Graph structure learning-based multivariate time series anomaly detection in Internet of Things for human-centric consumer applications

Article

Accepted Version

He, S., Li, G., Yi, T., Alfarraj, O., Tolba, A., Kumar Sangaiah, A. and Sherratt, R. S. ORCID: https://orcid.org/0000-0001-7899-4445 (2024) Graph structure learning-based multivariate time series anomaly detection in Internet of Things for human-centric consumer applications. IEEE Transactions on Consumer Electronics, 70 (3). pp. 5419-5431. ISSN 1558-4127 doi: https://doi.org/10.1109/TCE.2024.3409391 Available at https://centaur.reading.ac.uk/116885/

Graph Structure Learning-based Multivariate Time Series Anomaly Detection in Internet of Things for Human-Centric Consumer Applications.

Shiming He, Genxin Li, Tongzhijian Yi, Osama Alfarraj, Amr Tolba, *Senior, IEEE*, Arun Kumar Sangaiah, and R. Simon Sherratt, *Fellow, IEEE*

***Abstract*—As the Internet of Things system becomes more popular and ubiquitous, it has also gradually entered the consumer electronics field. For example, smart home systems have numerous sensors that monitor the environment and interact with the Internet to provide smart services. A large amount of multivariate time series data generated using sensors can provide services for consumers and identify faulty systems through multivariate time series anomaly detection (MTSAD), which is crucial for maintaining system stability. However, representing the complex relationships among multivariate time series is challenging. Recently, graph neural networks and graph structure learning, which can excellently learn complex time series relationships, have been applied to multivariate time series. However, existing research on graph structure learning only constructs k-Nearest Neighbor (kNN) graphs based on the pair-wise similarity between time series. This generates a quadratic cost and only considers partial relationships among sensors. Accordingly, we propose a lightweight graph structure learning-based multivariate time series anomaly detection (GSLAD), which exploits full graph parameterization to learn the graph structure without pair-wise similarity to overcome the quadratic cost and the limited neighbor relationship. GSLAD exploits diffusion convolutional recurrent neural network (DCRNN) to extract temporal and spatial features. The results from the extensive simulations performed on four public real-world datasets demonstrate that the F1 score improved by an average of 5% with less training time compared to existing state-of-the-art methods.**

***Index Terms*—Graph attention neural network, graph structure learning, multivariate time series**

This work partially supported by the National Natural Science Foundation of China under Grant 62272062, the Science and Technology Innovation Program of Hunan Province under Grant 2023RC3139, the Scientific Research Fund of Hunan Provincial Transportation Department under Grant 202143, the Open Fund of Key Laboratory of Safety Control of Bridge Engineering, Ministry of Education (Changsha University of Science Technology) under Grant 21KB07, the Open Research Fund of the Hunan Provincial Key Laboratory of Network Investigational Technology under Grant 2018WLZC003, and the Researchers Supporting Project Number (RSP2024R102), King Saud University, Riyadh, Saudi Arabia. (*Corresponding author: Arun Kumar Sangaiah.*)

Shiming He, Genxin Li and Tongzhijian Yi are with the School of Computer and Communication Engineering, and the Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation, Changsha University of Science and Technology, Changsha 410114, China. (e-mail: smhe_cs@csust.edu.cn; mapleleavesli@stu.csust.edu.cn; yitong@stu.csust.edu.cn)

Osama Alfarraj and Amr Tolba are with the Computer Science Department, Community College, King Saud University, Riyadh 11437, Saudi Arabia. (e-mail: oalfarraj@ksu.edu.sa; atolba@ksu.edu.sa)

Arun Kumar Sangaiah is with the Department of Electrical and Computer Engineering, Lebanese American University, Byblos, Lebanon, and the International Graduate Institute of AI, National Yunlin University of Science and Technology, Taiwan (ROC). (e-mail: aksangaiah@ieee.org)

R. Simon Sherratt is with the Department of Biomedical Engineering, the University of Reading, RG6 6AY, UK. (e-mail: sherratt@ieee.org).

I. Introduction

HUman-centric is at the heart of consumer electronics. In recent years, the development of the Internet of Things, 5G, and cloud computing technologies has brought about innovations in consumer electronics and generated security risks. As the functions provided by consumer electronics become more intelligent, more related devices are included in the system. For example, in a smart home system, the router serves as the core device. It is equipped with a central processing unit (CPU), memory, and ports, and is responsible for receiving and sending data packets. When the router fails, it may affect the smart home system's functions, leading to security risks [1], [2]. There are two distinct methodologies for detecting faults and safety risks in smart devices: hardware-based safety strategies [3] and data-based approaches. Hardware-based security strategies rely on specialized hardware support, whereas data-based approaches involve identifying faults or attacks by analyzing anomalies in the data generated by smart devices. In comparison to hardware-based solutions, data-based approaches offer a more practical alternative.

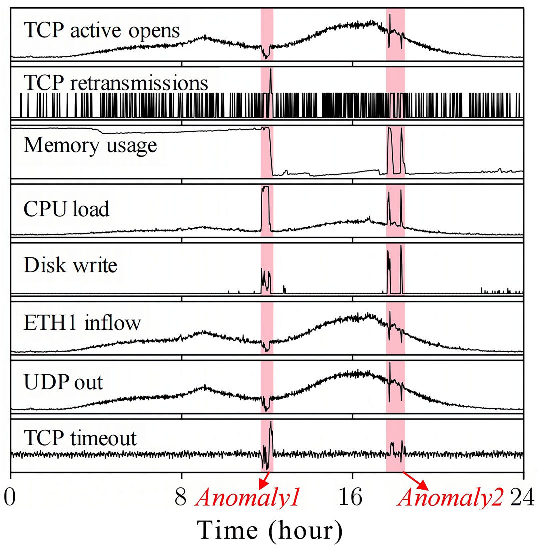

Fig. 1. An illustration of anomalies in multivariate time series. Two anomalous regions are highlighted in pink [4].

In practice, smart home devices generate multivariate indicators[1] and form multivariate time series (MTS) [5], [6].

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

Fig. 1 shows an example multivariate time series with two anomalous regions [4]. The MTS can reflect different aspects of a physical device or system and retain information more effectively than univariate time series data.

Multivariate time series anomaly detection (MTSAD) is an important and rapidly developing time series and anomaly detection sector. Existing MTSAD techniques [4], [7]–[11] focus primarily on identifying the timestamps where system errors, attacks, and behavior deviations occur.

MTS includes complex temporal and spatial correlations. MTSAD should consider each indicator's temporal relationship and the inter-correlations between indicators or sensors. As a result, Graph Neural Networks (GNN) models are incorporated into MTSAD to extract spatial features [12], [13]. GNNs are particularly effective in capturing the structural information and relationships within graphs. Moreover, the relationships among sensors have been exploited to improve forecasting and anomaly detection accuracy. Hence, most GNNs rely on a known graph structure. However, the graph structure is not always available, that is, the sensor relationships are hidden or costly to access. Therefore, graph structure learning (GSL), a system that can learn the unknown or hidden graph structure joint using the downstream GNN task, is proposed [14], [15]. However, multivariate time series anomaly detection based on graph structure learning and GNNs still faces many challenges.

- **Most existing methods learn a k-Nearest Neighbor (kNN) graph based on pair-wise similarity, introducing a quadratic cost, and only considering the partial relationships among sensors.** They randomly initialize the learnable node embeddings [14] or acquire them from the input data, measure pair-wise similarity among the node embeddings, and count the top-k closest nodes as the neighbors. In addition, node embeddings or acquired weights are learned with the downstream task. Consequently, a quadratic cost is generated to obtain the pair-wise similarity among nodes. The node only has k neighbors in the learned graph, ignoring the correlation with the remaining nodes. As a result, the correlation between the sensors is not extracted effectively.
- **Existing methods utilize static GNNs, which cannot capture the temporal feature.** Based on the known graph [12] or learned graph structure [14], these methods exploit the graph convolution or attention neural network to extract features for anomaly detection, where GNN treats the time series as static feature and ignore the order in each indicator. Thus, these methods cannot extract temporal features with static GNN effectively.
- **Long-term historical data is required when the Transformer model is used for prediction and anomaly detection.** Even though Graph Learning with Transformer for Anomaly detection (GTA) [15] directly learns a global graph structure, it utilizes the Transformer model for prediction. These Transformer models, with their self-attention and multi-head attention mechanisms, effectively enrich the extracted features. In addition, the complexity of the model's attention mechanisms increases the model training costs. The Transformer model also requires long-term historical data to improve its performance. In contrast, GTA requires 60 timestamps as input data, while only 5 timestamps are needed in Graph Deviation Network (GDN) [14]. Shortening the length of the input historical data for the Transformer results in a significant reduction in detection accuracy.

To overcome these three issues, we design a multivariate time series anomaly detection based on lightweight graph structure learning and recurrent graph convolution neural network (GSLAD). Our major contributions can be summarized as follows:

- We employ full graph parameterization to learn the graph structure without pair-wise similarity to overcome the quadratic cost and the limited neighbor relationship.
- We utilize diffusion convolutional recurrent neural network (DCRNN) instead of the traditional graph convolution network (GCN) to extract the temporal and spatial features. DCRNN is a powerful spatio-temporal graph convolutional neural network that accurately predicts future time series with low complexity and reduced input data requirements.
- Extensive simulations on four public real-world datasets demonstrate that our algorithm can achieve a higher F1 score and lower training costs than existing methods. This shows that our method is useful for rapidly and accurately monitoring electronic consumer product health.

The remainder of this paper is organized as follows. First, the related work is reviewed in Section II, and the problem description is presented in Section III. Then, the overview and detailed GSLAD steps are outlined in Section IV. Next, Section V discusses the performance evaluation via experiments. Finally, we conclude our research and discuss future work in Section VI.

II. Related work

We classify the related works into three types according to the features: temporal feature, temporal-spatial feature, and learnable spatial feature.

A. Temporal Feature-based Methods

Several temporal feature-based methods exist that can be applied in MTSAD. Long Short-Term Memory (LSTM) is first used in MTSAD due to its ability to process sequence data. LSTM-NDT [7] uses LSTM to achieve high prediction performance and provides a nonparametric, dynamic, and unsupervised anomaly thresholding approach to detect anomalies.

Recurrent neural network joints with generative models are widely applied in time series anomaly detection to reconstruct the time series. For instance, LSTM-VAE [8] projects multimodal observations and temporal dependencies into a latent space and reconstructs the expected distribution. In doing so, it regards the reconstruction error as an anomaly score. DAGMM [9] trains a deep autoencoding and Gaussian mixture model simultaneously to produce a low-dimensional representation and reconstruction error in order to detect anomalies. Omnianomaly [4] utilizes a stochastic recurrent neural network to capture the robust representations of normal patterns and reconstruct the observations. In addition, MAD-GAN [10] exploits LSTM as the base model in the GAN framework to capture the temporal correlation of time series distributions, extract the latent interactions, and detect anomalies through discrimination and reconstruction.

LSTM can model the temporal dependence of time series data well. Due to the heterogeneity between data indicators [16], the correlation between metrics also needs to be considered when performing multivariate time series anomaly detection.

B. Temporal-spatial Feature-based Methods

Graph Neural Networks are specifically designed to process and analyze graph-structured data. They excel at capturing correlations between indicators, enabling accurate behavior prediction. For instance, MTAD-TF [12] exploits multi-scale convolution and graph attention networks to capture information in temporal patterns, and regards the root means square between the predicted value and the ground truth as the anomaly score. MTAD-GAT [13] attempts to mine the correlations between different univariate time series and the temporal dependencies within each of them using a graph attention network. It integrates prediction and reconstruction-based approaches to detect anomalies. Moreover, Arvalus and its variant D-Arvalus [17] treat system components as nodes, and their dependencies and positions as edges to improve anomaly identification and location. The above mentioned methods make assumptions about the known graph structure (e.g., Arvalus and D-Arvalus), or they treat the MTS as a complete graph (e.g., MTAD-TF and MTAD-GAT). As a result, the computational cost for a complete graph rises with an increase in dimension.

C. Learnable Spatial Feature-based Methods

To process the MTS without a ground truth graph structure at a low cost, GSL is incorporated into MTSAD. For instance, GDN [14] learns a node embedding and constructs a kNN graph using the learned embedding's similarities. Then, an attention-based GNN predicts behavior by extracting the dependence relationships between the nodes, and the prediction error is used for the anomaly score. Furthermore, GTA [15] directly treats the adjacency matrix's elements as learnable parameters to automatically learn a graph structure. It models temporal dependency using a Transformer-based architecture.

However, the pair-wise similarity generates a quadratic cost and the kNN graph only considers partial relationships among sensors. In addition, Transformers require long-term historical data. Therefore, lightweight GSL is necessary for accurate MTSAD.

III. Problem Description and Preliminaries

A. Problem Description

MTS contains a large number of equally-spaced sampling and continuous observation points with K indicators and N timestamps, denoted by $X = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^K)^T \in R^{K\times N}$. Likewise, the i-th indicator can be represented by $\boldsymbol{x}^i = (x_1^i, x_2^i, \ldots, x_N^i)$. The t-th timestamp contains the indicators' K values denoted by $\boldsymbol{x}_t = (x_t^1, x_t^2, \ldots, x_t^K)^T$. Anomaly ground truths are available at all timestamps. When the label $\boldsymbol{y_t}$ is 1, it means that the value of t-th timestamp $\boldsymbol{x}_t$ is abnormal.

Definition 1 (Anomaly detection): Given the continuous $w + 1$ observations $\boldsymbol{x_{t-w}}, \ldots, \boldsymbol{x_t}$, the MTSAD task aims to identify whether the last observation $\boldsymbol{x_t}$ is anomalous.

B. Graph Structure Learning

GSL is a promising solution that learns the unknown or hidden graph topology joint with the downstream GNN task.

Definition 2 (Graph structure learning): Given the time series $X \in R^{K\times N}$, the graph structure learner's purpose is to obtain a graph $G = (V, E)$ and its topology or adjacency matrix $A \in R^{K\times K}$. The node v in the graph represents a sensor that produces an indicator, while the hidden relationship between the sensors is represented by the edges E. Furthermore, the adjacency matrix A stores the edge information in the graph, reflecting the underlying dependencies among indicators. In this instance, the adjacency matrix's elements are composed of 0 and 1. If $A_{i,j}$ is 1, it represents an edge between nodes i and j. On the contrary, if $A_{i,j} = 0$, there is no edge between nodes i and j.

There are three types of GSL methodologies: metric-based, neural network and direct approaches [18].

- Metric-based approaches utilize a metric function to calculate the similarity between the features or embeddings of node pairs, which is then used as the edge weight. Edges tend to connect similar nodes, assuming that network homogeneity is present. For example, AGCN [19] utilizes the generalized Mahalanobis distance of the node pair features with a Gaussian kernel to construct the adjacency matrix. Similarly, GRCN [20] uses the inner product of node embeddings to obtain the edge weights.
- Neural network approaches utilize deep neural networks to model edge weights based on node features or representations. For example, GLN [21] iteratively learns graph structures from local and global node embeddings by a simple single-layer neural network. PTDNet [22] obtains the adjacency matrix by a multilayer perceptron.
- Direct approaches treat the target graph's adjacency matrix as independent variables and jointly optimize the downstream task's model parameters, independent of the node representation. However, jointly optimizing the adjacency matrix and model parameters often involves non-differentiable operations, making gradient-based optimization methods infeasible. To address this, LDS [23] models the edges between each node pair by sampling Bernoulli distributions with learnable parameters and approximates the solution using hypergradient estimation. In contrast, GLNN [24] employs an alternating optimization scheme, iteratively updating the adjacency matrix and model parameters.

On the one hand, metric-based and neural network approaches have the advantage of end-to-end training but come with quadratic computation costs. On the other hand, direct approaches offer flexibility but require addressing the non-differentiability issue. Nonetheless, GSL is a thriving research field applied in MTSAD and in various time series prediction and classification tasks. Therefore, it is crucial to select an appropriate GSL approach that can be flexibly adapted to different downstream tasks.

IV. Proposed Methodology

A. Overview

In most real-world scenarios, there are complex topological relationships among indicators, and these relationships can be presented as a graph. Each indicator is regarded as a node and relationships between the indicators are treated as edges between nodes in the graph. The previous method [14], [25] calculates the similarity between each node pair based on the indicator features and connects the top-k most similar indicators as neighbors. This approach results in a kNN graph, which only considers the k nearest neighbor relationships. Therefore, to fully capture the relationships, we leverage the direct approaches in graph structure learning and propose a multivariate time series anomaly detection method based on lightweight graph structure learning and recurrent graph convolution neural networks. This allows us to comprehensively consider the relationships within the graph structure.

The process of GSLAD can be summarized as follows: firstly, the graph structure learner learns the adjacency matrix A. Secondly, based on the adjacency matrix A, a DCRNN-based predicts the value $\hat{\boldsymbol{x}}_t$ of the t-th timestamp from the historical observations $\boldsymbol{x}_{t-w}, \ldots, \boldsymbol{x}_{t-1}$. Then, the adjacency matrix is jointly trained with the prediction task, and the normalized prediction error between the predicted value $\hat{\boldsymbol{x}}_t$ and the ground truth $\boldsymbol{x}_t$ is treated as the anomaly score. Finally, a grid search algorithm is used to select a threshold for best detection accuracy. Once the anomaly score at timestamp t exceeds the threshold, timestamp t is taken as an anomaly.

To support the above process, GSLAD involves four main components: the graph structure learner, a DCRNN-based predictor, the anomaly score calculation, and the threshold selection.

(1) **Graph structure learner.** It learns the relationship between indicators. Hence, we employ a full graph parameterization learner due to the quadratic costs generated by metric-based and neural network approaches. This learner directly models each element of the adjacency matrix A with independent parameters. Furthermore, to handle the non-differentiability issue, we employ the Gumbel–Softmax reparameterization technique. This approach offers a more computationally efficient alternative than methods with quadratic costs.

(2) **DCRNN-based predictor.** It predicts the future value. DCRNN captures spatial relationships through diffusion convolution and temporal dependencies using an encoder–decoder architecture, enabling the simultaneous modeling of temporal and spatial features. Consequently, we opt to utilize DCRNN instead of GCN for prediction. According to the sliding window, a MTS is divided into multiple subsequences. Based on the adjacency matrix obtained by the graph structure learner, the recurrent graph convolution neural network predicts the following value $\hat{x}_t$ of the subsequence $\boldsymbol{x}_{t-w}, \ldots, \boldsymbol{x}_{t-1}$.

(3) **Anomaly score calculation.** The anomaly score is derived from the normalized prediction error between the predicted value $\hat{x}_t$ and the ground truth x_t. High anomaly scores indicate that the ground truth may deviate from the normal pattern, suggesting that anomalies are more likely to occur.

(4) **Threshold selection.** When the anomaly score exceeds the selected threshold, this indicates that an anomaly has occured. The threshold should be selected carefully. Therefore, we employ a grid search algorithm to determine the threshold that yields the best detection accuracy.

The notations in this paper are shown in Table I. The following subsection describes each part in detail.

TABLE I
List of Notations

Notation	Meaning
X	Multivariate time series
$\boldsymbol{x}^i$	The i-th indicator values
$\boldsymbol{x}_t$	The indicator values at timestamp t
$\hat{\boldsymbol{x}}$	The prediction
K	The number of indicators
N	The number of timestamps
y_t	The label of the t-th timestamp
G	Graph
A	Adjacency matrix
g_i	The samples from the Gumbel distribution
π_i	The probabilities of the $i-th$ class
τ	The temperature parameter
$H_{(t)}$	The hidden features at timestamp t
$Err_{\mathrm{i}}(t)$	The prediction error at timestamp t for sensor i
$s(t)$	The anomaly scores

B. Graph Structure Learner

To address the quadratic cost issue associated with pair-wise similarity in graph structure learning, we employ a full graph parameterization learner. This approach treats each element of the adjacency matrix as a learnable parameter, enabling joint optimization with the downstream task's model parameters. However, jointly optimizing the adjacency matrix and model parameters presents challenges. For instance, the adjacency matrix A consists of binary and discrete variables, rendering it non-differentiable. As a result, the backpropagation algorithm cannot accurately compute the parameter gradients.

Therefore, we apply a reparameterization technique (i.e., the Gumbel-Softmax method) [26], which is a continuous distribution sampled approximately from a class distribution. As a result, the discrete parameters can be represented by continuous parameters and parameter gradients can be easily computed using the Gumbel–Softmax approach.

First, the Gumbel-Max trick is a simple and effective method for sampling from a class distribution, and its formula is defined by Eq.(1). Gumbel-Max samples the i-th class using the class probability π_i.

$$\begin{aligned} z &= \underset{i}{\operatorname{argmax}}\left(g_i + \log \pi_i\right) \\ g_i &= -\log(-\log(u)), u \sim Uniform(0,1) \end{aligned} \tag{1}$$

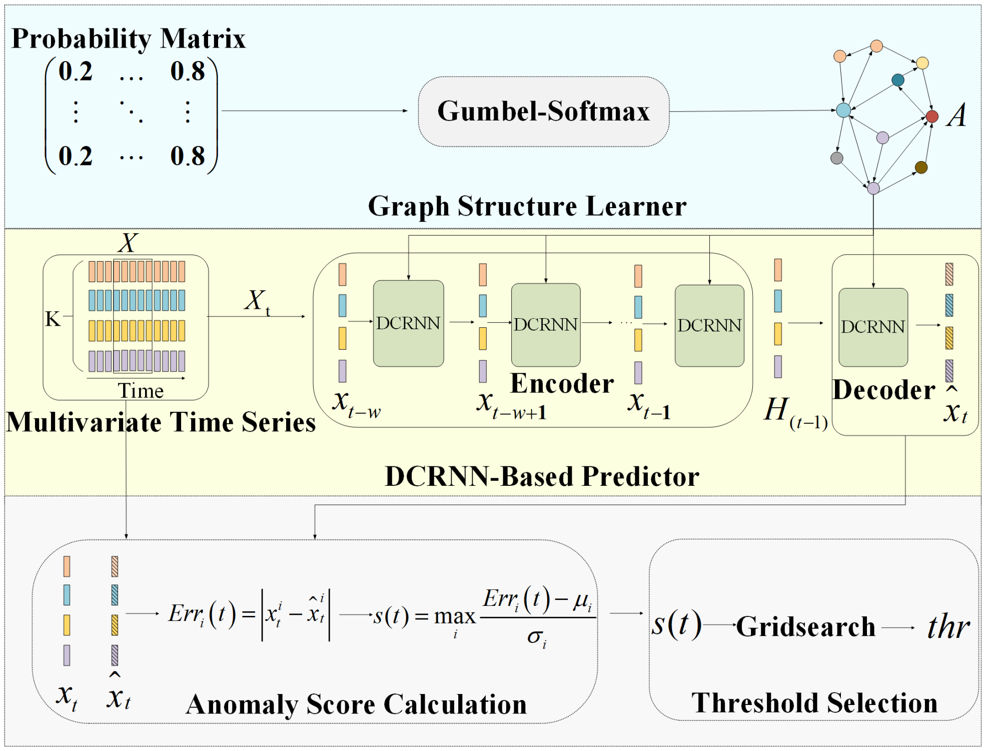

Fig. 2. In the GSLAD framework, the graph structure learner is used to construct the topological relationships among indicators, the DCRNN-based predictor obtains the indicators' predictions, the anomaly score calculation is used to calculate the errors between ground truths and predictions, and threshold selection is employed to determine whether anomalies are present based on the anomaly scores.

where u represents samples drawn from the Uniform(0,1) distribution, g_i is the Gumbel distribution obtained by computing u, π_i denotes the probability of the i-th class, and z shows the sampled class. The $argmax$ function is not differentiable. Thus, we search for a smooth $argmax$ approximation (i.e., the $softmax$ function). The $softmax$ function is used as a continuously differentiable approximation to the parameter's maximum value, as shown below:

$$z=\operatorname{softmax}\left(g_i+\log \pi_i\right) \tag{2}$$

For the GSL sampling problem, there are $n*n$ discrete variables in the adjacency matrix A and only two classes. Therefore, the Gumbel-Softmax for the adjacency matrix is finally defined as Eq.(3):

$$z_1^{i,j}=\frac{\exp\left(\left(\log \pi_1^{i,j}+g^{i,j}\right)/\tau\right)}{\sum_{v\in\{0,1\}}\exp\left(\left(\log \pi_v^{i,j}+g^{i,j}\right)/\tau\right)} \tag{3}$$

where $g^{i,j}$ is the gumbel distribution, and $\pi_1^{i,j}$ shows the value of row i and column j of a probability matrix $\pi_1 \in R^{K\times K}$. This represents the probability that node i is connected to node j in the graph. The probability matrix π_1 leads to π_0 ($\pi_0^{i,j}=1-\pi_1^{i,j}$), which represents the probability that there is no edge connecting nodes i and j. τ is the temperature parameter, which controls the smoothness of the sampling process. When τ approaches zero, the $softmax$ tends to the $argmax$, and samples (z_0, z_1) from the Gumbel-Softmax distribution tend to be one-hot vectors, i.e., one element is 1 and another is 0. Finally, the row i and column j in the adjacency matrix $A_{i,j}$ is set to $z_1^{i,j}$, which is set to 1 with probability $\pi_1^{i,j}$.

In our graph structure learner, we begin by randomly initializing the probability matrix π_1. The adjacency matrix is then generated by performing Gumbel–Softmax sampling on the probability matrix. This adjacency matrix represents whether or not there are connections between sensors and is used in the downstream prediction task. Afterward, the probability matrix is updated by loss backpropagation for the downstream task, as shown in Fig. 3. As the probability matrix is updated, the connections between sensors, i.e., the adjacency matrix, are also updated. Eventually, we obtain an adjacency matrix that is most suitable for the current downstream task.

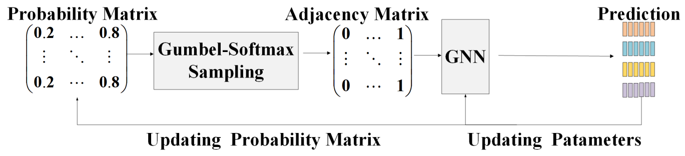

Fig. 3. The graph structure learning process.

C. DCRNN-based Predictor

The DCRNN-based predictor predicts future values based on past values and facilitates anomaly detection according to the prediction errors. At timestamp t, we define subsequence X_t according to a sliding window size w over the historical time series data as follows:

$$X_t=\left[\boldsymbol{x}_{t-w},\ldots,\boldsymbol{x}_{t-1}\right]\in R^{K\times w}. \tag{4}$$

The adjacency matrix A and the subsequence X_t are fed into the predictor to predict the value $\hat{x}_t$ of the next timestamp.

Moreover, DCRNN captures spatial dependency using diffusion convolution, and the temporal dependency using the encoder–decoder architecture. This unique combination allows DCRNN to simultaneously model and capture both temporal and spatial features. Therefore, we utilize DCRNN instead of GCN in our approach.

For capturing spatial dependency, DCRNN exploits a L-step diffusion process based on a random walk to model the proximity between nodes and perform feature aggregation based on the L-hop neighbor node proximity. It also utilizes an L-step diffusion convolutional layer to replace L-graph convolutional layers. The diffusion convolution $\circ$ is defined as follows:

$$W_Q \circ Y = \sum_{l=0}^{L} \left(w_{l,1}^{Q} \left(D_O^{-1}A\right)^{l} + w_{l,2}^{Q} \left(D_I^{-1}A^T\right)^{l} \right) Y \tag{5}$$

where D_O and D_I are the out- and in-degree matrices, $w_{l,1}^{Q}$ and $w_{l,2}^{Q}$ denote the model parameters, and L is the diffusion degree. Eq. (6) gives the expansion term of Eq. (5) when L=2. The first term $(w_{0,1}^{Q}Y + w_{0,2}^{Q}Y)$ is the node's own information. $\left(D_O^{-1}A\right)$ and $\left(D_I^{-1}A^T\right)$ obtain the first-order input and output neighbors, respectively. Therefore, the second line in Eq. (6) is used to aggregate the first-order neighbor information. Similarly, $\left(D_O^{-1}A\right)^2$ and $\left(D_I^{-1}A^T\right)^2$ obtain the second-order input and output neighbors, respectively. The third line is used to aggregate the second-order neighbor information. L adjusts the information about the neighbor nodes which can be aggregated.

$$\begin{aligned} &W_Q \circ Y = w_{0,1}^{Q}Y + w_{0,2}^{Q}Y + \\ &w_{1,1}^{Q}\left(D_O^{-1}A\right)Y + w_{1,2}^{Q}\left(D_I^{-1}A^T\right)Y + \\ &w_{2,1}^{Q}\left(D_O^{-1}A\right)^2 Y + w_{2,2}^{Q}\left(D_I^{-1}A^T\right)^2 Y \end{aligned} \tag{6}$$

For each node, the diffusion convolutional layer aggregates information from neighboring nodes up to L steps away in both the outward and inward directions. This allows the model to capture the influence of distant nodes in the graph structure, providing a more comprehensive understanding of the spatial dependencies in the data.

For capturing temporal dependency, DCRNN uses the Gated Recurrent Unit (GRU) which is a simple yet powerful variant of recurrent neural networks (RNNs). Specifically, DCRNN uses diffusion convolution instead of matrix multiplication in the GRU, where the diffusion convolution is the frequency domain graph convolution. The specific formula of DCRNN is as follows:

$$R_{\mathrm{t}} = \mathrm{sigmoid}\left(W_R \circ \left[x_t || H_{(t-1)}\right] + b_R\right) \tag{7}$$

$$C_{\mathrm{t}} = \tanh\left(W_C \circ \left[x_t || (R_t \odot H_{(t-1)})\right] + b_C\right) \tag{8}$$

$$U_{\mathrm{t}} = \mathrm{sigmoid}\left(W_U \circ \left[x_t || H_{(t-1)}\right] + b_U\right) \tag{9}$$

$$H_{(t)} = U_t \odot H_{(t-1)} + (1 - U_t) \odot C_t \tag{10}$$

where $||$ is the concatenation operation of two features, $\circ$ is the diffusion convolution operation, U_t is the update gate, R_t is the reset gate, and C_t is the candidate hidden state. An unit in DCRNN takes the node feature $H_{(t-1)}$ at timestamp $t-1$ and the value x_t at timestamp t as inputs. The update gate, reset gate, and the candidate hidden state are generated by diffusion convolution operation. These inputs are processed to obtain the node feature $H_{(t)}$ at the timestamp t. The structure of a DCRNN unit is given in Fig. 4.

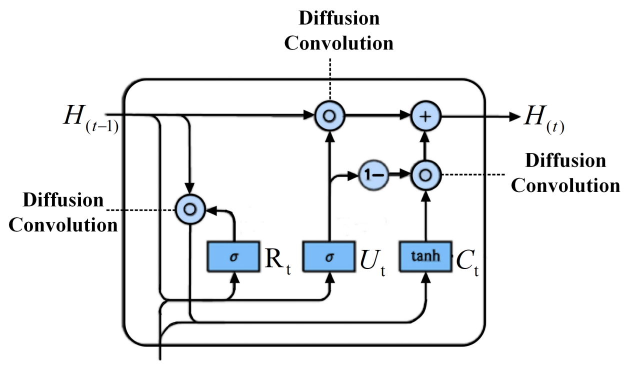

Fig. 4. The structure of a DCRNN unit.

Addtionally, DCRNN uses a sequence-to-sequence architecture to predict the value of the next timestamp. Its process can be summarized as follows.

In the encoder part of the model, the node 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 past timestamps, resulting in the total node feature $H_{(t-1)}$ of the subsequence, as shown in Fig. 2.

Regarding the decoder model, the total node feature $H_{(t-1)}$ is used as the input. After passing through a layer in the decoder model, a hidden feature is obtained along with the predicted value $\hat{x}_t$ at timestamp t.

D. Loss Function

There are two tasks: prediction task and graph structure learning task. For prediction task, we use the mean absolute error as the loss function $loss_p$ to ensure that the predictions are as close as possible to the ground truth value, as shown below:

$$loss_p = \frac{1}{K}\sum_{i=1}^{K} |\hat{x}_t^i - x_t^i| \tag{11}$$

where $\hat{x}_t^i$ and x_t^i are respectively the prediction and the ground truth of i-th indicator value in timestamp t.

For graph structure learning task, to improve the quality of the learned graph, some prior knowledge can be added to the model training process. When the graph structure is unknown, some properties of the kNN graph are still reasonable knowledge. Therefore, we utilize the kNN graph, denoted as A', to incorporate prior knowledge. This graph represents the similarity between indicator values. By adding a regularization term $loss_g$ to the model training process, we ensure that the learned graph structure A is consistent with our prior knowledge. The regularization term $loss_g$ is defined as the cross-entropy loss between the prior knowledge graph A' and the learned graph structure A. This regularization loss helps enforce the desired graph structure during the training process.

$$loss_g = \sum_{ij} -A'_{i,j} \log A_{i,j} - (1 - A'_{i,j}) \log(1 - A_{i,j}) \tag{12}$$

$$A'_{i,j} = \frac{\boldsymbol{x}^i \bullet \boldsymbol{x}^j}{||\boldsymbol{x}^i|| \bullet ||\boldsymbol{x}^j||} \tag{13}$$

To balance two loss functions, we introduce the balancing parameter λ. The total loss function of the model is defined as Eq. (14).

$$loss = loss_p + \lambda loss_g \tag{14}$$

E. Anomaly Score and Threshold Selection

1) Anomaly Score: Firstly, the comparison between the ground truth value and the predicted value is conducted at timestamp t, and the resulting prediction error $E\mathrm{rr_i}(t)$ is calculated for sensor i as follows:

$$E\mathrm{rr_i}(t) = |x_t^i - \hat{x}_t^i| \tag{15}$$

Subsequently, we performed a standard normalization of the prediction error and determined the anomaly score at timestamp t by determining the highest value among all sensors as follows:

$$s(t) = \max_i s_i(t) = \max_i \frac{Err_i(t) - \mu_i}{\sigma_i} \tag{16}$$

where μ_i and σ_i are the mean and standard deviation of $E\mathrm{rr_i}(t)$ respectively.

Finally, if the anomaly score at timestamp t exceeds the threshold, timestamp t is marked as an anomaly.

2) Threshold Selection: To determine the optimal threshold, we employ employ a grid search methodology. This involves iteratively adjusting the threshold within a predefined range and measuring the performance metric for each threshold setting. The threshold that yields the best performance metric is then selected as the final threshold. To conduct the grid search, we set the upper and lower limits of the threshold range as the maximum and minimum values of $s(t)$, respectively. Then, we use a step size of 0.01 to ensure thorough threshold evaluation within this range. By exhaustively evaluating every threshold, we aim to identify the threshold value that achieves the highest F1 score. This serves as the threshold for optimal performance. Furthermore, we implement a point-adjust strategy to fine-tune the anomaly score [4].

V. Performance Analysis

This section describes the experiments we conducted to answer the following research questions (RQs):

- RQ1: How effective and efficient is GSLAD?
- RQ2: How effective is GSLAD with different components?
- RQ3: How robust is GSLAD with different parameters?
- RQ4: How does the GSLAD graph learning work during anomaly detection? What is the relationship between the learned graph and the predictions?
- RQ5: How proficient is GSLAD in detecting anomalies in the filed of consumer electronics products (CEP)?

TABLE II
Dataset description

Dataset	SWAT	WADI	MSL	SMAP
# Features	51	127	55	25
# Training length	49500	76297	58317	135183
# Testing length	45000	17280	73729	427617
# Total length	94500	93577	132046	562800
# Anomalies	5387	1035	7904	56146
Anomaly ratio(%)	11.97%	5.99%	10.72%	13.13%

A. Datasets

In our experiments, we use four datasets widely used in the field of anomaly detection in IoT systems to evaluate the model performance [11], [15]. Table II presents datasets' statistics, including the number of features, the training and testing time series length, the number of anomalies, and the anomaly ratio. It is important to note that all the training time series exclusively consist of normal instances, while the anomalies are present only in the testing time series. Given that the anomalies in these datasets represent actual occurrences and cannot be artificially injected, the anomaly ratio remains static throughout the evaluation. Furthermore, the sample number can be calculated as the time series length minus the window size. The four datasets are described below:

1) The Safe Water Treatment (SWAT) dataset[2] comes from a water treatment testbed coordinated by the Public Utilities Authority of Singapore. The collection process lasts 11 days and the system operates 24 hours a day, recording the network traffic and values obtained by all 51 sensors and actuators.

2) The Water Distribution (WADI) dataset[3] is a distribution system consisting of a large number of water distribution pipelines. WADI, as an extension of the SWAT testbed, forms a more complete and realistic water treatment, storage, and distribution network. The dataset includes a total of 16 days of continuous operations, of which 14 days are regular operations and 2 days are attack scenarios. The entire testbed contains 127 sensors and actuators.

3) The Mars Science Laboratory rover (MSL) dataset contains sensor and actuator data collected from the Mars rover by the National Aeronautics and Space Administration (NASA). MSL includes a total of 55 metrics for 27 entities.

4) The Soil Moisture Active Passive satellite (SMAP) dataset is a collection of soil samples and telemetry gathered by NASA using the Mars rover. SMAP contains a total of 25 metrics for 55 entities.

Among the four datasets, SWAT and WADI are both generated from industrial water treatment IoT system sensor monitoring, while MSL and SMAP are both from spacecraft IoT systems. To handle the large amount of raw data in the SWAT and WADI datasets, a down-sampling technique is applied. It takes the median value within each 10-second window. Once an anomaly occurs within a 10-second window, it is marked as abnormal.

[2]https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_swat/
[3]https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_wadi/

B. Evaluation Metrics

We consider the detection accuracy and training time as evaluation metrics. Precision (Prec), recall (Rec), and F1 score (F1) are often used as evaluation metrics for anomaly detection accuracy, as shown in Eqs. (17), (18.) and (19). F1 score effectively balances precision and recall, providing a comprehensive evaluation in anomaly detection scenarios where imbalanced datasets are commom. Therefore, F1 scores are widely used as the metric of detection performance [13]–[15].

$$\mathrm{Prec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{17}$$

$$\mathrm{Rec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{18}$$

$$F1 = \frac{2 \times \mathrm{Prec} \times \mathrm{Rec}}{\mathrm{Prec} + \mathrm{Rec}}. \tag{19}$$

where TP, TN, FP, and FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively. Furthermore, the training time is the running time of each epoch during training.

C. Settings

1) Parameters: For GSLAD , the window size w is 12, the temperature parameter τ is 0.1, and the loss balance weight λ is 1 according to the experiment result in the section V-F2. The number of neighbors k in the prior kNN graph is 10. We take the Adam optimizer, the learning rate is 0.001, the batch size is 64, and the epoch number is 30. For DCRNN, the diffusion degree L is 3, the hidden dimensions of one DCRNN unit are 64, and the dimensions of node feature H_t are 3264.

For GDN and GTA, we use their default experimental parameters, where the window sizes of GDN and GTA are 5 and 60, respectively.

We run each experiment five times and report the mean value. All the experiments are conducted on the Google Colab with NVIDIA Tesla T4 Graphics Processing Units (GPUs).

2) Baselines: We compare GSLAD with 11 machine and deep learning methods, IF, AE, DAGMM, LSTM-NDT, LSTM-VAE, MAD-GAN, OmniAnomaly, USAD, MTAD-GAT, GDN, and GTA. Table III presents a comprehensive comparison of the various methods' characteristics, including method type, features, data requirements, application field, and complexity.

- AE: Autoencoder is utilized to reconstruct the input data, and the resulting reconstruction error is used as the anomaly score.
- IF [27]: 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 [9]: 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 [7]: It uses LSTM to achieve high prediction performance and provides a nonparametric, dynamic, and unsupervised anomaly thresholding method to detect anomalies.
- LSTM-VAE [8]: It projects multimodal observation and temporal dependencies into a latent space and reconstructs the expected distribution through LSTM-based VAE.
- MAD-GAN [10]: It exploits LSTM as the base model in the GAN framework to capture the temporal correlation of time series distributions.
- OmniAnomaly [4]: It is a prior-driven stochastic model for timestamp anomaly detection that directly returns the reconstruction probability.
- USAD [11]: It adversarially trains an encoder-decoder framework to achieve rapid and efficient training.
- MTAD-GAT [13]: It treats the relationship between indicators as a complete graph and utilizes graph attention neural networks for anomaly detection.
- GDN [14]: 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.
- GTA [15]: It involves automatically learning a graph structure and utilizes graph convolution and Transformer-based architecture to model temporal dependency.

D. RQ1. The GSLAD Performance

1) Accuracy: The anomaly detection results in terms of precision, recall, F1 score, and percent delta between GSLAD and the second-best method for all datasets are shown in Table X. The GSLAD performance is significantly better than that of the other methods. For the SWAT and WADI datasets, GSLAD achieves the best F1 scores of 95.13 and 83.8, respectively. Similarly, GSLAD outperforms the baselines on the SMAP and MSL datasets; it achieves F1 scores of 97.2 and 98.7, respectively. In contrast, the performance of WADI with all the methods is lower than other datasets, because the WADI dataset has the lowest anomaly rate, as shown in Table II. However, GSLAD still shows the best performance on the WADI dataset due to the full graph structure learning. Although GSLAD only improves the F1 score by 2-3% compared with GTS on the WADI dataset, GSLAD greatly reduces the computational complexity and thus the training time as shown in Table V. Therefore, GSLAD is more suitable for consumer electronics and IoT products.

Fig. 5 shows the distribution of F1 scores for the four datasets over multiple experiments. GSLAD has stable anomaly detection accuracy. Therefore, it displays improved effectiveness compared to existing methods even under high-dimensional time series and imbalanced sample scenarios, which is very important in practical applications.

2) Training Time: We select all the existing GSL-based methods (i.e., GDN and GTA) for training time comparison due to their superior performance compared to other models. The results are shown in Table V. our method's training time is less than that of GTA and more than that of GDN for all datasets. GSLAD exploits DCRNN for time series prediction, while GDN utilizes a static GNN. Although GSLAD has more

TABLE III
THE COMPARISON OF EXISTING MTSAD METHODS

Method	Type	Features Temporal	Features Indicator	Features Graph	Normal training data	Application field	Complexity
IF	Machine Learning					All fields	Low
AE	Deep Learning				✓	All fields	Low
DAGMM	Deep Learning				✓	Intrusion detection system, medical disease detection	Middle
LSTM-NDT	Deep Learning	✓			✓	Spacecraft monitoring systems	Low
LSTM-VAE	Deep Learning	✓			✓	Robot-assisted feeding system	Mid
MAD-GAN	Deep Learning	✓			✓	Cyber-physical systems	High
OmniAnomaly	Deep Learning	✓			✓	Spacecraft monitoring systems, IT system operations	High
USAD	Deep Learning	✓			✓	Cyber-physical systems, IT system operations	Low
MTAD-GAT	Deep Learning	✓	✓		✓	Spacecraft monitoring systems	High
GDN	Deep Learning		✓	✓	✓	Cyber-physical systems	Low
GTA	Deep Learning	✓	✓	✓	✓	Cyber-physical systems	High

*Temporal means the temporal feature, indicator means the inter-indicator feature, and graph means learnable graph structure features. ✓ means that the method considers the feature or training data should be normal.

TABLE IV
PRECISION, RECALL AND F1 SCORE ON SWAT AND WADI.

Method	SWAT Prec	SWAT Rec	SWAT F1	WADI Prec	WADI Rec	WADI F1	SMAP Prec	SMAP Rec	SMAP F1	MSL Prec	MSL Rec	MSL F1
IF	96.2	73.15	83.11	62.41	61.55	61.98	44.23	51.05	46.71	56.81	67.4	59.84
AE	72.63	52.63	61.03	34.35	34.35	34.35	72.16	97.95	77.76	85.35	97.48	87.92
DAGMM	27.46	69.52	39.37	54.44	26.99	36.09	63.34	99.84	71.24	75.62	98.03	81.12
LSTM-NDT	77.78	51.09	61.67	1.38	78.23	2.71	85.23	73.26	78.79	62.88	100	77.21
LSTM-VAE	96.24	59.91	73.85	87.79	14.45	24.82	71.64	98.75	75.55	85.99	97.56	85.37
MAD-GAN	98.97	63.74	77.54	41.44	33.92	37.3	81.57	92.16	86.54	85.16	99.3	91.69
OmniAnomaly	72.23	98.32	83.28	26.52	97.99	41.74	75.85	97.56	85.35	91.4	88.91	90.14
USAD	100.0	56	71.79	43.09	22.51	29.57	74.8	96.27	84.19	79.49	99.12	88.22
MTAD-GAT	21.03	64.46	31.71	11.72	30.55	16.94	79.91	99.91	88.8	79.17	98.24	87.68
GDN	99.35	68.12	80.82	97.5	40.19	56.92	74.8	98.91	85.18	93.08	98.92	<u>95.91</u>
GTA	93.9	85.7	<u>89.6</u>	79.6	79.4	<u>79.5</u>	89.11	91.76	<u>90.41</u>	91.04	91.17	91.11
GSLAD	96.77	93.53	**95.13**	78.2	90.2	**83.8**	94.56	100	**97.2**	97.5	100	**98.7**
Percentage Increase	+3.06%	+9.14%	+6.1%	-1.76%	+13.6%	+5.41%	+6.12%	+8.98%	+7.51%	+4.75%	+1.09%	+2.91%

*The highest and second-highest results are highlighted in boldface and underlined, respectively.

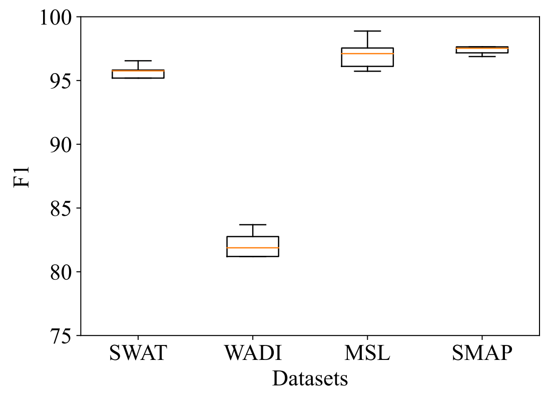

Fig. 5. The F1 scores of GSLAD on four datasets.

training time, it outperforms GDN significantly. Moreover, GTA utilizes the Transformer to make predictions, and its complexity far exceeds that of DCRNN and the static GNN. Thus, GTA's training time is the longest. GSLAD improves the F1 score and greatly reduces the training cost.

As the number of nodes in the graph increases, the training time of the model also increases. When G is sparse, Eq. (5) can be calculated efficiently using $O(L)$ recursive sparse-dense matrix multiplication with total time complexity $O(L|E|) \ll O(K^2)$ [28]. Based on prior knowledge, the WADI dataset with 127 indicators has the largest nodes among all the datasets

TABLE V
TRAINING TIME OF EACH EPOCH(S).

Method	SWAT	WADI	SMAP	MSL
GDN	11.13	42.4	0.67	0.57
GTA	107.38	154.33	8.46	4.17
GSLAD	21.72	65.91	1.2	0.85

in the anomaly detection field. The training time of GSLAD is still less than that of GTA. Notably, data collection and anomaly detection tasks for smart home and IoT systems are typically handled by their gateways or data centers. As a result, it is necessary to employ anomaly detection methods with low computational complexity and training time to reduce the computing capability requirements and deployment costs. Therefore, GSLAD is more suitable for smart home and IoT systems than other methods.

E. RQ2. Ablation Studies

We compare the effects of different anomaly score methods and threshold selection methods on experimental performance to demonstrate the effectiveness of our approach.

1) Anomaly Score: We select three anomaly score methods to compare with that of GSLAD, which is named by standard method.

Me-IQR is similar to the standard method, where the median and inter-quartile range (IQR) are used to normalize

TABLE VI
PRECISION, RECALL, AND F1 SCORE WITH DIFFERENT ANOMALY SCORES AND THRESHOLD SELECTION.

Threshold	Anomaly Score	SWAT			WADI			MSL			SMAP		
		Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
POT	Harmonic-Mean	31.4	17.8	22.8	31.2	85.2	45.7	54.8	79.9	64.8	95	46.6	61.6
	2-Norm	23.4	93	37	9.3	93.2	17	87.3	99.9	**93.3**	88.7	99.9	**94.1**
	Me-IQR	96.9	77.2	86	95.9	16.3	27.9	68.5	99.9	81.3	52.1	99.9	68.5
	Standard	99.07	90	**94.32**	74.76	56.03	**64.06**	70.5	99.9	82.2	95	41.2	57.4
Grid search	Harmonic-Mean	63.3	87.2	73.4	94.2	57.2	71.2	80.5	100	87.8	82.9	100	90.6
	2-Norm	93.5	80.3	86.4	9.65	100	17.6	41.1	100	58.2	60.5	100	75.4
	Me-IQR	90.8	86.3	88.5	66.8	38.7	49	66.6	100	79.9	87.5	100	93.3
	Standard	96.77	93.53	**95.13**	78.2	90.2	**83.8**	97.5	100	**98.7**	94.56	100	**97.2**

the anomaly score instead of the mean and variance, as shown below:

$$\begin{aligned} Err_i(t) &= |x_t^i - \hat{x}_t^i| \\ s_i(t) &= \frac{Err_i(t) - \widetilde{\mu_i}}{\widetilde{\sigma_i}} \\ s(t) &= \max_i s_i(t) \end{aligned} \quad (20)$$

where $\widetilde{\mu_i}$ and $\widetilde{\sigma_i}$ are the median and IQR of the $Err_i(\cdot)$, respectively.

2-Norm compares the ground truth x_t^i and the prediction $\hat{x}_t^i$. The anomaly score is obtained by squaring the second norm of the deviation between the x_t^i and the $\hat{x}_t^i$.

$$s(t) = \sum_{i=1}^{K} |x_t^i - \hat{x}_t^i|_2^2 \quad (21)$$

Harmonic-Mean uses Euclidean distance to measure the difference between the ground truth x_t^i and the prediction $\hat{x}_t^i$.

$$d_t^i = \left| x_t^i - \hat{x}_t^i \right| \quad (22)$$

To avoid anomaly scores in univariate time series being dominated by a single significant spike, it uses the anomaly scores' harmonic mean:

$$s(t) = K / \left(\sum_{i=1}^{K} \frac{1}{d_t^i} \right) \quad (23)$$

The results are summarized in the last four rows of Table VI. The standard anomaly score method outperforms the other three anomaly score methods for all the datasets. The standard method can make the normal and anomaly more obvious.

2) Threshold Selection: We compare the peak over threshold (POT) [29] technique with grid search in the GSLAD. The POT technique is a statistical theory that aims to find the extreme value law, and the extreme value is usually located in the probability distribution's tail. This technique's advantage is that it does not require making assumptions about the data distribution when searching for extreme values.

The results are summarized in Table VI. All the anomaly score methods with POT display poorer performance than those with grid search for the SWAT and WADI datasets. In addition, 2-Norm with POT is better than that with grid search for the MSL and SMAP datasets. Me-IQR with POT is better than with grid search for the MSL dataset. This suggests that using grid search with the standard anomaly score can identify anomaly data more effectively than other methods.

F. RQ3. Impact of Parameters

We consider the impact of the window sizes, loss function weight, and sampling interval.

TABLE VII
TRAINING TIME FOR DIFFERENT WINDOW SIZES.

Window Size	SWAT	WADI	SMAP	MSL
5	14.412	49.791	0.748	0.535
12	21.723	65.91	1.206	0.854
20	28.41	89.219	1.64	1.17
30	38.07	124.3	2.386	1.717
40	52.23	152.91	3.009	2.53

TABLE VIII
THE IMPACT OF THE SAMPLING INTERVAL

Sampling interval (s)	SWAT			WADI		
	Prec	Rec	F1	Prec	Rec	F1
5	97.71	92.52	95.04	81.9	79.44	80.65
10	96.77	93.53	95.13	78.2	90.2	83.8
20	98.85	87.07	92.59	99.83	65.94	79.42

1) Window Size: We analyze the effect of the window size on detection accuracy and training time. We set the windows w to lengths of $5, 12, 20, 30$, and 40. The results are summarized in Fig. 6 (left) and Table VII. The window size has little impact on the model's detection performance. Our method's performance is stable for the SWAT, SMAP, and MSL datasets. The performance on WADI on the WADI dataset slightly reduces with a 5 window size, because WADI has more time series and a lower anomaly rate. Table VII shows that the larger th window size, the longer the training time. In general, when the window size is 12, all the datasets can achieve a high detection accuracy with an adequate training time. Therefore, we set the window size to 12.

2) λ Parameter: In the previous sections, to improve the learned graph's quality, we add some prior knowledge to the graph and use the parameter λ to balance out the prediction and graph learning loss. In this subsection, we use the kNN graph as prior knowledge to show the effect of the parameter λ on the anomaly detection results for the four datasets. We set λ to 0, 1, 5, 10, and 20. The results are shown in Fig. 6 (right). When λ is 1, the F1 scores are the best, indicating that GSLAD has excellent anomaly detection performance when the prediction and graph learning loss are balanced. When λ is set to 0, GSLAD's detection accuracy without the graph learning loss decreases significantly for the MSL and WADI datasets. This indicates that the graph learning loss improves the detection accuracy.

3) Impact of Sampling Interval: Due to a large amount of raw data, the SWAT and WADI datasets are down-sampled. We analyze the impact of different sampling intervals on the

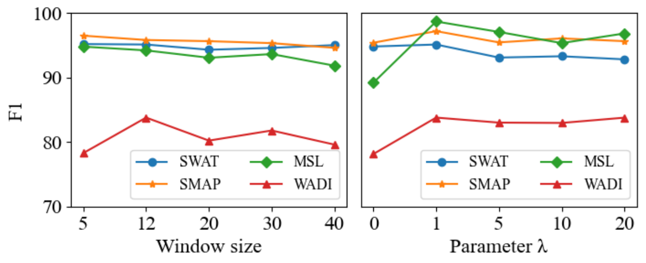

Fig. 6. The impact of parameter λ and window size.

TABLE IX
THE TOTAL NUMBER OF EDGES AND NODE AVERAGE DEGREE IN THE LEARNED GRAPH.

Method	SWAT		WADI		MSL		SMAP	
	Edges	Degrees	Edges	Degrees	Edges	Degrees	Edges	Degrees
GDN	167	3.27	388	3.05	74	2.96	152	2.76
GTA	1559	30.57	7492	58.99	404	16.16	2050	37.27
GSLAD	1757	34.45	11195	88.14	420	16.8	2063	37.51

SWAT and WADI datasets. We down-sample the original data every 5, 10, and 20 seconds. The results are summarized in Table VIII. When the sampling interval is 5 and 10 seconds, the detection performance is similar. When the sampling interval is 20 seconds, the performance decreases. The main reason is that average down-sampling enhances the data's smoothness. Additionally, it smoothes out anomalies, rendering them more challenging to detect.

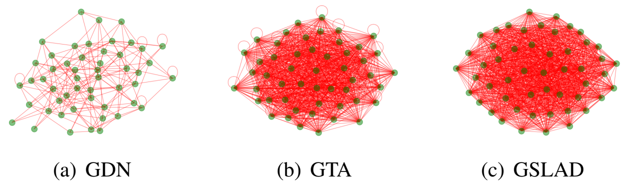

Fig. 7. Graph structure learned by different methods on SWAT dataset.

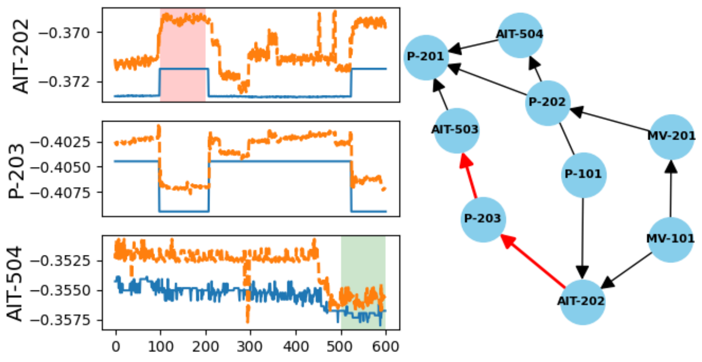

Fig. 8. **Left**: Understand the relationship between sensors with ground truth and prediction, in which abnormal segments are marked by red. **Right**: Partial graph structure learned on SWAT dataset.

G. RQ4: Graph Learning and Case Study

To illustrate GSL's impact, we visualize the learned graph structure of GDN, GTA, and GLSAD. Fig. 7 shows the learned graph structures of the SWAT dataset. Table IX shows the number of edges and the average node degree of the four datasets. Notably, the graph learned from GDN is the sparsest and the number of edges in GDN is the smallest. GDN learns the node embeddings and constructs a kNN graph based on the top-k most similar node. Moreover, the number of neighbors or edges depends on the superparameter k. GDN ignores the relationship among nodes out of top-k. In contrast, GTA and GLSAD learn dense graphs because they regard the adjacency matrix as independent variables without the number of neighbors limitation. GLSAD learned more edges without self-loop edges and has a higher average node degree than GTA. This is because the downstream prediction task model in GLSAD is a recurrent graph convolution neural network that accounts for the dynamic temporal and spatial features. Since the nodes in the graph represent indicators or features, the relationship among nodes is also the relationship among the indicators or features.

We conduct a case study, as shown in Fig. 8. Fig. 8 (left) is a partial graph structure learned by GSLAD, and Fig. 8 (right) shows our model's predictions for the relevant sensors. In this case, sensor AIT-202 is compromised between the 100 and 210 timestamps. Hence, GSLAD detected an attack due to the large difference between the predictions and ground truth on AIT-202 during these two timestamps. Due to the correlation between the sensors in the water treatment process, the attack on AIT-202 causes the dosing pump P-203 to shut down and affects the permeate conductivity analyzer AIT-504. Furthermore, GSLAD accurately predicts the changes changes occured in P-203 from the 100 to 210 timestamps and AIT-504 from the 500 timestamps. The values of P-203 and AIT-504 follow the changes in the ground truth and are not regarded as anomalies, as shown in Fig. 8 (right). This is because GSLAD learns the correlation among the three sensors correctly, as shown in Fig. 8 (left).

H. RQ5: Anomaly Detection in Monitoring Data of CEP

To evaluate GSLAD's efficiency in monitoring data of CEP , we conduct an experiment on the Identifying Age-Related Diseases (ICR)[4] dataset. ICR, which originates from the Kaggle competition, is a dataset that encompasses health-related information about individual patients or subjects. Generally, health-related information can be collected by wearable consumer electronics such as electronic bracelets and electronic watches. ICR consists of 56 anonymized health features or variables for each subject, along with a labeled list indicating whether or not the subject has been diagnosed with certain diseases. Due to the small size of this dataset, we employ the TimeGAN [30] model with default parameters for data generation. In total, we generate 10000 instances of data with an anomaly rate of 4.2%. Previous experiments have demonstrated the superiority of GSL-based methods over other approaches. Hence, in our evaluation, we solely compare GSLAD with various GSL-based methods, using the same metrics. As shown in Table

[4]https://www.kaggle.com/competitions/icr-identify-age-related-conditions/data

5, GSALD outperforms other methods in terms of F1 score while requiring less training time.

TABLE X
PRECISION, RECALL, F1 SCORE, AND TRAINING TIME ON ICR.

Method	Prec	Rec	F1	Training time(s)
GDN	75.31	96.13	84.42	2.13
GTA	91.23	90.76	90.94	16.36
GSLAD	94.91	100	**97.39**	3.87

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose a multivariate time series anomaly detection framework using lightweight and full graph structure learning. In this framework, we implement a simpler and more effective graph structure learning approach and prediction model by a recurrent graph convolution. Compared with the baseline methods for four public real-world datasets, our proposed GLSAD achieves the best performance with short-term data while reducing the training overhead. In the next generation of consumer electronics products, such as autonomous driving or wearable devices, sensors exhibit various interrelationships (e.g., sensor's distance and data similarity relationship) and the data contains noise. To take full advantage of various interrelationships, we can try to exploit multiple graph structure learners to learn multiple graph structures to characterize the relationships between sensors, obtain the hidden information from different graph structures by multiple GNNs, and finally fuse the hidden information by concatenation or on average for prediction. Besides, our method does not consider the presence of noisy data. To improve the robustness of the model to noise, we can use the autoencoder to extract the hidden features and detect anomalies on the hidden features instead of the original data.

REFERENCES

[1] M. Yamauchi, Y. Ohsita, M. Murata, K. Ueda, and Y. Kato, "Anomaly detection in smart home operation from user behaviors and home conditions," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 2, pp. 183–192, 2020.

[2] J.-Y. Son, J.-H. Park, K.-D. Moon, and Y.-H. Lee, "Resource-aware smart home management system by constructing resource relation graph," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1112–1119, 2011.

[3] G. Fragkos, C. Minwalla, J. Plusquellic, and E. E. Tsiropoulou, "Artificially intelligent electronic money," *IEEE Consumer Electronics Magazine*, vol. 10, no. 4, pp. 81–89, 2021.

[4] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and 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.

[5] S. He, M. Guo, Z. Li, Y. Lei, S. Zhou, K. Xie, and N. N. Xiong, "A joint matrix factorization and clustering scheme for irregular time series data," *Information Sciences*, vol. 644, p. 119220, 2023.

[6] S. He, Z. Li, J. Wang, and N. Xiong, "Intelligent detection for key performance indicators in industrial-based cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5799 – 5809, 2021.

[7] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and 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.

[8] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[9] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International Conference on Learning Representations*, 2018.

[10] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and 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.

[11] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and 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.

[12] Q. He, Y. Zheng, C. Zhang, and H. Wang, "Mtad-tf: Multivariate time series anomaly detection using the combination of temporal pattern and feature pattern," *Complexity*, vol. 2020, p. 8846608, 2020.

[13] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, "Multivariate time-series anomaly detection via graph attention network," in *2020 IEEE International Conference on Data Mining (ICDM)*, 2020, pp. 841–850.

[14] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.

[15] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning graph structures with transformer for multivariate time-series anomaly detection in iot," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9179–9189, 2022.

[16] J. Pei, Z. Yu, J. Li, M. A. Jan, and K. Lakshmanna, "Tkagfl: a federated communication framework under data heterogeneity," *IEEE Transactions on Network Science and Engineering*, 2022.

[17] D. Scheinert, A. Acker, L. Thamsen, M. K. Geldenhuys, and O. Kao, "Learning dependencies in distributed cloud applications to identify and localize anomalies," in *2021 IEEE/ACM International Workshop on Cloud Intelligence (CloudIntelligence)*, 2021, pp. 7–12.

[18] Y. Zhu, W. Xu, J. Zhang, Y. Du, J. Zhang, Q. Liu, C. Yang, and S. Wu, "A survey on graph structure learning: Progress and opportunities," 2021.

[19] R. Levie, M. M. Bronstein, and G. Kutyniok, "Transferability of spectral graph convolutional neural networks," *J. Mach. Learn. Res.*, vol. 22, pp. 272:1–272:59, 2021.

[20] D. Yu, R. Zhang, Z. Jiang, Y. Wu, and Y. Yang, "Graph-revised convolutional network," in *ECML/PKDD*, 2020.

[21] D. D. S. Pilco and A. R. Rivera, "Graph learning network: A structure learning algorithm," *ArXiv*, vol. abs/1905.12665, 2019.

[22] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang, "Learning to drop: Robust graph neural network via topological denoising," *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021.

[23] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *ICML*, 2019.

[24] X. Gao, W. Hu, and Z. Guo, "Exploring structure-adaptive graph learning for robust semi-supervised classification," *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2020.

[25] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[26] E. Jang, S. S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *ArXiv*, vol. abs/1611.01144, 2017.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[28] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=SJiHXGWAZ

[29] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouët, "Anomaly detection in streams with extreme value theory," *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[30] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *Advances in neural information processing systems*, vol. 32, 2019.