achieve fast and efficient training. LSTM-VAE [17] maps multimodal observations and temporal dependencies' relations to the latent space and reconstructs the expected distribution. MSCRED [25] combines convolutional layers, LSTM layers, and attention mechanisms to construct the Encoder, thus increasing the network's fitting capability.

Although RNNs have shown promising results in modeling the temporal dependence of time series data in the temporal dimension, they cannot directly capture the correlations between sensors.

3.2 Graph-based anomaly detection methods

The effectiveness of graph attention networks in predicting temporal dependencies and modeling correlations between sensors has been demonstrated in recent studies. For instance, MTAD-TF [6] employs multi-scale convolution and graph attention networks to capture the feature in temporal patterns. MTAD-GAT [28] uses two parallel graph attention layers to model correlations between indicators. Arvalus and its variant D-Arvalus [18] employs system deployment meta-information to construct a graph structure and introduces a new graph convolution (GC) technique to model correlations between indicators.

Although graph neural networks-based anomaly detection methods have shown promising results in modeling correlations between indicators, they still have some limitations. For instance, the Arvalus and its variant D-Arvalus [18] assume that there is a known graph structure predefined by domain knowledge. This assumption limits their generality and makes them sensitive to graph predefinition [27]. On the other hand, MTAD-TF [6] and MTAD-GAT [28] treat the relationship between indicators as a complete graph, which increases computational overhead.

3.3 Graph structure learning-based anomaly detection methods

To handle multivariate time series without a comprehensive real graph structure, GSL-based anomaly detection methods emerged. For instance, GDN [3] constructs a KNN graph by the similarity between the learned node embedding vectors. The learned graph structure is then fed into a graph attention neural network to extract dependencies between indicators and predict future behavior, where the prediction error is used to calculate the anomaly score. Similarly, FuSAGNet [5] partitions sensors based on their functions within a particular process and recursively encodes a group of sensors in the same process to construct a KNN graph. GTA [2] considers the elements of the adjacency matrix as learnable parameters and automatically learns the graph structure using a Transformer-based architecture to model temporal dependencies.

The graph structure obtained by pair-wise similarity, as with the GDN [3] and FuSAGNet [5], is undirected, which can not reflect the unidirectional dependencies between sensors. Although GTA [2] can obtain a uni-directional graph, the fully parametric learning method is complicated and has low training efficiency. Therefore, an efficient uni-directional graph structure learning method is necessary for MTSAD.

Compared with graph-based spatial and temporal learning, we use graph learning techniques rather than a predefined graph structure based on expert experience. Compared with previous temporal graph structural learning technique, we learn a uni-directional graph to model one-way relationships between sensors.

4 Problem definition and preliminaries

4.1 Problem definition

Multivariate time series data comprises a substantial amount of regularly spaced sampling and uninterrupted observation points, characterized by K indicators and N timestamps, which can be denoted by $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N)^T \in R^{K\times N}$. The i-th indicator can be represented by $\boldsymbol{x}^i = (x^i_1, x^i_2, \dots, x^i_N)$. The t-th timestamp contains K values of indicators, which is denoted by $\boldsymbol{x}_t = (x^1_t, x^2_t, \dots, x^K_t)^T$. Define the historical time series window of length ω at time t as subsequence $X_t = (\boldsymbol{x}_{t-\omega}, \boldsymbol{x}_{t-\omega+1}, \dots, \boldsymbol{x}_{t-1})^T \in R^{K\times\omega}$. When $\boldsymbol{x}_t$ is considered abnormal, the label $\boldsymbol{y}_t$ is set to 1.

Multivariate time series anomaly detection aims to identify whether the timestamp ($\boldsymbol{x}_t$) is anomalous. According to the historical time series X_t, it predicts the value of timestamp t, and the difference between the prediction and the ground truth is taken as an anomaly score to identify the anomaly. The process is formulated as follows:

$$\hat{x}_t = f(X_t) \tag{1}$$

$$s(t) = \varphi(x_t, \hat{x}_t) \tag{2}$$

$$\hat{y}_t = \begin{cases} 1, & \text{if } s(t) > T \\ 0, & \text{if } s(t) \le T \end{cases} \tag{3}$$

where $\hat{x}_t$ is the predicted value of timestamp t, $s(t)$ is the anomaly score of timestamp t, and T is the threshold. When the anomaly score s_t exceeds the threshold T, an anomaly is considered to have occurred at timestamp t.

4.2 Anti-symmetric matrix

A matrix A is said to be an *anti-symmetric matrix* if it satisfies the following conditions:

- A is a square matrix, meaning it has an equal number of rows and columns.
- For each element $A_{i,j}$ in matrix A, it holds that $A_{i,j} = -A_{j,i}$, i.e., $A^T = -A$.

The anti-symmetric matrix is exploited to implement uni-directional graph learning.

4.3 Graph structure learning

Given the time series $X \in R^{K\times N}$, the purpose of the graph structure learning is to obtain a graph $G = (V, E)$ and its graph topology or adjacency matrix $A \in R^{K\times K}$. The node v in the graph is a sensor that produces an indicator, and the hidden relationship between the sensors is considered the edge E. The adjacency matrix A stores the edge information in the graph, which reflects the underlying dependencies among indicators. The elements in the adjacency matrix are composed of 0 and 1. $A_{i,j}$ is 1, which represents an edge between node i and node j. On the contrary, there is no edge between node i and node j, when $A_{i,j} = 0$ [15, 21, 30].

If a graph is directed, its adjacency matrix satisfies that if $A_{i,j}$ equals 1, then $A_{j,i}$ must be 0.

5 Our proposed methodology

This section provides a comprehensive explanation of our proposed approach. The notation used in this section is described in Table 1.

Table 1 List of notations

Notation	Meaning
X	Multivariate time series
X_t	historical time series window of length ω at time t
$\tilde{X}_t$	The masked historical time series
$\boldsymbol{x}^i$	The i-th indicator values
$\boldsymbol{x}_t$	The indicator values at timestamp t
$T_f()$	The masking transformation
$\hat{\boldsymbol{x}}$	The prediction
K	The number of indicators
N	The number of timestamps
y_t	The label of timestamp t
G	Graph
A	Adjacency matrix
θ	Prior adjacency matrix
$V_i^{(t)}$	The feature vector of sensor i at timestamp t
E_1	The source embedding vector
E_2	The destination embedding vector
$Err_i(t)$	The prediction error at timestamp t for sensor i
$s(t)$	Anomlay scores

5.1 Overview

Complex topological relationships often exist among monitored indicators in real-world scenarios, which can be represented as a graph. In this graph, each indicator is regarded as a node, while the relationships between them are represented by edges connecting the nodes. Most previous methods [3, 5] learn an undirected graph that cannot represent one-way dependencies between sensors. Although GTA [2] uses a direct method to obtain a uni-directional graph, the GTA method is inefficient. Therefore, we propose a uni-directional Graph Structure Learning-based Multivariate Time Series Anomaly Detection with Dynamic Prior Knowledge. Figure 4 illustrates the framework for our method. Essentially, our method comprises four key components:

- **Uni-directional graph structure learner:** To mine one-way dependencies between sensors at a low cost, we construct a uni-directional graph using the antisymmetric matrix with the ReLu function.
- **Dynamic prior graph generator:** To improve the quality of the learning graph, we utilize a dynamic prior graph to provide dynamic prior information for learning graph.
- **DCRNN predictor based on timestamp mask:** To effectively handle data with missing values, we utilize the timestamp masking mechanism to eases the impact of missing values in the raw data. The time series is masked and then fed into DCRNN with the uni-directional graph to predict future values for each sensor.
- **Anomaly score calculation:** After the predictor is trained, the prediction error is used to calculate the anomaly score.

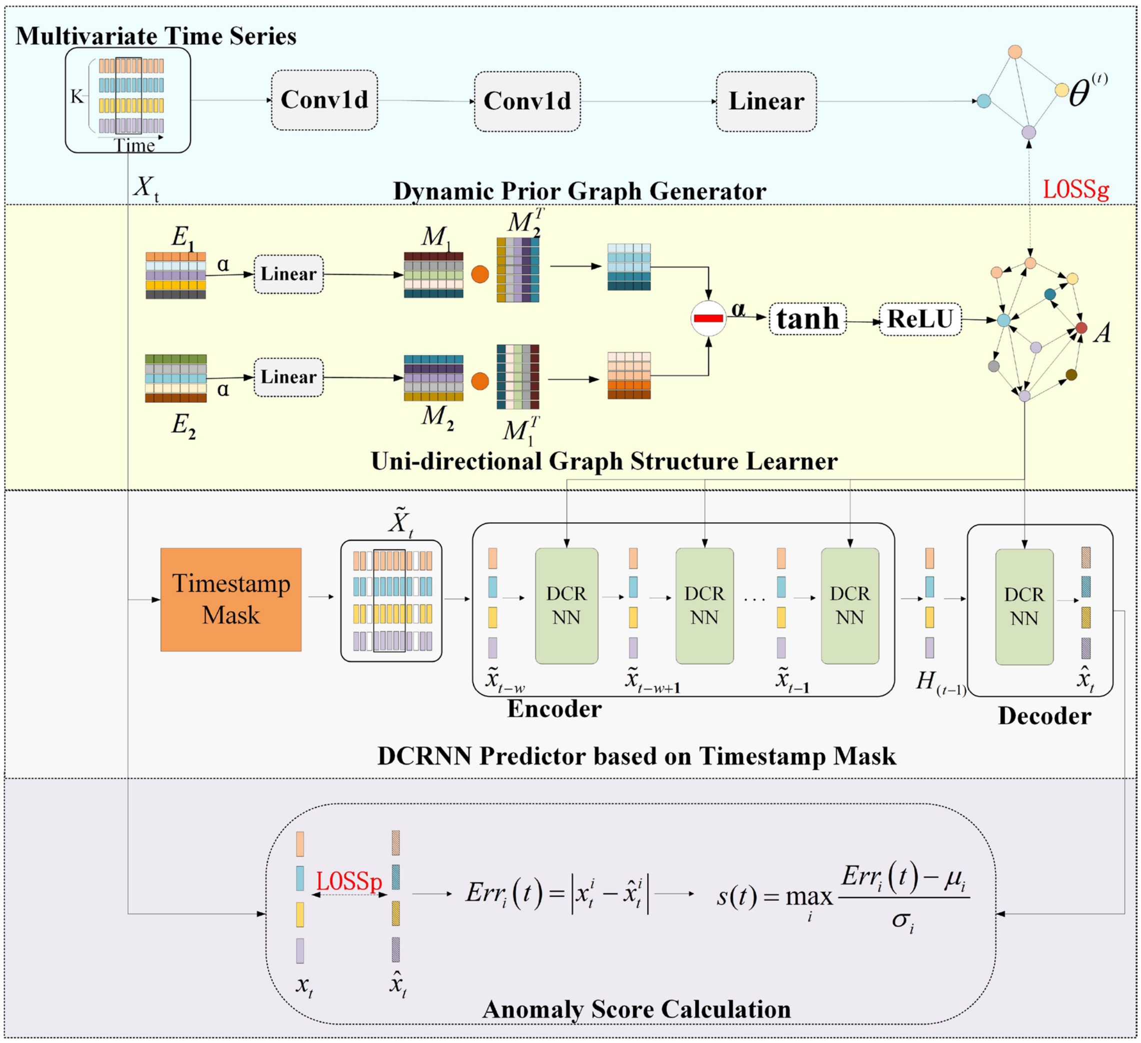

Fig. 4 The DPGLAD framework comprises four main modules. The first module, depicted in gold, models the inter-indicator relationships. The second module, depicted in cyan, is responsible for generating a dynamic prior graph that provides prior information to the graph structure learner. The third module, shown in grey, utilizes a timestamp mask-based DCRNN predictor to generate accurate predictions of indicators. The last module, shown in lilac, perform anomaly score calculation. Loss functions are highlighted in red

5.2 Uni-directional graph structure learner

Many of the current graph structure learning-based anomaly detection methods [3, 5] depend on node similarity metrics to construct the graph structure. Consequently, the graph structure is undirected and the relationship between nodes is symmetrical. However, there are uni-directional relationships between nodes.

Our uni-directional graph structure learner is specially tailored to identify and extract one-way dependencies. It is implemented by a source node embedding vector and a target node embedding vector as follows:

$$M_1 = \tanh\left(\alpha E_1 \beta_1\right) \tag{4}$$

$$M_2 = \tanh\left(\alpha E_2 \beta_2\right) \tag{5}$$

$$A = \mathrm{ReLU}\left(\tanh\left(\alpha\left(M_1 M_2^T - M_2 M_1^T\right)\right)\right) \tag{6}$$

where E_1 and E_2 represent the source and target node embedding vectors, respectively, β_1, β_2 are the model parameters, and the activation function's saturation rate is symbolized by α. E_1 and E_2 are initialized as random, which are updated by the backpropagation of graph learning loss. Subtraction operation $(M_1 M_2^T - M_2 M_1^T)$ in Eq. (6) can construct an anti-symmetric matrix according to Eq. (7). In an anti-symmetric matrix A, the value of $A_{j,i}$ equals $-A_{i,j}$.

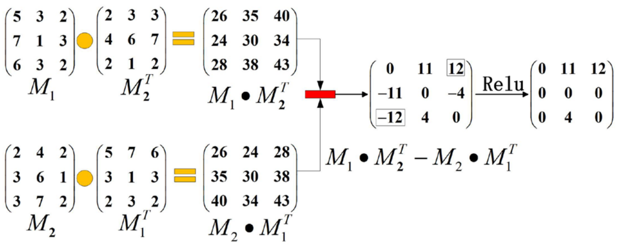

Fig. 5 The process of constructing a uni-directional matrix

$$\begin{aligned}\left(M_1M_2^T - M_2M_1^T\right)^T &= \left(M_1M_2^T\right)^T - \left(M_2M_1^T\right)^T \\ &= \left(M_2M_1^T\right) - \left(M_1M_2^T\right) \\ &= -\left(M_1M_2^T - M_2M_1^T\right)\end{aligned} \tag{7}$$

Subsequently, the uni-directional adjacency matrix is obtained by setting the negative value $A_{j,i}$ to 0 by the ReLU activation function. Figure 5 gives a simple example to illuminate the construction of a uni-directional matrix. To construct a one-way graph, it is necessary to calculate the similarity weights for every pair of node embedding vectors, which incurs a computational cost of $O(K^2)$.

To reduce the computation cost, only the first k large values are considered as neighbors, and the rest are set to 0 as follows:

$$\begin{aligned}&\text{for } i = 1, 2, \dots, K: \\ &\quad idx = \arg\operatorname{topk}\left(A_{i,:}\right) \\ &\quad A_{i,j} = 1, \quad j \in idx\end{aligned} \tag{8}$$

where the index of the top-k largest values of a vector is returned by arg topk(·).

5.3 Dynamic prior graph generator

To enhance the quality of the learning graph, the existing graph structure learning method provides prior knowledge in the format of a prior graph for graph learning. Currently prior graph generation methods typically utilize all raw data as input and transform them into a KNN graph as prior knowledge. This prior graph is a static graph and can only describe fixed relationships between indicators.

However, the relationships between indicators usually change over time. As shown in Fig. 6, the purple line and the blue line exhibit synchronous fluctuations from t_1 to t_2, but they diverge and move in opposite directions after t_3. Therefore, it is necessary to capture the various and dynamic relationships between sensors. Specifically, the dynamic prior graph generator comprises two essential components: the feature extractor and the KNN graph generator.

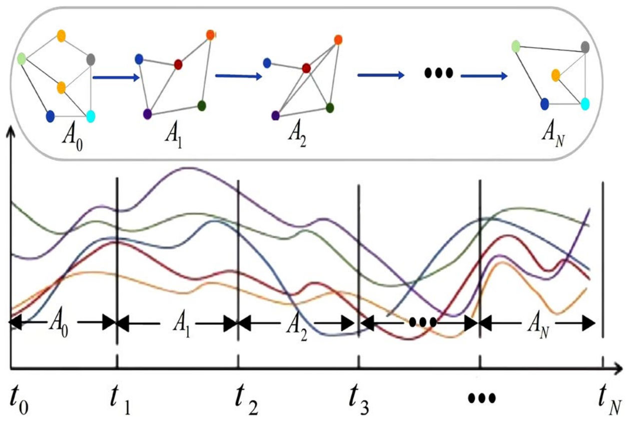

Fig. 6 The relationship between sensors changes over time

5.3.1 Feature extractor

To represent each sensor, we design a feature extractor that generates a feature vector for each sensor. The feature vector characterizes the diverse behaviors of different sensors. To capture the relationships changing over time, the input of the feature extractor at timestamp t is the subsequence X_t instead of the whole time series. The feature vector of the sensor i at timestamp t is obtained by applying two one-dimensional convolutional layers and a fully connected layer [19] as follows:

$$V_i^{(t)} = \mathrm{FC}\left(\mathrm{Conv}\left(\mathrm{Conv}\left(x_i^{(t)}\right)\right)\right) \tag{9}$$

where $x_i^{(t)} = (x_{t-\omega}^i, x_{t-\omega+1}^i, \dots, x_{t-1}^i) \in R^w$, and $V_i^{(t)}$ is the feature vector of sensor i at timestamp t. Conv stands for the one-dimensional convolutional layer and FC stands for the fully connected layer.

5.3.2 KNN graph generator

The KNN graph generator uses the features vector from the feature extractor to generate prior graphs. The initial step of the KNN graph generator is to compute the cosine similarity of two feature vectors as follows:

$$\cos\left(V_i^{(t)}, V_j^{(t)}\right) = \frac{V_i^{(t)} \bullet V_j^{(t)}}{\left\|V_i^{(t)}\right\| \bullet \left\|V_j^{(t)}\right\|} \tag{10}$$

where $\|\cdot\|$ denotes magnitude. We choose the most similar k nodes for each node as follows:

$$\theta_{i,j}^{(t)} = 1, j \in \operatorname{topk}\left(\cos\left(V_i^{(t)}, V_j^{(t)}\right)\right) \tag{11}$$

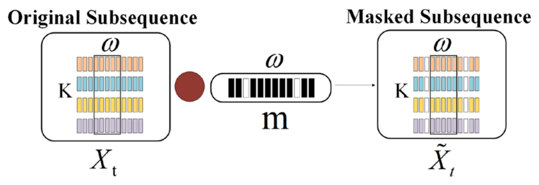

Fig. 7 A simple example of timestamp mask

where $\theta^{(t)}_{i,j}$ denote the element located in the i-th row and j-th column of the prior graph and topk(·) selects the k node indices with the highest cosine similarities [3].

In this way, we can generate a list of prior graphs $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(\frac{N}{\text{stride}}-\omega+1)}$, where N is the total length of the time series, the *stride* is the step size of the sliding window, and ω is the size of the sliding window.

Although the obtained prior graph is undirected, it still provides efficient knowledge for uni-directional graph structure learning. In detail, the undirected edges in the prior graph represent the relationships between connected nodes, and the graph learner strives to determine the direction of these edges within the constraint of the prior graph. The prior graph $\theta^{(t)}$ serves as the direction candidates for graph learning.

5.4 DCRNN predictor based on timestamp mask

To handle the missing value in prediction, we propose the timestamp mask-based DCRNN predictor. It combines a timestamp mask mechanism and a recurrent graph neural network to robustly make predictions of future values by actively masking some values. The masked subsequence $\tilde{X}_t$ and the learned adjacency matrix A from the graph structure learner are utilized as input for the recurrent graph neural network to predict the following value of the subsequence.

5.4.1 Timestamp masking

For a given subsequence $X_t \in R^{K\times\omega}$, a masking vector $m \in \{0,1\}^\omega$ is first sampled, where each element is drawn from a Bernoulli distribution with probability p independently [24]. Then, we mask the subsequence X_t with m, resulting in the creation of a masked subsequence denoted by $\tilde{X}_t$.

$$\tilde{X}_t = T_f(X_t, m) = \left[\boldsymbol{x}^1 \odot m, \ldots, \boldsymbol{x}^K \odot m\right]^T \tag{12}$$

where $T_f()$ is the masking transformation, and $\boldsymbol{x}^i \in R^\omega$ is the transpose of the i-th row vector of X_t. Fig. 7 gives a simple example of masking process. The input of the DCRNN predictor consists of $\tilde{X}_t$ and the adjacency matrix A.

5.4.2 DCRNN predictor

DCRNN [13] is designed for uni-directional graphs and can capture the temporal and spatial features simultaneously. DCRNN captures spatial features by diffusion convolution and temporal features by Gated Recurrent Unit (GRU).

The diffusion convolution can aggregate L hops neighbors features, which is defined as follows:

$$W_Q \circ Y = \sum_{l=0}^{L} \left(w^Q_{l,1} \left(D_O^{-1}A\right)^l + w^Q_{l,2} \left(D_I^{-1}A^T\right)^l \right) Y \tag{13}$$

where $\circ$ represents diffusion convolution operation, D_O and D_I are the out- and in-degree matrices, $w^Q_{l,1}$ and $w^Q_{l,2}$ denote the model parameters, and L is the diffusion degree.

GRU is designed to capture the temporal features. To compensate for the capability of capturing spatial features, DCRNN uses the diffusion convolution operation to replace the linear multiplication in GRU, which is formulated as follows:

$$R_t = \text{sigmoid}\left(W_R \circ \left[\tilde{x}_t || H_{(t-1)}\right] + b_R \right) \tag{14}$$

$$C_t = \tanh\left(W_C \circ \left[\tilde{x}_t || (R_t \odot H_{(t-1)})\right] + b_C \right) \tag{15}$$

$$U_t = \text{sigmoid}\left(W_U \circ \left[\tilde{x}_t || H_{(t-1)}\right] + b_U \right) \tag{16}$$

$$H_{(t)} = U_t \odot H_{(t-1)} + \left(1 - U_t\right) \odot C_t \tag{17}$$

where $\tilde{x}_t$ and $H_{(t)}$ are the input and output at timestamp t, $\circ$ is the diffusion convolution operation, || is the concatenation operation of two features, R_t, C_t, and U_t are the update gate, reset gate and the candidate hidden state, respectively.

For the masked subsequence $\tilde{X}_t$, DCRNN utilizes an encoder and decoder architecture to predict the value of the next timestamp. Its process can be summarized as follows: In the encoder, the hidden feature $H_{(\cdot)}$ is updated from timestamp $t - w$ to timestamp $t - 1$, where w denotes the length of the subsequence. This updating process accumulates the information from multiple historical timestamps, resulting in the total hidden feature $H_{(t-1)}$ of the subsequence, as illustrated in Fig. 4. In the decoder, the total hidden feature $H_{(t-1)}$ is decoded by a DCRNN layer to predict the value $\hat{x}_t$ at timestamp t.

5.5 Loss function

Generally, the mean absolute error is employed as the loss function $loss_p$ for the prediction task.

$$loss_p = \frac{1}{K}\sum_{i=1}^{K}\left|\hat{x}_t^i - x_t^i\right| \tag{18}$$

where $\hat{x}_t^i$ and x_t^i represent the predicted value and ground truth of the i-th indicator at timestamp t, respectively.

To improve the quality of the learning graph, we introduce a graph learning loss during model training to impose constraints on the learning graph. The graph learning loss $loss_g$ is expressed as the cross-entropy between the prior knowledge $\theta^{(t)}$ and the learned graph structure A as follows:

$$loss_g = \sum_{ij} -\theta_{i,j}^{(t)} \log A_{i,j} - \left(1 - \theta_{i,j}^{(t)}\right) \log\left(1 - A_{i,j}\right) \tag{19}$$

To mitigate overfitting, an L_2 regularization term is incorporated into the loss function. The total loss function of the model is defined as follows:

$$loss = loss_p + \lambda_1 loss_g + \lambda_2 \|\boldsymbol{w}\|_2^2 \tag{20}$$

where the parameter λ_1 and λ_2 is the regularization magnitude. The prediction loss $loss_p$ makes the prediction as close as possible to the ground truth. Meanwhile, the graph learning loss $loss_g$ controls the learning direction of the graph learning to capture meaningful spacial dependencies. Regularization term prevents from overfitting and enhances the model's generalization to unseen data.

5.6 Anomaly score calculation

We identify anomalies that deviate from normal behavior based on the ground truth and prediction values. As a result, the first step is to calculate the individual anomaly score for each sensor. These scores are later combined to obtain the aggregative anomaly score for each timestamp. Whenever the aggregative anomaly score surpasses a predetermined threshold, it is regarded as an anomaly.

We compare the ground truth and the predicted value at timestamp t to calculate the prediction error $Err_i(t)$ for sensor i as follows:

$$Err_i(t) = \left|x_t^i - \hat{x}_t^i\right| \tag{21}$$

To ensure the consistency of metric scales among sensors with varying value ranges, a standard normalization is conducted on the prediction error as follows:

$$s_i(t) = \frac{Err_i(t) - \mu_i}{\sigma_i} \tag{22}$$

where μ_i and σ_i are the mean and standard deviation of $\mathrm{Err}_\mathrm{i}(t)$, respectively.

Then, the aggregative anomaly score at timestamp t is determined by the highest anomaly score among all sensors as follows:

$$s(t) = \max_i s_i(t) \tag{23}$$

5.7 Threshold selection

To determine the optimal threshold, a grid search technique is employed. The upper and lower bounds of the threshold are defined as the maximum and minimum values of $s(t)$, respectively. An exhaustive search of all possible thresholds is conducted with a step size of 0.01. The threshold with the highest F1 score is selected as the optimal threshold. In addition, we use a point-adjust strategy for anomaly scores according to Ref. [20].

6 Experiments and performance analysis

In this section, we explain our experiments in detail and answer the following research questions:

- **RQ1 (Detection Performance, Efficiency and Computation Complexity):** Does our method outperform the baseline method in terms of both performance and efficiency in anomaly detection? What is the computer complexity of the model?
- **RQ2 (Parameter Influence):** How sensitive is DPGLAD with different parameters?
- **RQ3 (Graph Structure Learner Performance):** Does our proposed UGL outperform other graph learning methods in anomaly detection models in terms of performance and efficiency?
- **RQ4 (Ablation Studies):** How does each component of DPGLAD affect its performance, and is DPGLAD more effectively to handle input data with missing values?

6.1 Datasets

In our experiments, we utilize four public and real-world datasets. The statistics of all these datasets are presented in Table 2.

The Safe Water Treatment (SWAT) dataset[2] originates from a water treatment testbed that is overseen by the Public Utilities Authority of Singapore. The data collection process spanned 11 days and is continuously operating for 24 h a day, during which network traffic and values from all 51 sensors and actuators are recorded.

The Water Distribution (WADI) dataset[3] is a comprehensive water distribution system consisting of a

[2] https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_SWAT/.

[3] https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_WADI/.

Table 2 Description of datasets

Dataset	SWAT	WADI	MSL	SMAP
Indicators	51	127	55	25
# of training	495000	762970	58317	135183
# of testing	450000	172800	73729	427617
Anomalies	11.97%	5.99%	10.72%	13.13%

multitude of pipelines. As an extension of the SWAT testbed, WADI presents a more thorough and lifelike representation of water treatment, storage, and distribution networks. The dataset encompasses a continuous period of 16 days, during which 14 days encompass regular operations, and the remaining 2 days cover attack scenarios. The testbed is equipped with 127 sensors and actuators.

The Mars Science Laboratory rover (MSL) is a dataset of sensor and actuator data from the Mars rover by NASA. This dataset comprises 55 distinct metrics for 27 unique entities.

The Soil Moisture Active Passive satellite (SMAP) is a dataset of soil samples and telemetry collected by NASA using the Mars rover. This dataset comprises 25 metrics for 55 entities.

To accommodate the extensive volume of raw data, a down-sampling process is implemented every 10 s for both the SWAT and WADI datasets. The median value is captured during this interval. Once an anomaly occurs within a 10-s window, it is marked as abnormal.

6.2 Evaluation metrics

It is common to adopt F1-Score (F1), precision (Prec), and recall (Rec) as evaluation metrics of anomaly detection performance as shown in Eqs. (24), (25) and (26):

$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{24}$$

$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{25}$$

$$F1 = \frac{2 \times \text{Prec} \times \text{Rec}}{\text{Prec} + \text{Rec}}. \tag{26}$$

where TP, TN, FP, and FN are the numbers of true positives, true negatives, false positives, and false negatives.

Besides, the efficiency is evaluated by assessing the training time per epoch for model training.

6.3 Baselines

We compare DPGLAD with twelve machine learning and deep learning methods, which are AE, IF, DAGMM, LSTM-NDT, LSTM-VAE, MAD-GAN, OmniAnomaly, USAD, MTAD-GAT, GDN, FuSAGNet, and GTA.

- AE: Autoencoder is utilized to reconstruct the input data, and the reconstruction error is used as the anomaly score.
- IF [14]: The isolation forest method is a tree-based anomaly detection algorithm. It effectively identifies anomalous samples by gaining insight into the distribution of the input data.
- DAGMM [29]: It simultaneously trains a deep autoencoding and Gaussian mixture model, with the objective of generating a low-dimensional representation and identifying anomalies based on reconstruction errors.
- LSTM-NDT [10]: It uses LSTM to achieve high prediction performance and provides a nonparametric, dynamic, and unsupervised anomaly thresholding method to detect anomalies.
- LSTM-VAE [17]: It projects multimodal observation and temporal dependencies into a latent space and reconstructs the expected distribution through LSTM-based VAE.
- MAD-GAN [12]: It exploits LSTM as the base model in the GAN framework to capture the temporal correlation of time series distributions.
- OmniAnomaly [20]: It is a prior-driven stochastic model for timestamp anomaly detection that directly returns the reconstruction probability.
- USAD [1]: It adversarially trains an encoder-decoder framework to achieve rapid and efficient training.
- MTAD-GAT [28]: It treats the relationship between indicators as a complete graph and utilizes graph attention neural networks for anomaly detection.
- GDN [3]: It uses pair-wise cosine similarity between nodes to construct graph structures and utilizes attentional GNNs to learn the dependencies between time series and predict behavior.
- FuSAGNet [5]: It learns the graph structure through pair-wise cosine similarity between recursive sensor embeddings and obtains a sparse representation of the input data through a sparse autoencoder, which is fed into a graph attention network to predict future sensor behavior.
- GTA [2]: It involves automatically learning a graph structure and utilizes Transformer-based architecture to model temporal dependency.

Table 3 Precision, recall and F1 score on SWAT and WADI

Method	SWAT			WADI			SMAP			MSL		
	Pre	Re	F1	Pre	Re	F1	Pre	Re	F1	Pre	Re	F1
AE	72.6	52.6	61.1	34.3	34.3	34.3	72.1	97.9	77.7	85.3	97.4	87.9
IF	96.2	73.2	83.1	62.4	61.6	62.0	44.2	51.1	46.7	56.8	67.4	59.8
DAGMM	27.5	69.5	39.4	54.4	27.0	36.1	63.3	99.8	71.2	75.6	98.0	81.1
LSTM-NDT	77.8	51.1	61.7	1.4	78.2	2.7	85.2	73.3	78.8	62.9	*100.0*	77.2
LSTM-VAE	96.2	59.9	73.9	*87.8*	14.5	24.8	71.6	98.8	75.6	86.0	97.6	85.4
MAD-GAN	99.0	63.7	77.5	41.4	33.9	37.3	81.6	92.2	86.5	85.2	99.3	91.7
OmniAnomaly	72.2	**98.3**	83.3	26.5	**98.0**	41.7	75.9	97.6	85.4	91.4	88.9	90.1
USAD	**100.0**	56.0	71.8	43.1	22.5	29.6	74.8	96.3	84.2	79.5	99.1	88.2
MTAD-GAT	21.0	64.5	31.7	11.7	30.6	16.9	79.9	*99.9*	88.8	79.2	98.2	87.7
GDN	*99.4*	68.1	80.8	**97.5**	40.2	56.9	74.8	98.9	85.2	*93.1*	98.9	*95.9*
FuSAGNet	98.7	72.6	83.7	82.9	47.8	60.7	9.3	67.9	16.5	80.6	98.9	88.9
GTA	93.9	85.7	*89.6*	79.6	79.4	*79.5*	*89.1*	91.8	*90.4*	91.0	91.2	91.1
DPGLAD	95.1	*93.0*	**94.0**	80.7	*87.8*	**83.9**	**94.2**	**100.0**	**97.0**	**99.8**	**100.0**	**99.9**

The highest and second-highest results are highlighted with boldface and italics, respectively

Table 4 Experimental parameter setting

Parameter	sliding window size ω	λ_1	λ_2	learning rate	*batchsize*	*epochs*
MSL	12	1	0.0001	0.005	64	30
SMAP	12	1	0.0001	0.005	64	30
SWAT	15	1	0.0005	0.005	64	30
WADI	15	1	0.0005	0.005	64	30

6.4 Settings

The parameters for our experiments are set as shown in Table 4. All experiments were conducted using Python 3.8, PyTorch 1.10, and CUDA version 11.3, and were trained on a server equipped with an Intel(R) Xeon(R) Platinum 8255C CPU and NVIDIA RTX 3080 GPU.

6.5 RQ1. Detection performance, efficiency and computation complexity

Firstly, We evaluate the performance of DPGLAD by comparing it with all other baseline methods. Secondly, we compare its training time with existing GSL-based anomaly detection methods [2, 3, 5] Finally, we calculate the number of parameters of the model and plot the ROC curve.

6.5.1 Detection performance

As shown in Table 3, the DPGLAD significantly outperforms other baseline methods. DPGLAD uses spatial and temporal graph neural networks relative to traditional CNN and LSTM (IF, DAGMM, LSTM-NDT, LSTM-VAE, MAD-GAN, OmniAnomaly, USAD) to effectively extract the relationship between indicators. In comparison to the latest GSL-based anomaly detection methods, such as GDN [3], FuSAGNet [5] and GTA [2], DPGLAD efficiently extracts one-way relationships between nodes and enhances learning graph quality by dynamic prior graphs. The ROC-AUC curve of DPGLAD is shown in Fig. 8. All AUC of four dataset are over 0.9.

The performance of all methods on the WADI dataset is comparatively lower than the other datasets. This can be attributed to the WADI dataset's longer length, larger number of indicators, and lower anomaly rates compared to the other datasets, as shown in Table 2. However, DPGLAD outperforms the baseline method on the WADI dataset due to its utilization of dynamic prior graphs as prior knowledge for graph structure learning. Thus, DPGLAD is effective in high-dimensional time series and sample imbalance scenarios, making it suitable for practical applications.

The anomaly detection performance of FuSAGNet on the SMAP and MSL datasets is insufficient. The reason is that FuSAGNet is primarily designed for Cyber-Physical Systems (CPSs), where CPS sensors can be categorized into various specific processes [5]. FuSAGNet, as a result, incorporates sensors in each process individually. In contrast, the sensors present in the SMAP and MSL datasets lack specific processes. Besides, LSTM-NDT [10] is also insufficient on the WADI datasets. The reason is that LSTM-NDT uses a

Table 5 Running time of each epoch(s)

Method	SWAT	WADI	SMAP	MSL
GDN	11.13	42.4	0.67	0.57
FuSAGNet	44.23	75.29	1.57	1.14
GTA	107.38	154.33	8.46	4.17
Our	26.37	42.5	1.67	0.83

Table 6 Computation time and the number of parameters of DPGLAD

Dataset	Computation Time(s)	Parameters
SWAT	791.1	1586090
WADI	1275	1627814
SMAP	50.1	1269917
MSL	24.9	1270511

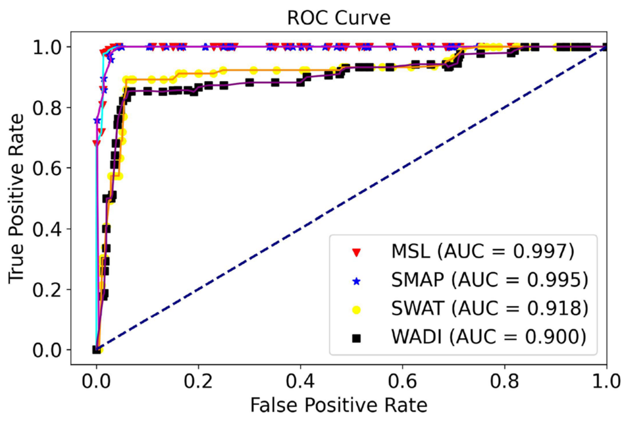

Fig. 8 The ROC curve of the model

dynamic error threshold method to compute the threshold, which is difficult to obtain an appropriate threshold when the dataset has a low anomaly rate.

6.5.2 Training time and computation complexity

Our method, when compared to FuSAGNet [5] and GTA [2], boasts a shorter training time across all datasets while remaining in close proximity to GDN as shown in Table 5. GDN's shorter training time can be attributed to its simplistic architecture, only using a graph attention network. Conversely, GTA uses the Transformer for prediction, leading to a significantly increased complexity. FuSAGNet joint trains a sparse autoencoder and a graph attention network for reconstruction and prediction. Its performance is superior to GDN [3] and inferior to GTA [2], while its training time is inferior to GDN [3] and superior to GTA [2]. DPGLAD uses timestamp mask-based DCRNN for prediction, which greatly reduces the training time while ensuring the model's performance. To evaluate the computation complexity of DPGLAD, we also present the computation time and the number of parameters of DPGLAD, as shown in Table 6. The WADI dataset with 127 indicators has the largest nodes among all the datasets in the anomaly detection field. The training time of DPGLAD is still less than that of GTA. Therefore, the scalability of DPGLAD on large datasets is also commendable.

6.6 RQ2. Parameter influence

To demonstrate the stability of our method to different parameters, we analyze the impact of hyperparameters and regularization parameters.

6.6.1 Window size

In this experiment, we examine the impact of window sizes. The window size is set from 5 to 60. The results are presented in Fig. 9. Notably, our method demonstrates a stable performance with different window sizes across several datasets, including SWAT, SMAP, and MSL. However, the window size has an impact on the detection performance of the WADI dataset. The reason is that the WADI dataset has a large sampling number and a low anomaly rate. In particular, DPGLAD outperforms both GDN and FuSAGNet with their default window size ω of 5. Moreover, the DPGLAD with a window size of 15 outperforms GTA with its default window size of 60. Thus, compared with GDN, FuSAGNet, and GTA, our method exhibits superior performance, higher F1 scores, and lower complexity, while requiring only short-term history data.

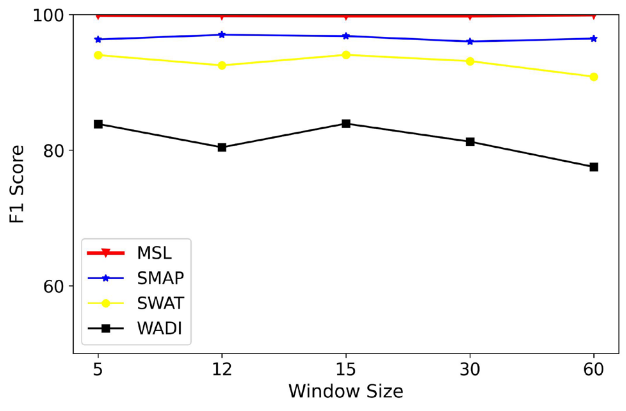

Fig. 9 The impact of window size

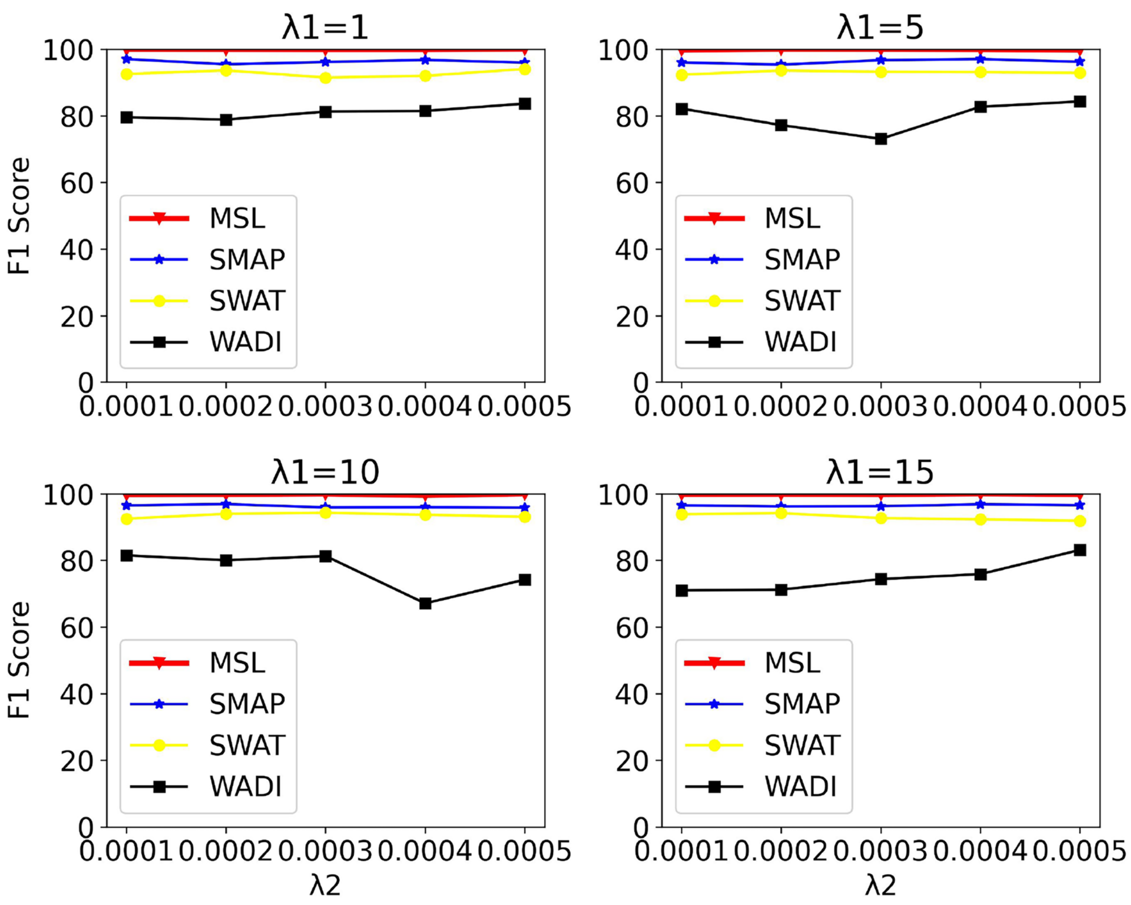

Fig. 10 The effect of regularization parameters

6.6.2 Regularization parameters

In this experiment, we present the impact of the regularization parameters λ_1 and λ_2 on four datasets. We vary λ_1 from 1 to 15, and λ_2 from 0.0001 to 0.0005. As shown in Fig. 10, the optimal performance is achieved and the performance appears to be generally stable, when λ_1 is set to 1. λ_2 has little effect on the performance.

6.7 RQ3. Graph structure learner performance

To demonstrate the effectiveness of the uni-directional graph structure learner of DPGLAD, we compare UGL with two existing graph structure learner methods: The k-neighbour Method (KNM) and the Fully Parameterized Method (FPM). Besides, we modify the UGL with only one node embedding to generate an undirected graph for comparison, which is termed as Undirected Graph Method (UGM). Among the three comprised methods, KNM and UGM learn an undirected graph.

6.7.1 k-neighbour method

This method produces an embedding vector for each sensor and subsequently evaluates the cosine similarity between sensors. Finally, the top k most similar sensors are selected to build the adjacency matrix. It is referred to as KNM and exploited by GDN [3] and FuSAGNet [5].

$$\cos\left(E_i, E_j\right) = \frac{E_i \bullet E_j}{||E_i|| \bullet ||E_j||} \tag{27}$$

$$A_{i,j} = 1, j \in \mathrm{topk}(\cos(E_i, E_j)) \tag{28}$$

where $\|\cdot\|$ denotes magnitude, $A_{i,j}$ denote the element located in the i-th row and j-th column of the adjacency matrix and $\mathrm{topk}(\cdot)$ selects the k node indices with the highest cosine similarities [3].

6.7.2 Undirected graph method

To demonstrate the validity of the uni-directional graph structure, we use one node embedding instead of two node embeddings to construct an undirected graph using pair-wise similarity. It is referred to as UGM.

$$M_1 = \tanh(\alpha E_1 \beta_1) \tag{29}$$

$$A = \text{ReLU}(\tanh(\alpha(M_1 M_1^T)) \tag{30}$$

$$\begin{aligned} &\text{for } i = 1, 2, \ldots, K : \\ &\quad idx = \arg \text{topk}(A_{i,:}) \\ &\quad A_{i,j} = 1, \quad j \in idx \end{aligned} \tag{31}$$

where E_1 represents the node embedding vectors, β_1 are the model parameters, and the activation function's saturation rate is symbolized by α. The index of the top k largest values of a vector is returned by arg topk$(\cdot)$. $M_1 M_1^T$ is a symmetric matrix, which generates a symmetric adjacency matrix after tanh and ReLU functions.

6.7.3 Fully parameterized method

This method first randomly initializes the probability matrix $\pi_1 \in R^{K \times K}$. The adjacency matrix is Gumbel-Softmax sampled from the π_1. It is referred to as FPM and exploited by GTA [2].

$$g_i = -\log(-\log(u)), u \sim \text{Uniform}(0, 1) \tag{32}$$

$$z_1^{i,j} = \frac{\exp\left(\left(\log \pi_1^{i,j} + g_1^{i,j}\right)/\tau\right)}{\sum_{v \in \{0,1\}} \exp\left(\left(\log \pi_v^{i,j} + g_v^{i,j}\right)/\tau\right)} \tag{33}$$

where u is samples drawn from the Uniform(0,1) distribution, $g_v^{i,j}$ satisfy the gumbel distribution, and $\pi_1^{i,j}$ is the value of row i and column j of the probability matrix π_1, which represents the probability that node i is connected to node j. τ is the temperature parameter. As the temperature τ approaches 0, $z_1^{i,j}$ is close to 0 or 1 and the Gumbel-Softmax distribution becomes identical to the class distribution. Finally, $A_{i,j}$ in adjacency matrix is set to $z_1^{i,j}$, which is set to 1 with probability $\pi_1^{i,j}$.

Table 8 Running time of each epoch(s) with different graph structure learner

Method	SWAT	WADI	SMAP	MSL
KNM	23.03	36.02	1.4	0.67
UGM	25.02	34.3	1.53	0.73
FPM	29.06	62.42	1.30	0.83
UGL	26.37	42.5	1.67	0.83

Table 7 and Table 8 compare the anomaly detection performance and training time of three methods with our proposed UGL. Although the K-neighbour and UGM exhibit superior training time compared to the UGL, their anomaly detection performance is compromised due to their inability to capture the uni-directional relationships among sensors. Although the FPM produces a uni-directional graph, it has a longer training time than the UGL method and lower detection performance.

We conduct a case study, as depicted in Fig. 11. On the left side of Fig. 11, we can observe a partial graph structure that is learned by DPGLAD. On the right side, we present predictions for the relevant sensors. In this case, sensor AIT-202 is compromised, resulting in changes in its value between timestamps 708 and 730. Due to the uni-directional correlation between sensors in the water treatment process, the attack on AIT-202 causes the dosing pump P-203 to shut down, which in turn affects the permeate conductivity analyzer AIT-503 at timestamp 800. As depicted in the right side of Fig. 11, DPGLAD accurately predicts the behavior of AIT-503 and successfully detects the attack by identifying the significant difference between the predictions and the ground truth of AIT-503 at timestamp 800. On the left side of Fig. 11, the correlation among the three sensors involved in this case is correctly represented.

Table 7 Precision, recall, and F1 score with different graph structure learners

Method	SWAT			WADI			SMAP			MSL		
	Pre	Re	F1	Pre	Re	F1	Pre	Re	F1	Pre	Re	F1
KNM	96.9	88.9	92.7	78.9	78.1	78.5	92.2	100	95.9	99.1	100	99.5
UGM	95.3	90.4	92.8	82.9	75.2	78.9	92.6	100	96.2	99.3	100	99.6
FPM	97.4	90.5	*93.8*	80.2	78.1	*79.1*	92.8	100	*96.2*	99.2	100	*99.6*
UGL	95.1	93.0	**94.0**	80.7	87.8	**83.9**	94.2	100	**97.0**	99.8	100	**99.9**

The highest and second-highest results are highlighted with boldface and italics, respectively

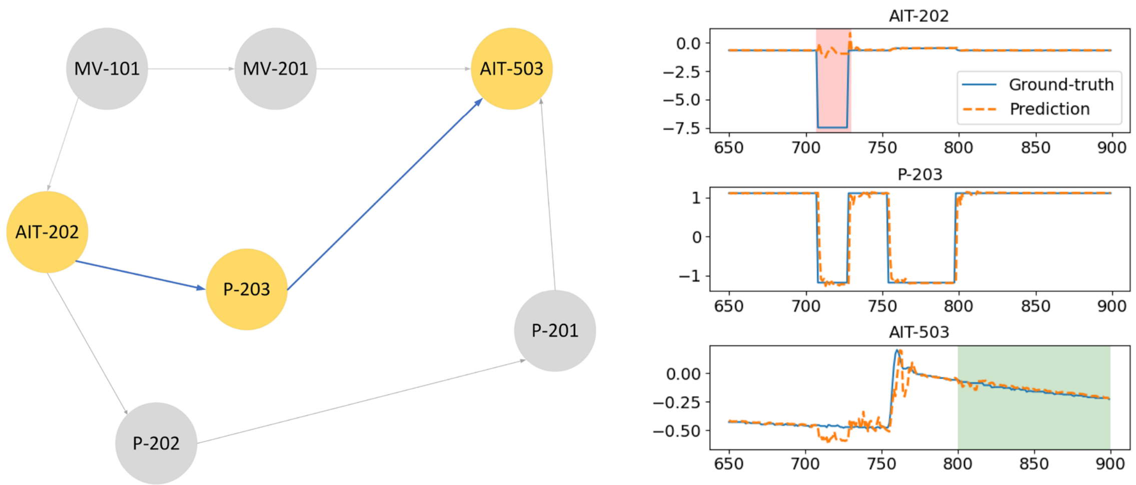

Fig. 11 **Left**: Partial graph structure learned on SWAT dataset. **Right**: Understand the relationship between sensors with ground truth and prediction

6.8 RQ4. Ablation studies

In this experiment, ablation studies are conducted to showcase the necessity of DPGLAD components in achieving optimal detection performance. We focus on the dynamic prior graph generator and timestamp masking.

For the dynamic prior graph, we consider the following two methods.

- NPGLAD stands for DPGLAD in the absence of the priory knowledge and the graph learning loss function.
- SPGLAD utilizes a static prior graph extracted from all row data to replace the dynamic prior graphs in the graph learning loss function.

As shown in Table 9, the dynamic prior graph is proven to be more flexible and provides better prior information for the learning graph compared to no prior graph and the static prior graph, resulting in a better detection effect.

We evaluate the effectiveness of timestamp masking (*TM*) on the raw datasets and the datasets with missing values. We consider the method without *TM*.

- DPGLAD without *TM* eliminates the timestamp masking component, in which the raw subsequence X_t is fed into the DCRNN.

To demonstrate the performance with missing value, we randomly delete the data from the raw data as missing dataset. The percentage of missing values ranges from 0.05 to 0.2. As shown in Fig. 12, as the increase of percentage of missing values, the F1 scores of two methods decreases. However, DPGLAD always outperforms that without *TM* in all missing percentages. Therefore, the *TM* component is found to be effective in handling data containing missing values.

7 Conclusion and future work

In this paper, we propose a uni-directional graph structure learning-based multivariate time series anomaly detection method with dynamic prior knowledge. In this method, we implement a more effective uni-directional graph structure learning method to capture the one-way relationship between sensors and use dynamic prior graphs to improve the quality of the learning graph. Besides, we combine the learning graph and *TM*-based DCRNN predictor to efficiently predict the future behavior of sensors. Compared with the baseline

Table 9 Ablation Study of the Dynamic Prior Graph

Method	SWAT			WADI			SMAP			MSL		
	Pre	Re	F1	Pre	Re	F1	Pre	Re	F1	Pre	Re	F1
NPGLAD	91.2	93.9	92.5	66.9	91.1	77.2	89.0	100	94.2	99.1	100	99.4
SPGLAD	95.3	92.0	*93.6*	81.0	78.1	*79.5*	91.2	100	*95.4*	99.3	100	*99.6*
DPGLAD	95.1	93.0	**94.0**	80.7	87.8	**83.9**	94.2	100	**97.0**	99.8	100	**99.9**

The highest and second-highest results are highlighted with boldface and italics, respectively

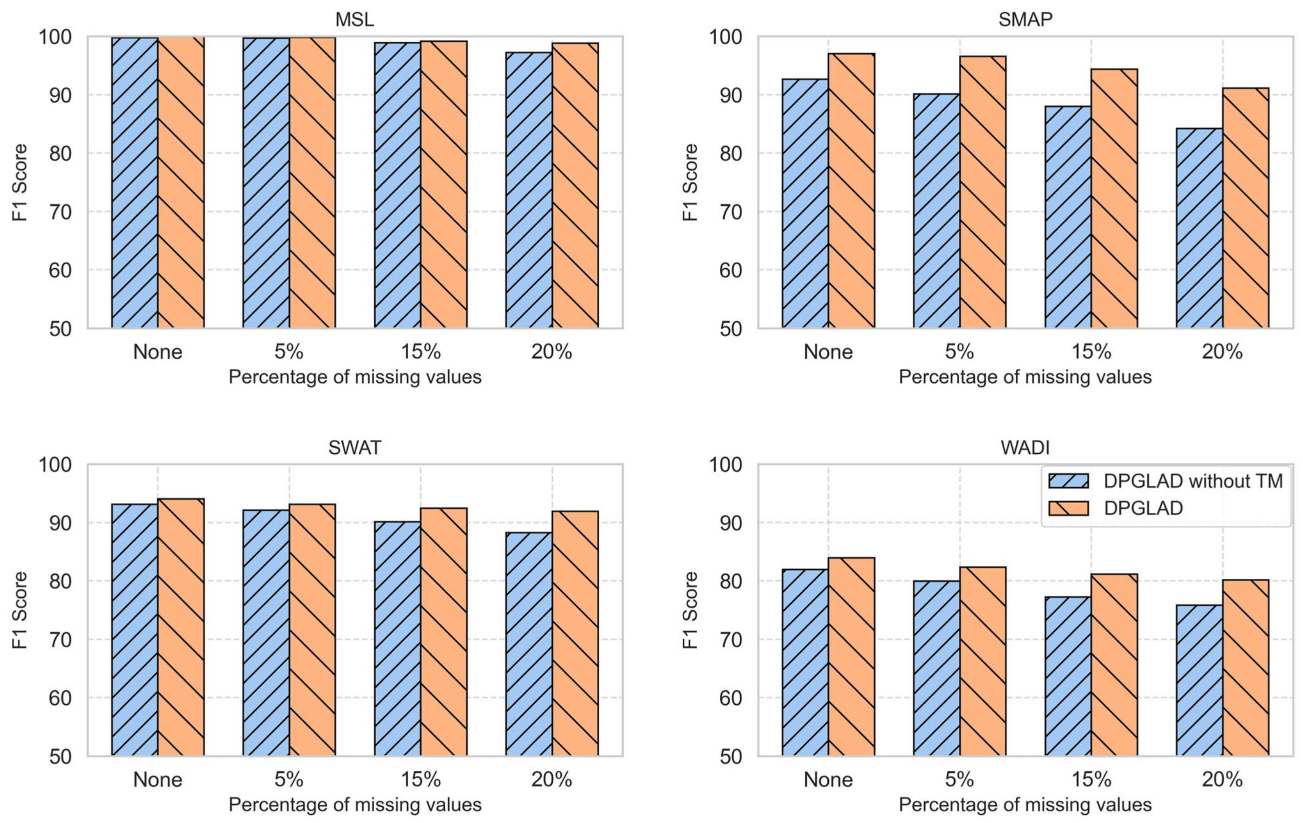

Fig. 12 The impact of missing values

on four public datasets, our proposed method achieves the best performance with short-term data while reducing the training overhead. Modeling the interconnections among sensors from multiple perspectives and improving the scalability when dealing with large graphs are future work.

Data availability The datasets used are all public datasets, and the links to obtain the datasets are as follows. SWAT: https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_SWAT/, WADI: https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_WADI/, MSL/SMAP: https://s3-us-west-2.amazonaws.com/telemanom/data.zip.

References

1. Audibert J, Michiardi P, Guyard F, et al (2020) USAD: unsupervised anomaly detection on multivariate time series. In: KDD '20: The 26th ACM SIGKDD Conference on knowledge discovery and data mining, virtual event. ACM, pp 3395–3404, https://doi.org/10.1145/3394486.3403392
2. Chen Z, Chen D, Zhang X et al (2022) Learning graph structures with transformer for multivariate time-series anomaly detection in iot. IEEE Internet of Things J 9(12):9179–9189. https://doi.org/10.1109/JIOT.2021.3100509
3. Deng A, Hooi B (2021) Graph neural network-based anomaly detection in multivariate time series. In: Thirty-Fifth AAAI Conference on artificial intelligence, pp 4027–4035, https://doi.org/10.1609/AAAI.V35I5.16523
4. Goh J, Adepu S, Junejo KN, et al (2016) A dataset to support research in the design of secure water treatment systems. In: Critical information infrastructures security: 11th International Conference, CRITIS, pp 88–99, https://doi.org/10.1007/978-3-319-71368-7_8
5. Han S, Woo SS (2022) Learning sparse latent graph representations for anomaly detection in multivariate time series. In: Proceedings of the 28th ACM SIGKDD Conference on knowledge discovery and data mining, pp 2977–2986, https://doi.org/10.1109/JIOT.2021.3100509
6. He Q, Zheng Y, Zhang C et al (2020) Mtad-tf: multivariate time series anomaly detection using the combination of temporal pattern and feature pattern. Complexity 2020:8846608. https://doi.org/10.1155/2020/8846608
7. He S, Li Z, Wang J et al (2021) Intelligent detection for key performance indicators in industrial-based cyber-physical systems. IEEE Trans Ind Inf 17(8):5799–5809. https://doi.org/10.1109/TII.2020.3036168
8. He S, Guo M, Li Z et al (2023) A joint matrix factorization and clustering scheme for irregular time series data. Inf Sci 644:119220. https://doi.org/10.1016/j.ins.2023.119220
9. He S, Guo M, Yang B et al (2023) Fine-grained multivariate time series anomaly detection in iot. Comput Mater Continua 75(3):5027–5047. https://doi.org/10.32604/cmc.2023.038551
10. Hundman K, Constantinou V, Laporte C, et al (2018) Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In: Proceedings of the 24th ACM SIGKDD International Conference on knowledge discovery & data mining, pp 387–395, https://doi.org/10.1145/3219819.3219845

11. Khoshnevisan F, Fan Z (2019) Rsm-gan: a convolutional recurrent gan for anomaly detection in contaminated seasonal multivariate time series. arXiv preprint arXiv:1911.07104
12. Li D, Chen D, Jin B, et al (2019) Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In: International Conference on artificial neural networks, Springer, pp 703–716
13. Li Y, Yu R, Shahabi C, et al (2017) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926
14. Liu FT, Ting KM, Zhou ZH (2008) Isolation forest. In: 2008 Eighth IEEE International Conference on data mining, pp 413–422, https://doi.org/10.1109/ICDM.2008.17
15. Liu Y, Zheng Y, Zhang D, et al (2022) Towards unsupervised deep graph structure learning. In: WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25-29, 2022, pp 1392–1403, https://doi.org/10.1145/3485447.3512186
16. Meng F, Gao Y, Wang H et al (2022) TSLOD: a coupled generalized subsequence local outlier detection model for multivariate time series. Int J Mach Learn Cybern 13(5):1493–1504. https://doi.org/10.1007/S13042-021-01462-X
17. Park D, Hoshi Y, Kemp CC (2018) A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. IEEE Robot Autom Lett 3(3):1544–1551. https://doi.org/10.1109/LRA.2018.2801475
18. Scheinert D, Acker A, Thamsen L, et al (2021) Learning dependencies in distributed cloud applications to identify and localize anomalies. In: 2021 IEEE/ACM International Workshop on cloud intelligence (CloudIntelligence), pp 7–12
19. Shang C, Chen J, Bi J (2021) Discrete graph structure learning for forecasting multiple time series. arXiv preprint arXiv:2101.06861
20. Su Y, Zhao Y, Niu C, et al (2019) 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, https://doi.org/10.1145/3292500.3330672
21. Sun Q, Li J, Peng H, et al (2022) Graph structure learning with variational information bottleneck. In: Proceedings of the AAAI Conference on artificial intelligence, pp 4165–4174, https://doi.org/10.1609/AAAI.V36I4.20335
22. Yang Z, Zhang G, Wu J, et al (2023) Minimum entropy principle guided graph neural networks. In: Proceedings of the Sixteenth ACM International Conference on web search and data mining, WSDM 2023, Singapore, 27 February 2023-3 March 2023, pp 114–122, https://doi.org/10.1145/3539597.3570467
23. Yang Z, Zhang G, Wu J, et al (2023) State of the art and potentialities of graph-level learning. arXiv preprint arxiv:2301.05860
24. Yue Z, Wang Y, Duan J, et al (2022) Ts2vec: Towards universal representation of time series. In: Proceedings of the AAAI Conference on artificial intelligence, pp 8980–8987, https://doi.org/10.1609/AAAI.V36I8.20881
25. Zhang C, Song D, Chen Y, et al (2019) A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In: Proceedings of the AAAI Conference on artificial intelligence, pp 1409–1416, https://doi.org/10.1609/AAAI.V33I01.33011409
26. Zhang C, Zuo W, Yin A et al (2021) ADET: anomaly detection in time series with linear time. Int J Mach Learn Cybern 12(1):271–280. https://doi.org/10.1007/S13042-020-01171-X
27. Zhang W, Zhang C, Tsung F (2022) Grelen: multivariate time series anomaly detection from the perspective of graph relational learning. In: Proceedings of the Thirty-First International Joint Conference on artificial intelligence, IJCAI-22, pp 2390–2397, https://doi.org/10.24963/IJCAI.2022/332
28. Zhao H, Wang Y, Duan J, et al (2020) Multivariate time-series anomaly detection via graph attention network. In: 20th IEEE International Conference on data mining. IEEE, pp 841–850, https://doi.org/10.1109/ICDM50108.2020.00093
29. Zong B, Song Q, Min MR, et al (2018) Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: International Conference on learning representations
30. Zou D, Peng H, Huang X, et al (2023) SE-GSL: a general and effective graph structure learning framework through structural entropy optimization. In: Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023-4 May 2023, pp 499–510, https://doi.org/10.1145/3543507.3583453