Information Sciences 728 (2026) 122790

Contents lists available at ScienceDirect

Information Sciences

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

# GDCMAD: Graph-based dual-contrastive representation learning for multivariate time series anomaly detection

Sheng He[1], Wenxuan He[1], Mingjing Du*, Xiang Jiang, Yongquan Dong*

*Jiangsu Key Laboratory of Educational Intelligent Technology, School of Computer Science and Technology, Jiangsu Normal University, Xuzhou, 221116, China*

ARTICLE INFO

*Keywords:*
Multivariate time series
Anomaly detection
Contrastive learning
Graph neural network

ABSTRACT

The increasing amount of multivariate time series (MTS), coupled with scarce labeled samples, has driven the development of unsupervised anomaly detection. While contrastive learning has shown promise in learning discriminative representations, existing contrastive learning-based MTS anomaly detection methods still suffer from limited representation power and inadequate discrimination ability. In this paper, we propose a novel model, graph-based dual-contrastive representation learning for detecting anomalies in multivariate time series, called GDCMAD. GDCMAD first constructs two relational graphs for capturing inter-variable and temporal dependencies then integrates an improved Kolmogorov–Arnold network (KAN)-based attention mechanism into a reconstruction framework. Additionally, it incorporates an LSTM-based external contrastive learning module to further enhance the separation between normal and abnormal patterns. Experiments on six public datasets show that GDCMAD achieves better performance than nine state-of-the-art methods in detecting anomalies, confirming its effectiveness for MTS data. To access the source code of GDCMAD, please visit the repository located at https://github.com/Du-Team/GDCMAD.

## 1. Introduction

Anomaly detection plays an essential role in identifying outliers that deviate significantly from the majority of data [1]. This task can involve a variety of data types, including images, videos, and time series. Time series data, in particular, receives significant attention because of its inherent temporal nature [2]. A time series comprises data points ordered chronologically [3]. Often, these data points originate from multiple sensors, resulting in complex multivariate time series (MTS) [4]. Compared to univariate time series, MTS data is more intricate and diverse, rendering anomaly detection more complex. Anomalies in MTS include not only temporal anomalies (patterns within a single series) but also inter-variable anomalies (unusual interactions between variables), as illustrated in Fig. 1. Traditional anomaly detection methods often focus on temporal dependencies while neglecting inter-variable relationships. Furthermore, labeling multiple types of anomalies in multivariate data requires higher human labor costs, creating a significant barrier for supervised and semi-supervised learning methods that depend on large labeled datasets.

Recently, unsupervised learning has become a mainstream approach for MTS anomaly detection. With the rapid advancement of deep learning, research has shifted toward techniques that can automatically learn and capture deep features from data. Among these, reconstruction-based techniques have been widely adopted. The fundamental concept involves using models to identify the

* Corresponding authors.
*Email addresses:* hes@jsnu.edu.cn (S. He), hewx@jsnu.edu.cn (W. He), dumj@jsnu.edu.cn (M. Du), xjiang@jsnu.edu.cn (X. Jiang), tomdyq@163.com (Y. Dong).
[1] Equal contribution.

https://doi.org/10.1016/j.ins.2025.122790
Received 1 February 2025; Received in revised form 11 October 2025; Accepted 13 October 2025
Available online 20 October 2025

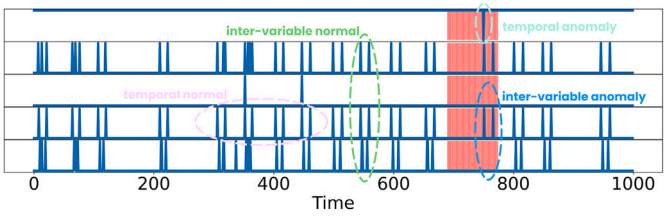


**Fig. 1.** Diagram of two kinds of MTS anomalies. Temporal anomalies: points deviate from their expected temporal dependencies, such as the curve sharply declining instead of following a smooth trend, highlighted by the green circles within the red-shaded area on the right. Inter-variable anomalies: the expected relationships between variables are disrupted, such as the abnormal change in the fifth dimension deviating from its typical parallel movement with the fourth dimension, highlighted by blue circles in the red-shaded area on the right.

distribution patterns of representations within normal time series data. Meanwhile, contrastive learning has demonstrated its effectiveness in enhancing representation learning across various domains. As a result, recent research increasingly focuses on combining reconstruction-based approaches with contrastive learning to more effectively capture the intricate relationships within MTS data [5,6].

Despite the significant progress of the reconstruction methods based on contrastive learning, existing methods still face two major challenges: limited data representation and insufficient discriminative ability. Regarding representation capabilities, many methods primarily focus on extracting local temporal and inter-variable dependencies, neglecting the comprehensive capture of global features. Moreover, while Transformer-based models have shown effectiveness across various domains, their attention mechanisms are underexplored, restricting the models' representational ability in MTS data. In terms of discriminative ability, existing contrastive learning-based methods often fail to fully leverage the learned representations due to inadequate alignment between the raw data and the learned features. This misalignment hampers the ability to capture meaningful distinctions, resulting in insufficient differentiation between normal and abnormal patterns and limiting the model's discriminative power.

A graph-based dual-contrastive representation learning method for MTS anomaly detection, called GDCMAD, is proposed to address these issues. In detail, we embed a graph-based contrastive learning module within the reconstruction framework and subsequently add an LSTM-based external contrastive learning module, establishing a dual mechanism that improves both representation learning and discrimination capabilities. More specifically, we introduce a graph contrastive learning module into the autoencoder architecture. This module transforms multivariate time series into two graph structures from two distinct perspectives (temporal dependency and variable interaction), employs two graph neural networks to extract features from these perspectives, and uses the different features extracted from the same sample as positive pairs in contrastive learning to learn richer representations. Additionally, we design an attention mechanism based on an improved Kolmogorov-Arnold network (KAN) [7] and incorporate it into the autoencoder structure to improve the model's representation ability for time series. Finally, we design an LSTM-based external contrastive learning module, where raw and reconstructed data are paired and fed into the module. This brings normal data closer while pushing anomalous data farther apart, amplifying the distinction between the two and improving anomaly detection accuracy.

This paper is contributed as follows:

- We propose GDCMAD, an innovative unsupervised anomaly detection framework in MTS, integrating an improved autoencoder architecture and an external contrastive learning module, to enhance anomaly detection performance.
- We develop an autoencoder architecture that introduces a dual-branch graph contrastive learning module and an improved KAN-based attention mechanism to effectively capture complex relationships and features in MTS.
- We design an external contrastive learning module that leverages pairs of raw data and their reconstructed counterparts to optimize the separation between normal and anomalous patterns, thereby enhancing the model's discriminative power.

## 2. Related work

In this section, two key topics relevant to this paper are briefly presented: deep MTS anomaly detection, and contrastive learning.

### *2.1. Deep multivariate time series anomaly detection*

There is a surge of research interest in deep MTS anomaly detection due to the excellent performance of deep learning methods for data representation. Among them, reconstruction-based methods are more widely adopted. A basic idea behind these techniques involves training a model to understand the distribution within normal time series data, which is then used to reconstruct incoming data. Anomalies are identified by measuring the reconstruction error, which indicates significant deviations between the reconstructed data and the raw data. An encoder-decoder structure is often used for such methods. OmniAnomaly [8] utilizes random variable concatenation and other strategies to uncover the latent data distribution. However, it primarily emphasizes the temporal relationships in time series, with less focus on the interactions across feature dimensions. To address this issue, MSCRED [9] assesses the condition of multi-layered architectures across diverse timescales through the innovative development of a multi-scale signature matrix. In addition to this, it effectively utilizes convolutional encoders to encode feature relationships simultaneously and a custom-designed network to detect temporal patterns. In contrast, MEGA [10] incorporates the discrete wavelet transform into an autoencoder framework, which

is used to break down data into separate frequency components before reconstructing them. However, due to the use of fixed wavelet basis functions, the method tends to focus more on localized temporal features, which may limit its ability to effectively capture global characteristics of time series. MST–GAT [5] leverages a spatiotemporal convolutional network and a multimodal graph attention network, offering an alternative approach. The combination of the two aims to capture the spatio-temporal dependencies in MTS. Yet, its reliance on a single reconstruction error metric hampers its ability to differentiate between normal and anomalous patterns with similar error profiles. MGDRF [11] employs a multi-grain mask approach for extracting semantic features across different levels in MTS. Moreover, it effectively extracts different semantic features at various levels by combining a hierarchy of loss functions that leverage both receptive field principles and model-based approaches. Drawing on ensemble learning, MGDRF further integrates the results from different granularity dynamic receptive fields to delve into the interactive features between different granularity MTS. However, ensemble learning may inadvertently reduce discriminative power due to feature redundancy or conflicts.

### 2.2. Contrastive learning

Contrastive learning is a self-supervised learning method that improves performance by making the representation consistent through appropriate data transformations. The InfoNCE [12] method achieves contrastive learning by bringing the positive samples closer together and pulling the negative samples away from each other in the same projection space. However, the introduction of negative samples may introduce uncertainty and lead to performance degradation [13]. To solve this problem, BYOL [14] proposes a novel contrastive learning framework. The framework is characterized by the fact that it does not use negative samples and can bring similar features automatically closer together in the projection space while keeping non-similar features automatically away. Graph contrastive learning demonstrates significant potential for non-Euclidean data representation. Chen et al. [15] propose a predictive task to augment graph self-supervised contrastive learning. Hassani et al. [16] learn node/graph representations by contrasting structural views. Zhu et al. [17] design topology- and semantics-enhanced strategies to optimize representation learning.

However, none of the above methods focuses on time-series studies. MGCLAD [6] proposes a reconstruction-based multiview contrastive learning framework for modeling temporal context and capturing dependencies between signals. COCA [13], on the other hand, employs a contrastive learning strategy designed to address with the normality assumption and one-class classification problems. This framework treats the raw data representations and their reconstructed representations as positive pairs without negative samples. Furthermore, COCA tailors a contrastive loss function for one-class classification by integrating positive example pairs. Although specifically designed for time series, it primarily emphasizes temporal dependency modeling and lacks an explicit mechanism for capturing cross-variable interactions. The recently proposed DCdetector [18] learns permutation-invariant representations through a dual-branch attention structure (patch-wise and in-patch attention networks), combined with channel-independent patch methods to enhance local semantic features. Its multi-scale attention design effectively alleviates information loss during time series patching operations, while instance normalization strategies reduce model complexity and overfitting risks, providing new insights for contrastive learning applications in temporal anomaly detection.

This paper is different from existing research in several ways. Firstly, we develop an attentional mechanism incorporating improved KAN in the contrastive learning module of the autoencoder, which helps our method capture data features more comprehensively. Secondly, we add an external contrastive learning module, which not only enhances the differential association between the raw and reconstructed data but also effectively reduces the additional uncertainty that may arise from contrastive learning based on data augmentation.

## 3. Preparatory work

Before detailing the model, we present the notations and explain the fundamentals of two key components: graph neural networks and Kolmogorov-Arnold networks.

### 3.1. Notation descriptions

Define the multivariate time series as a matrix $\boldsymbol{X} \in \mathbb{R}^{T\times M}$, where each row $\boldsymbol{X}_{t\cdot} = [x_{t,1}, x_{t,2}, \dots, x_{t,M}] \in \mathbb{R}^M$ represents a data point at time $t$, denoted as $\boldsymbol{x}_t$. Each column $\boldsymbol{X}_{\cdot i} = [x_{1,i}, x_{2,i}, \dots, x_{T,i}]^\top \in \mathbb{R}^T$ represents the $i$-th univariate time series across all time steps, denoted as $\boldsymbol{f}_i$. Here, $T$ denotes the timestamp length, and $M$ denotes the dimensional size of each point [19,20].

A test time series is utilized with the length of $\hat{T}$. The complete testing results for each point in the dataset are expressed as $\boldsymbol{y} = [y_1, y_2, \dots, y_{\hat{T}}]^\top$. The testing result at a specific time $t$, denoted as $y_t$, takes a value in $\{0, 1\}$, where 1 indicates an anomaly.

Table 1 provides a detailed overview of the notations and corresponding descriptions employed throughout this paper.

### 3.2. Graph neural network

In recent years, graph structure learning has received more and more attention because of its great benefit to the learning of dependencies between variables. Graph neural network (GNN) generates an embedded representation of a node by using node information in a graph. Its core process can be summarized in three steps: aggregation, update, and loop. The aggregation step evaluates the characteristics of a node by integrating information from neighboring nodes. The update step, on the other hand, fuses the aggregated information obtained in the aggregation step with its node information to generate the updated node features. The loop step then implies that the above two steps are repeated until each node in the graph can effectively establish connections with all other nodes.

**Table 1**
Summary of main notations.

| Notations | Descriptions |
|---|---|
| $T$ | The timestamp length |
| $\boldsymbol{x}_t$ | The data point at time step $t$ |
| $\boldsymbol{f}_i$ | The vector of the $i_{th}$ dimension of the input data |
| $M$ | The dimension sizes for data points |
| $K$ | The context window length |
| $\boldsymbol{\epsilon}$ | The minor constancy vector |
| $y_t$ | The test result of a data point at time step $t$ |
| $\boldsymbol{y}$ | The test result of each data point of the test data |
| $\boldsymbol{S}_t$ | The anomaly scores of $x_t$ |
| $\boldsymbol{W}'_t$ | The reconstructed window data at time step $t$ |
| $\boldsymbol{Z}$ | The latent variable |
| $\boldsymbol{W}_{hor}$, $\boldsymbol{W}_{ver}$ | Two input transformations for the contrastive learning module |
| $\boldsymbol{z}_{hor}$, $\boldsymbol{z}_{ver}$, $\boldsymbol{z}_u$, $\boldsymbol{z}_l$ | The projection feature vectors generated by the contrastive learning module |
| $e$ | The training epoch |

Specifically, we define a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{F})$, where $\boldsymbol{V}$ denotes the set of nodes, $\boldsymbol{E}$ represents the set of edges, and $\boldsymbol{F}$ represents the set of features of the nodes. Suppose the vector of node $i$ in layer $l$ is represented by $\boldsymbol{h}_i^l$. The aggregation and update process can be represented as follows:

$$\boldsymbol{a}_{\boldsymbol{N}_i}^l = Agg(\boldsymbol{h}_j^{l-1}), j \in \boldsymbol{N}_i \tag{1}$$

$$\boldsymbol{h}_i^l = Update(\boldsymbol{h}_i^{l-1}, \boldsymbol{a}_{\boldsymbol{N}_i}^l), \tag{2}$$

here, $\boldsymbol{N}_i$ represents the neighboring nodes of node $i$ in the graph. $Agg(\cdot)$ refers to the aggregation step, and $Update(\cdot)$ refers to the update step.

### 3.3. Kolmogorov-arnold network

KAN, as a novel network structure, replaces the traditional multilayer perceptron (MLP) network. Unlike the process of MLP, which first performs a linear transformation and then extracts nonlinear features through activation functions, KAN extracts nonlinear features directly from the data. The main characteristic of KAN is that it places its activation function at the network edges, and these activation functions are learnable and are often parameterized using B-splines. This design allows KAN to exhibit higher accuracy and interpretability when dealing with problems such as complex function fitting and partial differential equation solving. The specific implementation is given as follows:

$$KAN(\boldsymbol{x}) = (\Phi_L \circ \cdots \circ \Phi_l \circ \cdots \circ \Phi_1)\boldsymbol{x}. \tag{3}$$

where $\Phi_l$ is denoted as layer $l$ of KAN. $\circ$ is represented as a composite mapping of $\Phi_l$ and $\Phi_{l-1}$, where $2 \le l \le L$.

## 4. Method

Fig. 2 illustrates that GDCMAD is composed of two parts: an autoencoder with a contrastive learning module and improved attentional mechanisms, and an external contrastive learning module involving the raw and reconstructed data. Firstly, the data is normalized and subsequently segmented to generate window data, i.e., raw data. These raw data, after being transformed, are input into the two branches of the encoder's contrastive learning module. These branches are designed to model temporal dependencies and inter-variable relationships, respectively, and subsequently extract the corresponding projection features. Next, these features are fused and concatenated to form latent variables, followed by a decoder generating reconstructed data. In the final step, raw data is fed together with the reconstructed data to the external contrastive learning module, which contains two independent LSTMs.

### 4.1. Preprocessing

Before feeding the data into the model, the data is first normalized and subsequently, the corresponding window data is extracted. We will normalize time series $\boldsymbol{X}$ as follows:

$$\boldsymbol{x}_t \leftarrow \frac{\boldsymbol{x}_t - \min(\boldsymbol{X})}{max(\boldsymbol{X}) - \min(\boldsymbol{X}) + \boldsymbol{\epsilon}}, \tag{4}$$

where $\boldsymbol{x}_t$ is a data point, $\min(\boldsymbol{X})$ denotes the minimum vector of time series, and $\max(\boldsymbol{X})$ is represented as the maximum vector. $\boldsymbol{\epsilon}$ is a small constant vector that prevents any single value in the sequence from making the denominator of the above equation zero, ensuring the equation remains calculable.

Then, for each data point $\boldsymbol{x}_t$ at time $t$, a context window of length $K$ is constructed. The context window starting at time $t-K+1$ until $t$ is defined as $\boldsymbol{W}_t \in \mathbb{R}^{K \times M}$. We extract the data for each timestamp within this window and construct the set $\{\boldsymbol{X}_{(t-K+1)\cdot}, \ldots, \boldsymbol{X}_{t\cdot}\}$, which captures temporal dependencies in MTS. Simultaneously, we extract the data for each variable across the context window and

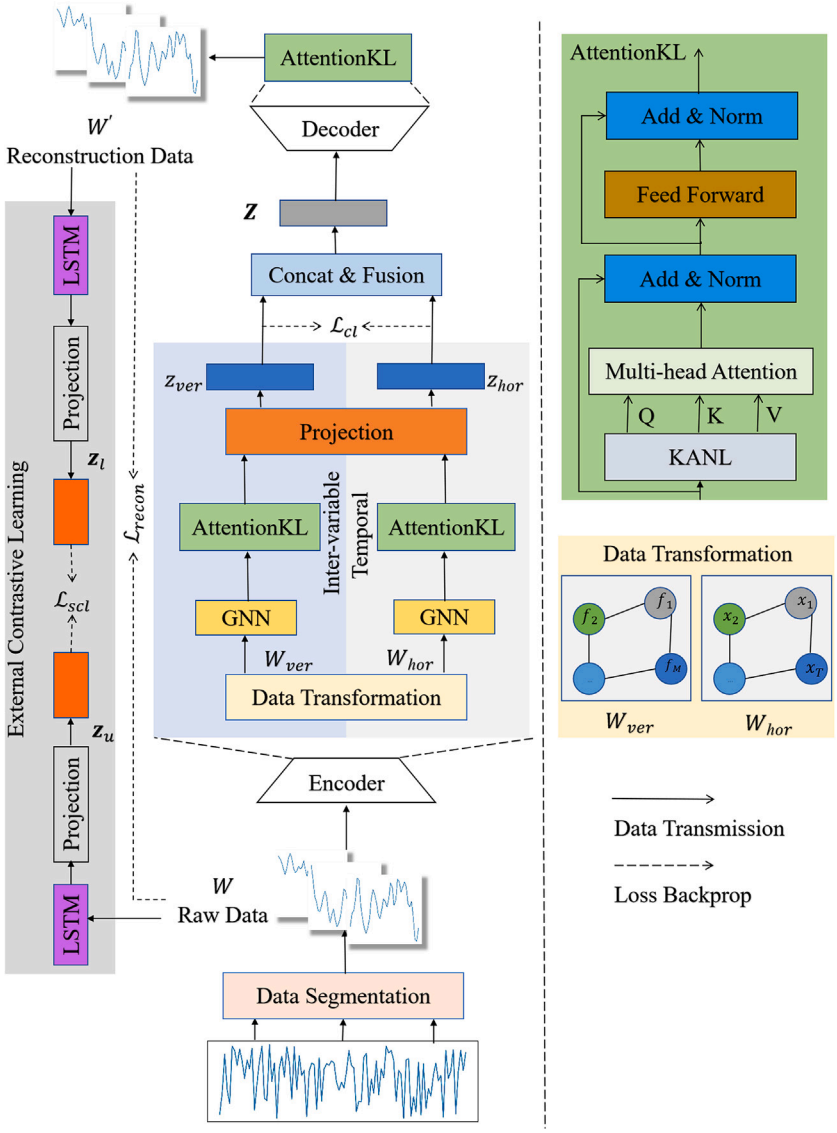


**Fig. 2.** The GDCMAD model.

form the set $\{\hat{\boldsymbol{X}}_{\cdot,1}, \ldots, \hat{\boldsymbol{X}}_{\cdot,M}\}$, where $\hat{\boldsymbol{X}}_{\cdot,i} = [x_{(t-K+1),i}, \ldots, x_{t,i}]^\top$, capturing inter-variable dependencies in the MTS. The anomaly score $\boldsymbol{S}_t$ of $\boldsymbol{x}_t$ is calculated based on $\boldsymbol{W}_t$, subsequently utilized to ascertain the anomalous status of $\boldsymbol{x}_t$ [21].

### 4.2. Autoencoder

To construct more reasonable positive pairs for contrastive learning on multivariate time series, the data is transformed into two graph structures from distinct perspectives: temporal dependency and variable interaction. These two graph structures are fed into two separate branches of the encoder's contrastive learning module—one branch utilizes the graph constructed across time steps to model temporal dependencies, while the other leverages the graph built across variables to capture inter-variable relationships. The features extracted from both branches for the same instance are then treated as a positive pair for contrastive learning. Subsequently, these graph representations are turned into corresponding projection features. These features are subsequently concatenated and fused to generate latent variables, which are passed to the decoder. The decoder incorporates an attentional mechanism with an improved KAN (also applied in the encoder) to generate reconstructed data.

Different from some models that adopt data augmentation-based contrastive learning strategies in autoencoders—which often risk disrupting intrinsic temporal features and assume that augmentations introduce no uncertainty—a graph-based contrastive learning approach is used. An improved KAN is introduced into the attention mechanism. While the KAN exhibits strong representation

learning capabilities, its performance on time-series data remains limited. To address this issue, the B-spline curve in KAN is replaced with a LeakyReLU activation function.

**Encoder.** The encoder structure, centered on the graph-based contrastive learning module, leverages dual branches to capture temporal and inter-variable dependencies in data.

Input data $\boldsymbol{W}_t$ is first transformed into two distinct forms: $\boldsymbol{W}_{hor}$, representing the cross-time-step representation, and $\boldsymbol{W}_{ver}$, representing the cross-variable representation. These two data parts are processed by separate GNNs to capture temporal dependencies and inter-variable relationships. An attention mechanism based on the improved KAN is then applied to capture global information, producing the projection features $\boldsymbol{z}_{hor}$ and $\boldsymbol{z}_{ver}$. These features are used to calculate the contrastive learning loss, which drives the model's training process.

The process of acquiring $\boldsymbol{z}_{hor}$ is expressed in the following equation:

$$\boldsymbol{Q}_{hor} = \boldsymbol{K}_{hor} = KAN_1(GNN_1(\boldsymbol{W}_{hor})) \tag{5}$$

$$\boldsymbol{V}_{hor} = KAN_2(GNN_1(\boldsymbol{W}_{hor})) \tag{6}$$

$$\boldsymbol{H}_{hor} = LN(GNN_1(\boldsymbol{W}_{hor}) + Attention_1(\boldsymbol{Q}_{hor}, \boldsymbol{K}_{hor}, \boldsymbol{V}_{hor})) \tag{7}$$

$$\boldsymbol{z}_{hor} = Projection_1(LN(FF_1(\boldsymbol{H}_{hor}) + \boldsymbol{H}_{hor})) \tag{8}$$

where $\boldsymbol{Q}_{hor}$, $\boldsymbol{K}_{hor}$ and $\boldsymbol{V}_{hor}$ are considered as the query, key, and value. $GNN_1(\cdot)$ is responsible for graph structure learning operation. $FF_1(\cdot)$ is a fully connected network. $KAN_1(\cdot)$ and $KAN_2(\cdot)$ denote the improved KANs. $LN(\cdot)$ represents a layer normalization operation. $Attention_1(\cdot,\cdot,\cdot)$ refers to the attention mechanism. $Projection_1(\cdot)$ represents denoted as projection operation.

Similarly, the acquisition process of $\boldsymbol{z}_{ver}$ can be shown in the following equation:

$$\boldsymbol{Q}_{ver} = \boldsymbol{K}_{ver} = KAN_3(GNN_2(\boldsymbol{W}_{ver})) \tag{9}$$

$$\boldsymbol{V}_{ver} = KAN_4(GNN_2(\boldsymbol{W}_{ver})) \tag{10}$$

$$\boldsymbol{H}_{ver} = LN(GNN_2(\boldsymbol{W}_{ver}) + Attention_2(\boldsymbol{Q}_{ver}, \boldsymbol{K}_{ver}, \boldsymbol{V}_{ver})) \tag{11}$$

$$\boldsymbol{z}_{ver} = Projection_1(LN(FF_2(\boldsymbol{H}_{ver}) + \boldsymbol{H}_{ver})) \tag{12}$$

where $\boldsymbol{Q}_{ver}$, $\boldsymbol{K}_{ver}$ and $\boldsymbol{V}_{ver}$ are considered as the query, key, and value. $GNN_2(\cdot)$ is responsible for graph structure learning operation. $FF_2(\cdot)$ is a fully connected network. $KAN_3(\cdot)$ and $KAN_4(\cdot)$ denote the improved KANs. $Attention_2(\cdot,\cdot,\cdot)$ represents the attention mechanism.

Subsequently, we use $\boldsymbol{z}_{hor}$ and $\boldsymbol{z}_{ver}$ to calculate the loss of contrastive learning. The approach presented in this paper differs from traditional contrastive learning strategies, which rely on pairs of positive and negative samples. We mainly refer to BYOL [14], which enables efficient learning without the need for large batch sizes by reducing the dependence on massive negative samples. Specific loss function formulation reads as follows:

$$\mathcal{L}_{cl} = \frac{1}{N}\sum_{i=1}^{N} -\frac{\boldsymbol{z}_{ver}^{(i)}}{\|\boldsymbol{z}_{ver}^{(i)}\|_2}\frac{\boldsymbol{z}_{hor}^{(i)}}{\|\boldsymbol{z}_{hor}^{(i)}\|_2}. \tag{13}$$

where $N$ denotes the sample number.

**Fusion.** For successful completion of the subsequent reconstruction task, we need to fuse the projection features $\boldsymbol{z}_{hor}$ and $\boldsymbol{z}_{ver}$ to generate the latent variable $\boldsymbol{Z}$. This is done as shown below:

$$\boldsymbol{Z} = Liner(Concat(\boldsymbol{z}_{ver}, \boldsymbol{z}_{hor})), \tag{14}$$

where $Liner(\cdot)$ is the fully connected layer. $Concat(\cdot)$ is a concatenation operation.

**Decoder.** After obtaining the latent variable $\boldsymbol{Z}$, we perform the reconstruction task in the decoder. The decoder contains a module with an attention mechanism designed to convert the latent variable $\boldsymbol{Z}$ into the reconstructed data $\boldsymbol{W}_t'$. The steps are as follows:

$$\boldsymbol{Q}' = \boldsymbol{K}' = KAN_5(\boldsymbol{Z}) \tag{15}$$

$$\boldsymbol{V}' = KAN_6(\boldsymbol{Z}) \tag{16}$$

$$\boldsymbol{H}' = LN(\boldsymbol{Z} + Attention_3(\boldsymbol{Q}', \boldsymbol{K}', \boldsymbol{V}')) \tag{17}$$

$$\boldsymbol{W}_t' = LN(FF_3(\boldsymbol{H}') + \boldsymbol{H}') \tag{18}$$

where $\boldsymbol{Q}'$, $\boldsymbol{K}'$, and $\boldsymbol{V}'$ are considered as the query, key, and value, respectively. $FF_3(\cdot)$ is a fully connected network. $KAN_5(\cdot)$ and $KAN_6(\cdot)$ denote the improved KANs. $Attention_3(\cdot,\cdot,\cdot)$ represents the attention mechanism.

Then, we can obtain the reconstruction loss corresponding to $\boldsymbol{W}_t$ and $\boldsymbol{W}_t'$:

$$\mathcal{L}_{recon} = \left\|\boldsymbol{W}_t - \boldsymbol{W}_t'\right\|_F^2. \tag{19}$$

### *4.3. External contrastive learning*

To enhance the accuracy, we introduce an LSTM-based external contrastive learning module. The core idea is to pair the raw and reconstructed data and feed them into the module. This strategy brings normal data points closer while pushing anomalous ones further apart, thereby amplifying the distinction between normal and abnormal data. The outcome is a more effective separation, which directly improves the model's ability to detect anomalies. Notably, the module can be applied to some data augmentation-based

methods, mitigating the negative effects of data augmentation, such as the disruption of inherent temporal characteristics and the introduction of uncertainty. We also validate this in the subsequent experiments.

This module consists of two different LSTMs in two branches. The specific implementation is as follows:

$$\boldsymbol{z}_u = Projection_2(LSTM_1(\boldsymbol{W}_t)) \tag{20}$$

$$\boldsymbol{z}_l = Projection_2(LSTM_2(\boldsymbol{W}'_t)), \tag{21}$$

where $\boldsymbol{z}_u$ and $\boldsymbol{z}_l$ are referred to as projection features. $Projection_2(\cdot)$ is referred to as projection operation.

Then, we use both to calculate the corresponding contrastive learning loss:

$$\mathcal{L}_{scl} = \frac{1}{N}\sum_{i=1}^{N} -\frac{\boldsymbol{z}_u^{(i)}}{\|\boldsymbol{z}_u^{(i)}\|_2}\frac{\boldsymbol{z}_l^{(i)}}{\|\boldsymbol{z}_l^{(i)}\|_2}. \tag{22}$$

### 4.4. Training objective and anomaly detection

#### 4.4.1. Training objective

After introducing the contrastive learning loss in the encoder, the reconstruction loss, and the external contrastive learning loss, we merge these three losses into a final loss function that is used to guide the training process. Specifically:

$$\mathcal{L} = \mathcal{L}_{recon} + \lambda_1\mathcal{L}_{cl} + \lambda_2\mathcal{L}_{scl}, \tag{23}$$

where $\lambda_1$ and $\lambda_2$ are used as the scaling factors.

Algorithm 1 shows the whole training process.

#### 4.4.2. Anomaly detection

After training the model, the model is employed to detect anomalies in the test data, as described in Algorithm 2. Anomaly scores are computed as follows:

$$\boldsymbol{S}_t = \left\|\hat{\boldsymbol{W}}_t - \hat{\boldsymbol{W}}'_t\right\|_F^2 \tag{24}$$

where $\hat{\boldsymbol{W}}_t$ signifies unobserved window data. If their scores surpass the predefined threshold $\lambda$, points are deemed abnormal, if not, they are regarded as normal [22]. As indicated in the following:

$$s_{an} = \begin{cases} abnormal & \boldsymbol{S}_t \geq \lambda \\ normal & \boldsymbol{S}_t < \lambda \end{cases}, \tag{25}$$

$$y_t = \begin{cases} 1 & \boldsymbol{S}_t \geq \lambda \\ 0 & \boldsymbol{S}_t < \lambda \end{cases}, \tag{26}$$

where $s_{an}$ represents the anomaly state at the timestamp $t$.

**Algorithm 1** Training algorithm.

**Input:** Normal window Dataset $\mathcal{W} = \{\boldsymbol{W}_1, ..., \boldsymbol{W}_T\}$, number epochs $N_{epoch}$
**Output:** Trained GDCMAD

$e \leftarrow 1$
**while** $e \leq N_{epoch}$ **do**
  **for** $t = 1$ *to* $T$ **do**
    $\boldsymbol{W}_{hor}, \boldsymbol{W}_{ver} \leftarrow \boldsymbol{W}_t$
    $\boldsymbol{H}_{hor} \leftarrow LN(GNN_1(\boldsymbol{W}_{hor}) + Attention_1(\boldsymbol{Q}_{hor}, \boldsymbol{K}_{hor}, \boldsymbol{V}_{hor}))$
    $\boldsymbol{z}_{hor} \leftarrow Projection_1(LN(FF_1(\boldsymbol{H}_{hor}) + \boldsymbol{H}_{hor}))$
    $\boldsymbol{H}_{ver} \leftarrow LN(GNN_2(\boldsymbol{W}_{ver}) + Attention_2(\boldsymbol{Q}_{ver}, \boldsymbol{K}_{ver}, \boldsymbol{V}_{ver}))$
    $\boldsymbol{z}_{ver} \leftarrow Projection_1(LN(FF_2(\boldsymbol{H}_{ver}) + \boldsymbol{H}_{ver}))$
    $\mathcal{L}_{cl} \leftarrow \frac{1}{N}\sum_{i=1}^{N} -\frac{\boldsymbol{z}_{ver}^{(i)}}{\|\boldsymbol{z}_{ver}^{(i)}\|_2}\frac{\boldsymbol{z}_{hor}^{(i)}}{\|\boldsymbol{z}_{hor}^{(i)}\|_2}$
    $\boldsymbol{H}' \leftarrow LN(\boldsymbol{Z} + Attention_3(\boldsymbol{Q}', \boldsymbol{K}', \boldsymbol{V}'))$
    $\boldsymbol{W}'_t \leftarrow LN(FF_3(\boldsymbol{H}') + \boldsymbol{H}')$
    $\mathcal{L}_{recon} \leftarrow \left\|\boldsymbol{W}_t - \boldsymbol{W}'_t\right\|_F^2$
    $\boldsymbol{z}_u \leftarrow Projection_2(LSTM_1(\boldsymbol{W}_t))$
    $\boldsymbol{z}_l \leftarrow Projection_2(LSTM_2(\boldsymbol{W}'_t))$
    $\mathcal{L}_{scl} \leftarrow \frac{1}{N}\sum_{i=1}^{N} -\frac{\boldsymbol{z}_u^{(i)}}{\|\boldsymbol{z}_u^{(i)}\|_2}\frac{\boldsymbol{z}_l^{(i)}}{\|\boldsymbol{z}_l^{(i)}\|_2}$
    $\mathcal{L} \leftarrow \mathcal{L}_{recon} + \lambda_1\mathcal{L}_{cl} + \lambda_2\mathcal{L}_{scl}$
  $e \leftarrow e + 1$

**Algorithm 2** Testing algorithm.

**Input:** Test window Dataset $\hat{\boldsymbol{W}} = \left\{ \hat{\boldsymbol{W}}_1, \ldots, \hat{\boldsymbol{W}}_{T'} \right\}$, threshold $\lambda$

**Output:** Labels y:$\{ y_1, \ldots, y_{T'} \}$

**for** $t = 1$ *to* $T'$ **do**

$\hat{\boldsymbol{W}}_t \leftarrow D(E(\hat{\boldsymbol{W}}_t))$

$\boldsymbol{S}_t \leftarrow \left\| \hat{\boldsymbol{W}}_t - \hat{\boldsymbol{W}}'_t \right\|_F^2$

**if** $\boldsymbol{S}_t \geq \lambda$ **then**

$s_{an} \leftarrow abnormal$

$y_t \leftarrow 1$

**else**

$s_{an} \leftarrow normal$

$y_t \leftarrow 0$

**Table 2**
Benchmark datasets.

| Dataset name | Dimension number | Train set size | Test set size | Anomaly ratio (%) |
|---|---|---|---|---|
| MSL | 55 | 58,317 | 73,729 | 10.72 |
| SMD | 38 | 708,405 | 748,420 | 4.16 |
| SWaT | 51 | 475,200 | 449,919 | 10.72 |
| PSM | 25 | 132,481 | 87,841 | 27.8 |
| ASD | 19 | 102,331 | 51,840 | 4.61 |
| SMAP | 25 | 135,183 | 427,617 | 13.13 |

### *4.5. Complexity analysis*

Theoretically, the computational complexity of the GDCMAD model is primarily contributed by four core components: data preprocessing, autoencoder operations, LSTM-based contrastive learning, and loss function computation. The data preprocessing stage involves the window division and graph construction of multivariate time series: for the input data with dimension M and length T, the complexity of the division process with window size K is $O(T \cdot M)$, and the subsequent construction of the time-dimension graph (across time-step dependencies) and the variable-dimension graph (across variable dependencies) has a complexity of $O(K^2)$ and $O(M^2)$, respectively. In the autoencoder operations, the graph neural network (GNN) handles temporal branching (number of nodes $N_t = K$) and variable branching (number of nodes $N_v = M$) with a single-layer GNN with a complexity of $O(E \cdot d + N \cdot d^2)$ (E is the number of edges, and d is the dimension of the hidden layer). The total complexity for both branching structures sums to $O((K^2+M^2)\cdot d+(K+M)\cdot d^2)$. The single-head attentional complexity of the improved KAN attentional mechanism in double branching is $O(N^2 \cdot d)$, and the overall realisation $O((K^2 + M^2) \cdot d)$. The feature fusion and reconstruction phase consists of projected feature splicing,a decoding process with complexity $O(d \cdot (K + M) \cdot d)$ and an attention mechanism in the decoder with a fully connected layer of $O(K \cdot M \cdot d)$. In the LSTM-based contrastive learning, the complexity of a single-layer LSTM for processing two-branch data is $O(2 \cdot K \cdot M \cdot d)$, whereas the contrastive loss computation is achieved by projecting the cosine similarity between features with a complexity of $O(N \cdot d)$ (N is the batch size). The total loss function covers the reconstruction loss $L_{recon}$, the graph contrastive loss $L_{cl}$ and the external contrastive loss $L_{scl}$.Their computational complexities are $O(K \cdot M)$, $O(N \cdot d)$ and $O(N \cdot d)$ respectively, which together constitute the final computational load for model training.

## 5. Experimental results

### *5.1. Setup*

We employ a total of five datasets in our experiments: Mars Science Laboratory Rover (MSL) [23], Server Machine Dataset (SMD) [8], Secure Water Treatment (SWaT) [24], Pooled Server Metrics (PSM) [25], Application Server Dataset (ASD) [26], and Soil Moisture Active Passive satellite (SMAP) [23]. A summary of their detailed features is provided in Table 2.

Furthermore, our method's and baselines' performance evaluation is conducted using metrics such as F1* score (F1*), F1 score (F1), recall (R), and precision (P) [27].

We adopt the following default hyperparameters: The batch size is consistently set to 128 across the training, validation, and test datasets. The network model is optimized using the Adam optimizer with an initial learning rate of $10^{-4}$. During training, the model runs for a maximum of 250 epochs, with early stopping (patience = 5) applied to halt training if performance stagnates. The GNN architecture consists of a single layer.

**Table 3**
Performance comparison.

| Methods | MSL | | | | SMD | | | |
|---|---|---|---|---|---|---|---|---|
| | F1* | F1 | R | P | F1* | F1 | R | P |
| LSTM | 0.9189 | 0.8931 | **0.9999** | 0.8500 | 0.8766 | 0.8596 | 0.8682 | 0.8852 |
| VAE | 0.9308 | 0.9096 | 0.9897 | 0.8785 | 0.8050 | 0.7763 | 0.7488 | 0.8703 |
| LSTM-VAE | 0.8899 | 0.8555 | 0.9805 | 0.8147 | 0.8856 | 0.8707 | 0.8764 | 0.8950 |
| BeatGAN | 0.9330 | 0.9199 | 0.9743 | 0.8951 | 0.8608 | 0.8525 | 0.8408 | 0.8817 |
| USAD | 0.9118 | 0.8866 | 0.9400 | 0.8852 | 0.7766 | 0.7493 | 0.7438 | 0.8124 |
| AMFormer | 0.9260 | 0.9132 | 0.9960 | 0.8652 | 0.7619 | 0.7343 | 0.8579 | 0.6852 |
| COCA | 0.9384 | 0.9217 | **0.9999** | 0.8777 | 0.9223 | 0.9186 | 0.9460 | 0.8997 |
| MGCLAD | 0.9308 | 0.9088 | 0.9897 | 0.8785 | 0.8907 | 0.8870 | 0.8741 | 0.8945 |
| MTGFLOW | 0.9163 | 0.8836 | 0.9652 | 0.8721 | 0.9303 | 0.9273 | 0.9346 | 0.9261 |
| GDCMAD | **0.9507** | **0.9423** | 0.9804 | **0.9229** | **0.9523** | **0.9496** | **0.9591** | **0.9455** |
| Methods | SWaT | | | | PSM | | | |
| | F1* | F1 | R | P | F1* | F1 | R | P |
| LSTM | 0.5912 | 0.5912 | 0.5426 | 0.4387 | 0.8780 | 0.8780 | 0.8971 | 0.9597 |
| VAE | 0.8444 | 0.8444 | 0.7604 | 0.9494 | 0.8400 | 0.8400 | 0.8319 | 0.8483 |
| LSTM-VAE | 0.8195 | 0.8195 | 0.7190 | 0.9526 | 0.9201 | 0.9201 | 0.9193 | 0.9209 |
| BeatGAN | 0.6723 | 0.6723 | 0.8310 | 0.5644 | 0.8627 | 0.8627 | 0.8193 | 0.9111 |
| USAD | 0.8619 | 0.8619 | 0.8110 | 0.9196 | 0.8877 | 0.8877 | 0.8130 | **0.9775** |
| AMFormer | 0.5410 | 0.5410 | 0.7488 | 0.4235 | 0.9613 | 0.9613 | 0.9628 | 0.9599 |
| COCA | 0.9441 | 0.9441 | 0.9364 | 0.9520 | 0.9477 | 0.9477 | 0.9326 | 0.9633 |
| MGCLAD | 0.9493 | 0.9493 | 0.9334 | 0.9658 | 0.9560 | 0.9560 | 0.9296 | 0.9839 |
| MTGFLOW | 0.9310 | 0.9310 | **0.9523** | 0.9107 | 0.9339 | 0.9339 | 0.9340 | 0.9460 |
| GDCMAD | **0.9501** | **0.9501** | 0.9232 | **0.9787** | **0.9666** | **0.9666** | **0.9674** | 0.9657 |
| Methods | ASD | | | | SMAP | | | |
| | F1* | F1 | R | P | F1* | F1 | R | P |
| LSTM | 0.8455 | 0.8128 | 0.7763 | 0.9281 | 0.8371 | 0.7723 | 0.9922 | 0.7240 |
| VAE | 0.5771 | 0.4793 | 0.5944 | 0.5608 | 0.8367 | 0.773 | 0.9821 | 0.7288 |
| LSTM-VAE | 0.8718 | 0.8507 | 0.8607 | 0.8833 | 0.8506 | 0.7917 | 0.988 | 0.7468 |
| BeatGAN | 0.8311 | 0.8092 | 0.8135 | 0.8494 | 0.8271 | 0.7757 | 0.951 | 0.7318 |
| USAD | 0.6027 | 0.4890 | 0.5384 | 0.6842 | 0.8312 | 0.7678 | 0.9938 | 0.7144 |
| AMFormer | 0.7460 | 0.7244 | 0.8342 | 0.6746 | 0.8459 | 0.8097 | 0.9919 | 0.7374 |
| COCA | 0.8249 | 0.8032 | 0.8947 | 0.7653 | 0.8693 | 0.8142 | 0.9903 | 0.7746 |
| MGCLAD | 0.9251 | 0.9177 | 0.9176 | 0.9326 | 0.8462 | 0.7909 | 0.9956 | 0.7358 |
| MTGFLOW | 0.9082 | 0.8995 | 0.9055 | 0.9110 | 0.8464 | 0.7866 | 0.9847 | 0.7421 |
| GDCMAD | **0.9304** | **0.9248** | **0.9242** | **0.9367** | **0.9071** | **0.8858** | **0.9987** | **0.8310** |

### 5.2. Overall performance

In real-world manufacturing environments, abnormal events typically do not emerge singly but instead persist for a duration, creating sustained periods of irregularity[27]. Consequently, our method focuses on the behavior of data over extended intervals rather than the performance at individual moments. To address this issue, we employ the point adjustment technique [27] for model optimization. Upon the detection of an anomaly across a specific time period, that interval is designated as the anomaly period, with the contained data points being labeled as abnormal points.

For the assessment of our method's comprehensive performance, we compare it with baseline methods such as LSTM [28], VAE [29], LSTM-VAE [30], BeatGAN [31], USAD [27], AMFormer [32], COCA [13], MGCLAD [6], and MTGFLOW [33]. To ensure a fair comparison, although different methods may propose various threshold determination strategies, for each method, we unify the approach by selecting the threshold that delivers the optimal performance.

Table 3 shows the results for all methods across the MSL, SMD, SWaT, PSM, ASD, and SMAP datasets. Bolding indicates optimal values, and underlining indicates sub-optimal values. Our method outperforms the other methods on all datasets. In particular, it outperforms the F1 scores of the optimal benchmark by 2.2 %, 2.4 %, and 8.8 % for the MSL, SMD, and SMAP datasets, respectively. This is mainly due to the more comprehensive consideration of global features in our method. In addition, our method is significantly different from those autoencoder frameworks (e.g., COCA, MGCLAD) that rely solely on reconstruction mechanisms. In our method, we introduce an external contrastive learning module that imposes specific constraints in processing both raw and reconstructed data. This design not only improves the overall effect of data reconstruction but also achieves mutual proximity between normal data points. Meanwhile, abnormal data points are pushed further away from each other. In this way, we successfully enhance the model's anomaly detection capability. Additionally, it also mitigates to some extent the problem of uncertainty that may arise from contrastive learning based on data augmentation.

Moreover, we perform the Nemenyi test at a 0.05 significance level for a rigorous performance comparison (Critical Difference, CD = 5.5301). As depicted in the Fig. 3, the most critical finding is that GDCMAD is statistically superior to the majority of the

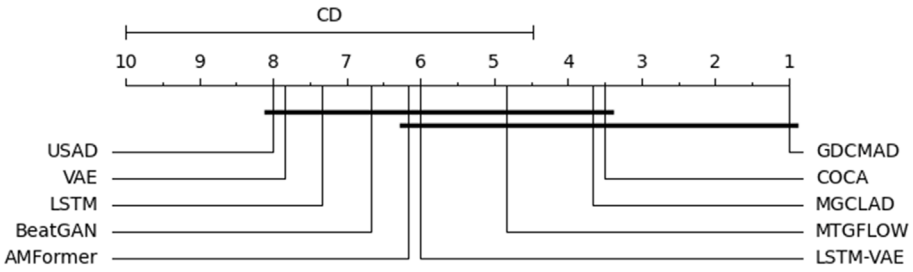


**Fig. 3.** Significance testing.

**Table 4**
F1-score of different methods across multiple random seeds.

| Methods | MSL | SMD | SWaT | PSM | ASD | SMAP |
|---|---|---|---|---|---|---|
| LSTM | 0.8836 ± 0.0134• | 0.8484 ± 0.0159• | 0.5830 ± 0.0117• | 0.8761 ± 0.0028• | 0.8022 ± 0.0150• | 0.7722 ± 0.0002• |
| VAE | 0.8919 ± 0.0251• | 0.7689 ± 0.0105• | 0.8422 ± 0.0031• | 0.8298 ± 0.0145• | 0.4746 ± 0.0066• | 0.7647 ± 0.0117• |
| LSTM-VAE | 0.8463 ± 0.0130• | 0.8662 ± 0.0064• | 0.8064 ± 0.0185• | 0.9181 ± 0.0028• | 0.8439 ± 0.0096• | 0.7917 ± 0.0001• |
| BeatGAN | 0.9193 ± 0.0009• | 0.8448 ± 0.0109• | 0.6590 ± 0.0189• | 0.8623 ± 0.0006• | 0.8062 ± 0.0042• | 0.7641 ± 0.0164• |
| USAD | 0.8862 ± 0.0006• | 0.7443 ± 0.0071• | 0.8618 ± 0.0001• | 0.8809 ± 0.0096• | 0.4842 ± 0.0069• | 0.7602 ± 0.0108• |
| AMFormer | 0.8995 ± 0.0194• | 0.7153 ± 0.0269• | 0.5398 ± 0.0017• | 0.9607 ± 0.0009∘ | 0.7104 ± 0.0199• | 0.7954 ± 0.0203• |
| COCA | 0.9189 ± 0.0040• | 0.9174 ± 0.0018• | 0.9418 ± 0.0033• | 0.9408 ± 0.0098• | 0.7898 ± 0.0190• | 0.7976 ± 0.0235• |
| MGCLAD | 0.8979 ± 0.0154• | 0.8724 ± 0.0024• | 0.9473 ± 0.0029∘ | 0.9533 ± 0.0039• | 0.9169 ± 0.0011• | 0.7717 ± 0.0272• |
| MTGFLOW | 0.8789 ± 0.0067• | 0.9259 ± 0.0020• | 0.9264 ± 0.0066• | 0.9250 ± 0.0126• | 0.8868 ± 0.0180• | 0.7848 ± 0.0025• |
| GDCMAD | **0.9409 ± 0.0020** | **0.9490 ± 0.0009** | **0.9492 ± 0.0013** | **0.9633 ± 0.0047** | **0.9251 ± 0.0004** | **0.8738 ± 0.0170** |

compared methods. Specifically, its average rank (1.0000) significantly surpasses those of LSTM-VAE (6.0000), AMFormer (6.1667), BeatGAN (6.6667), LSTM (7.3333), VAE (7.8333), and USAD (8.0000), as the differences in rank all exceed the CD value. While the connecting bar indicates no statistically significant difference between GDCMAD and other high-performing methods like COCA, MGCLAD, and MTGFLOW, which collectively form the top tier, it is crucial to highlight that GDCMAD is the only model in this elite group to maintain a significant statistical advantage over all lower-ranked models (from LSTM-VAE to USAD). This distinction firmly establishes its superior and stable performance, cementing its leading position.

### *5.3. Robustness across random seeds*

To ensure that the superior performance of the proposed GDCMAD model is not attributable to randomness, we conduct additional experiments by running all methods with five random seeds across six benchmark datasets. For each dataset, every model is executed five times under different random seeds, and the mean and standard deviation of the F1-score are reported in Table 4. Our proposed GDCMAD consistently achieves the highest mean F1 scores across all datasets, accompanied by notably small standard deviations, underscoring its reliability and insensitivity to random seed variations.

To further evaluate whether the observed performance differences are statistically significant, we conduct paired two-tailed t-tests at a 0.05 significance level between GDCMAD and each competing method. A solid circle • denotes that GDCMAD is statistically superior to the compared method, while an open circle ∘ indicates no significant difference. Out of the 54 statistical comparisons conducted (9 baselines across 6 datasets), GDCMAD demonstrates statistically significant superiority in 52 cases, representing 96.3 % of the total comparisons. This highlights that its improvements are consistent and not incidental.

Overall, these results confirm that GDCMAD not only surpasses existing methods in terms of average performance but also exhibits remarkable stability across different random seeds. This robustness further underscores the reliability and generalizability of our model in time series anomaly detection tasks.

### *5.4. Visualization of anomaly scores*

For further exploration of the different methods' performance, we provide a detailed analysis of their normalized anomaly scores computed on the SMD dataset. Fig. 4 illustrates the distribution of anomaly scores for LSTM, VAE, LSTM-VAE, BeatGAN, USAD, AMFormer, COCA, MGCLAD, MTGFLOW and our GDCMAD. In these plots, the shaded areas indicate that the corresponding moments are labeled as anomalies.

In Fig. 4(a) and (h), although they improve precision and recall due to their temporal feature extraction capability, the scores of their normal and abnormal points are too close to the thresholds for effective differentiation, which also affects their performance. Inspecting Fig. 4(b), VAE lacks in-depth mining of time series dependencies, resulting in many normal data points being incorrectly classified as abnormal, which directly affects its performance. From Fig. 4(c)–(g), it can be seen that the model based on the autoencoder architecture does not impose appropriate constraints when dealing with the raw data and its reconstructed data. This absence leads to an obvious consequence: anomaly scores for data points are generally high. High anomaly scores indicate that there is a significant difference between the raw and reconstructed data, in other words, the reconstruction process produces larger errors.

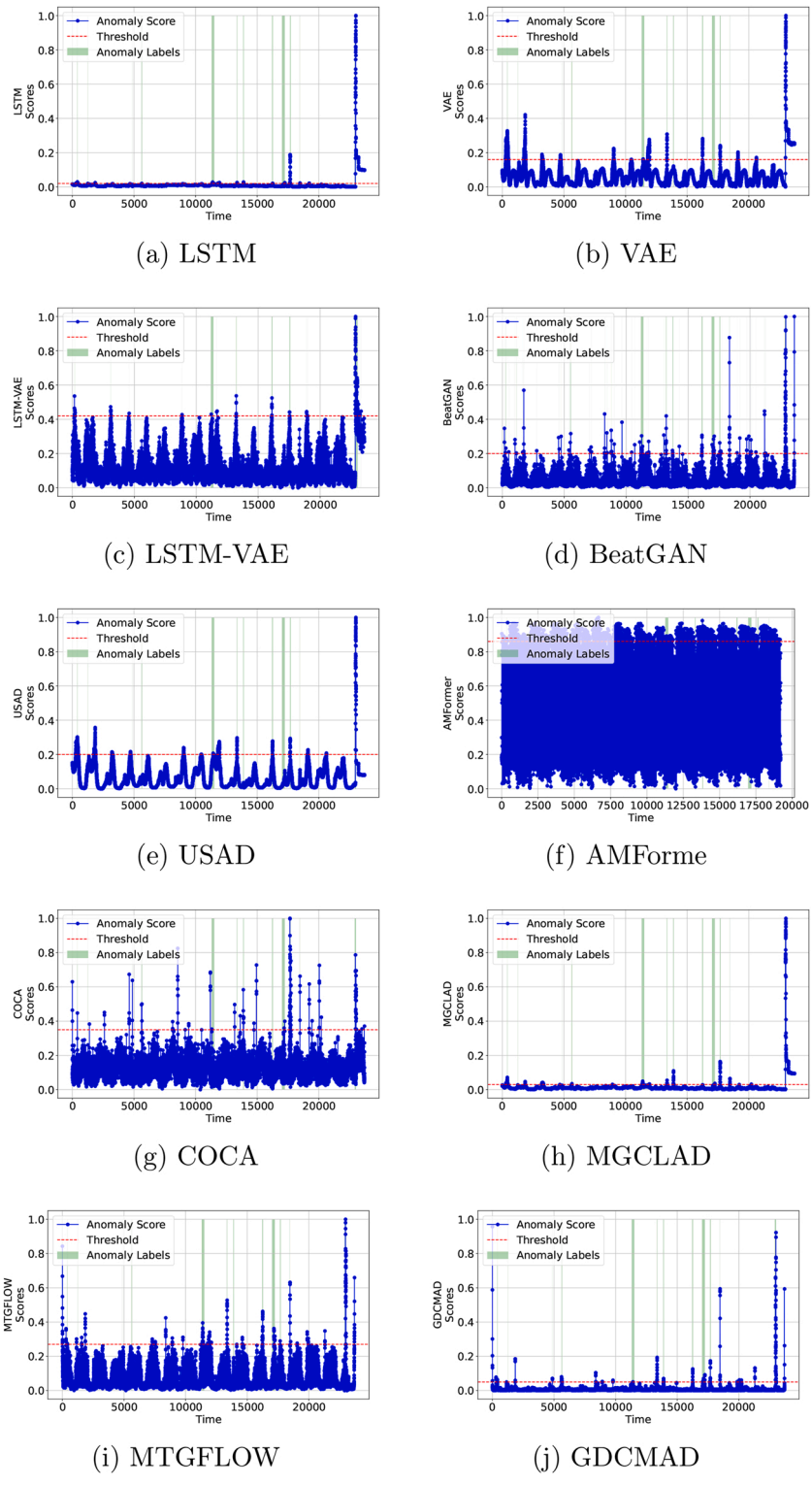


**Fig. 4.** The visualization of the anomaly score.

This leads to a degradation in performance. On the contrary, Fig. 4(j) demonstrates the excellent performance of our method in data reconstruction, where the gap between scores for abnormal and normal points is larger than that in Fig. 4(i) and other methods. This larger difference helps to distinguish normal and abnormal data more effectively, which in turn improves the detection performance.

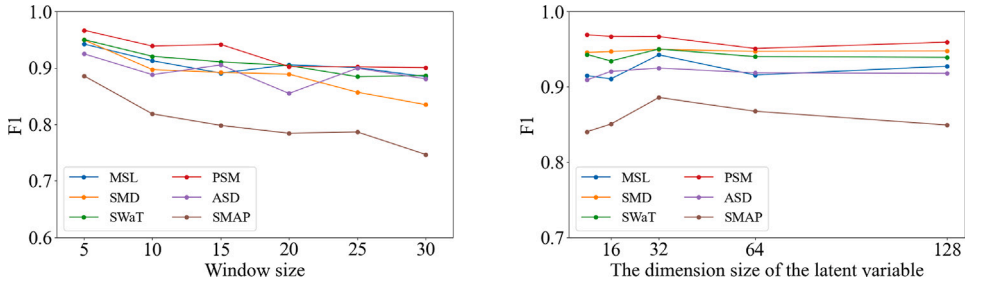


**Fig. 5.** Effect of parameters on the F1 score. Each figure represents one parameter.

### 5.5. Effect of parameters

In our experiments, we evaluate the impact of different parameters on the performance of the method, including two key parameters: window size and dimension size of the latent variable.

**Window size.** This parameter is tested across six datasets. Fig. 5(a) displays the findings from the six experimental sets for different window sizes. It is observed that the experimental results of MSL and ASD datasets fluctuate more as the window size is enlarged, but in general, the F1 scores of each dataset decrease as the window size increases. This is mainly due to larger window sizes allowing small segments of anomalous data to be included in longer sequences, which makes them difficult to detect. Nonetheless, F1 scores of GDCMAD remain at a high level, indicating its excellent performance in handling long sequences.

**The dimension size of the latent variable.** Fig. 5(b) shows the six groups of experimental results under different dimension sizes of the latent variables. Observations show that the experimental results of MSL and SWaT datasets fluctuate greatly as the dimension size increases, but in general, F1 scores for each dataset show a tendency to increase and subsequently decrease. As the dimension size begins to increase, F1 scores rise, and we speculate that this may be because the model has not learned enough features from the sequential data. However, when the value of the dimension size continues to increase, we notice that the F1 score begins to show a decreasing trend. A possible reason for this decline is that the learned latent variable features begin to contain more and more redundant information. This redundant information adversely affects the data reconstruction process, and it may interfere with the extraction of key features by the model. As a result, the precision of reconstruction drops, which subsequently results in a decline in overall performance.

### 5.6. Ablation study

The following two ablation experiments are conducted to assess the contribution of key modules to anomaly detection performance: firstly, by removing or replacing the three core components, we create three different variants. Secondly, to accurately demonstrate the role of the KAN module in the detection behavior of time series and how optimizations we implemented have enhanced detection, we create two variants of the model for comparative analysis.

Comparisons are made between our original method and three variants: *w/o* CL, in which the external contrastive learning module is removed from the model, *w/o* GTime, which is aimed at the extraction of temporal dependencies where we use LSTM instead of graph structure learning, and *w/o* A, which removes the attention mechanism when adding global features. The details are shown in Table 5.

Concerning the impact of the external contrastive learning module, we can gain some insights from the experimental results of the *w/o* CL variant. When removing this module, we observe a decrease in the model's average performance, with an average decrease of 1.7 %. Especially on the PSM dataset, the model's performance degradation is particularly significant, approaching 4 %. The results illustrate that it is crucial to impose specific constraints when processing both raw and reconstructed data. The constraints not only help maintain the quality of data reconstruction but also promote the proximity of normal data points to one another. At the same time, the constraints also enable abnormal data points to be more clearly distinguished from normal data. Such a treatment is essential for enhancing the overall performance.

The experimental results of the *w/o* GTime variant provide important insights when we evaluate the role of graph structure learning in extracting temporal dependencies. Specifically, when we replace the graph structure learning module with a traditional LSTM network, the model shows a decrease in average performance on six different datasets, with an average decrease of 2.3 %. Especially on the SMAP dataset, the performance degradation is particularly significant, reaching 4 %. The results clearly show that graph structure learning has a more comprehensive ability to capture temporal dependencies between data points, which can significantly enhance model performance.

In exploring the effectiveness of the attention mechanism for acquiring global features, the experimental results of the *w/o* A variant provide us with some insights. Specifically, when we remove the attention mechanism from the model, we find a decrease in average performance on all six datasets, with an average decrease of 1.6 %. Notably, the performance degradation is particularly noticeable on the SMAP dataset, which is close to 4.1 %. We analyze that the absence of the attention mechanism may lead to an incomplete capture of global features by the model. This incomplete capture limits the effectiveness of the contrastive learning module

**Table 5**
Variants Comparison.

| Methods | MSL | | | | | |
|---|---|---|---|---|---|---|
| | CL | $\text{Graph}_{Time}$ | Attention | F1 | R | P |
| *w/o* CL | | √ | √ | 0.9333 | 0.9804 | 0.9103 |
| *w/o* GTime | √ | | √ | 0.9191 | 0.9804 | 0.8956 |
| *w/o* A | √ | √ | | 0.9167 | **0.9999** | 0.8715 |
| Ours | √ | √ | √ | **0.9423** | 0.9804 | **0.9229** |
| Methods | SMD | | | | | |
| | CL | $\text{Graph}_{Time}$ | Attention | F1 | R | P |
| *w/o* CL | | √ | √ | 0.9468 | 0.9615 | 0.9382 |
| *w/o* GTime | √ | | √ | 0.9448 | 0.9626 | 0.9330 |
| *w/o* A | √ | √ | | 0.9476 | **0.9657** | 0.9347 |
| Ours | √ | √ | √ | **0.9496** | 0.9591 | **0.9455** |
| Methods | SWaT | | | | | |
| | CL | $\text{Graph}_{Time}$ | Attention | F1 | R | P |
| *w/o* CL | | √ | √ | 0.9455 | **0.9350** | 0.9562 |
| *w/o* GTime | √ | | √ | 0.9364 | **0.9350** | 0.9377 |
| *w/o* A | √ | √ | | 0.9381 | **0.9350** | 0.9412 |
| Ours | √ | √ | √ | **0.9501** | 0.9232 | **0.9787** |
| Methods | PSM | | | | | |
| | CL | $\text{Graph}_{Time}$ | Attention | F1 | R | P |
| *w/o* CL | | √ | √ | 0.9297 | 0.8970 | 0.9648 |
| *w/o* GTime | √ | | √ | 0.9306 | 0.8956 | **0.9684** |
| *w/o* A | √ | √ | | 0.9618 | 0.9668 | 0.9568 |
| Ours | √ | √ | √ | **0.9666** | **0.9674** | 0.9657 |
| Methods | ASD | | | | | |
| | CL | $\text{Graph}_{Time}$ | Attention | F1 | R | P |
| *w/o* CL | | √ | √ | 0.9187 | 0.9038 | **0.9446** |
| *w/o* GTime | √ | √ | | 0.9168 | 0.9038 | 0.9410 |
| *w/o* A | √ | | √ | 0.9130 | 0.9038 | 0.9338 |
| Ours | √ | √ | √ | **0.9248** | **0.9242** | 0.9367 |
| Methods | SMAP | | | | | |
| | CL | $\text{Graph}_{Time}$ | Attention | F1 | R | P |
| *w/o* CL | | √ | √ | 0.8522 | 0.9952 | 0.7947 |
| *w/o* GTime | √ | | √ | 0.8519 | 0.9952 | 0.7962 |
| *w/o* A | √ | √ | | 0.8512 | 0.9882 | 0.7973 |
| Ours | √ | √ | √ | **0.8858** | **0.9987** | **0.8310** |

in the encoder, which further affects the data reconstruction. Finally, these factors collectively contribute to the model's decline in anomaly detection performance.

Additionally, we conduct experiments to evaluate how KAN influences time-series anomaly detection and how its improvements contribute to enhanced detection performance. The detailed experimental results can be found in Table 6. We compare two variants: (1) *w/o* IK & *w/* MLP, where the attention mechanism replaces the improved KAN with an MLP, and (2) *w/o* IK & *w/* KAN, where the attention mechanism retains the original KAN. In comparing the performance of "*w/o* IK & *w/* MLP" and "*w/o* IK & *w/* KAN" on the six datasets, we find that the detection outcomes for the two methods complement each other across various datasets. This phenomenon may be attributed to the fact that KAN performs better on specific tasks such as fitting data and solving partial differential equations, whereas its effect may be less significant when dealing with real-world scenarios such as time series data. In our study, we improve KAN by replacing the B-spline function with LeakyReLU to obtain the GDCMAD proposed in this paper. Our model demonstrates improved detection performance on all datasets, regardless of their size, as shown by the experimental results.

### *5.7. Validation study*

In this section, our goal is to explore how time-series data augmentation might influence the inherent properties of time-series data in the context of contrastive learning. Experimental results are presented in Table 7. To this end, we perform two experimental setups on the three datasets PSM, ASD, and SMAP: 1) *w/* AUG: In this setup, we input the augmented time series data into GDCMAD along with the raw data. 2) *w/* AUG & *w/o* EXT: In this setup, we remove the external contrastive learning module and input the

**Table 6**
A comparative study on the effects of KAN variants.

| Methods | MSL | | | SMD | | |
|---|---|---|---|---|---|---|
| | F1 | R | P | F1 | R | P |
| *w/o* IK & *w/* MLP | 0.9255 | **0.9804** | 0.9052 | 0.9456 | 0.9589 | 0.9377 |
| *w/o* IK & *w/* KAN | 0.9150 | **0.9804** | 0.8921 | 0.9482 | **0.9622** | 0.9404 |
| GDCMAD | **0.9423** | **0.9804** | **0.9229** | **0.9496** | 0.9591 | **0.9455** |
| Methods | SWaT | | | PSM | | |
| | F1 | R | P | F1 | R | P |
| *w/o* IK & *w/* MLP | 0.9365 | **0.9350** | 0.9379 | 0.9310 | 0.8967 | 0.9680 |
| *w/o* IK & *w/* KAN | 0.9356 | **0.9350** | 0.9362 | 0.9309 | 0.8956 | **0.9690** |
| GDCMAD | **0.9501** | 0.9232 | **0.9787** | **0.9666** | **0.9674** | 0.9657 |
| Methods | ASD | | | SMAP | | |
| | F1 | R | P | F1 | R | P |
| *w/o* IK & *w/* MLP | 0.9242 | 0.9141 | **0.9438** | 0.8146 | 0.9769 | 0.7540 |
| *w/o* IK & *w/* KAN | 0.9217 | 0.9551 | 0.8995 | 0.8540 | 0.9769 | 0.8146 |
| GDCMAD | **0.9248** | **0.9242** | 0.9367 | **0.8858** | **0.9987** | **0.8310** |

**Table 7**
Validation study.

| Methods | PSM | | | ASD | | |
|---|---|---|---|---|---|---|
| | F1 | R | P | F1 | R | P |
| *w/* AUG | 0.9632 | **0.9681** | 0.9584 | 0.9221 | 0.9090 | **0.9484** |
| *w/* AUG & *w/o* EXT | 0.9527 | 0.9969 | 0.9122 | 0.9192 | 0.9141 | 0.9370 |
| GDCMAD | **0.9666** | 0.9674 | **0.9657** | **0.9248** | **0.9242** | 0.9367 |
| Methods | SMAP | | | AVERAGE | | |
| | F1 | R | P | F1 | R | P |
| *w/* AUG | 0.8554 | **0.9987** | 0.8045 | 0.9135 | 0.9586 | 0.9038 |
| *w/* AUG & *w/o* EXT | 0.8551 | 0.9922 | 0.8038 | 0.9090 | **0.9677** | 0.8843 |
| GDCMAD | **0.8858** | **0.9987** | **0.8310** | **0.9257** | 0.9634 | **0.9111** |

augmented time series data along with the raw data. Based on the figures in Table 7, we find that the performance of both *w/* AUG and *w/* AUG & *w/o* EXT degrades with the addition of the augmented data, which indicates that time-series data augmentation may negatively impact the intrinsic properties of the time-series data. In addition, *w/* AUG slightly outperforms *w/* AUG & *w/o* EXT, which suggests that augmenting time-series data could potentially harm its inherent properties.

### *5.8. Noise experiment*

In this section, we evaluate the robustness of GDCMAD to real-world noisy data by injecting Gaussian noise at intensities of 0 %, 5 %, 10 %, 15 %, and 20 % into the PSM and SWaT datasets. As shown in Fig. 6, the F1-score remains relatively stable as the noise level increases, exhibiting only a slight decline. Notably, even under 20 % Gaussian noise, the model achieves a high F1-score with just a 2.6 % reduction compared to the noise-free baseline. This robustness stems from our dual contrastive learning framework, which integrates graph structural contrast to preserve spatio-temporal topological consistency and LSTM-based temporal contrast to learn noise-invariant representations. The combination of these two objectives enables the model to mitigate the impact of noise on anomaly detection performance.

### *5.9. Model interpretability analysis*

To elucidate the decision-making mechanism of the proposed GDCMAD model, we conduct an interpretability analysis using SHAP (SHapley Additive exPlanations) values. This analysis aims to quantify the contribution of each input feature to the final anomaly score, thereby providing insights into which variables and time steps the model deems most critical for its detection. Given the dual emphasis of our model on capturing both inter-variable and temporal dependencies, we extend the SHAP framework to evaluate feature importance across the entire spatio-temporal input matrix. Fig. 7 shows the SHAP heatmap on the PSM dataset. In this visualization, the horizontal axis denotes time, while the vertical axis denotes variables. The color intensity at any coordinate reflects the magnitude of the SHAP value, with brighter hues indicating a stronger contribution to the anomaly score. The heatmap reveals that the model's decision is not homogeneously distributed across the input space. Instead, it is predominantly driven by a concentrated subset of variables exhibiting significant deviations within a specific, narrow time period. This pattern demonstrates that GDCMAD effectively pinpoints anomalies by focusing on localized spatio-temporal patterns rather than responding to global,

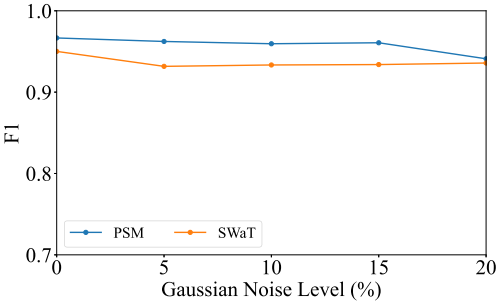


**Fig. 6.** Effect of noise level on the F1 score.

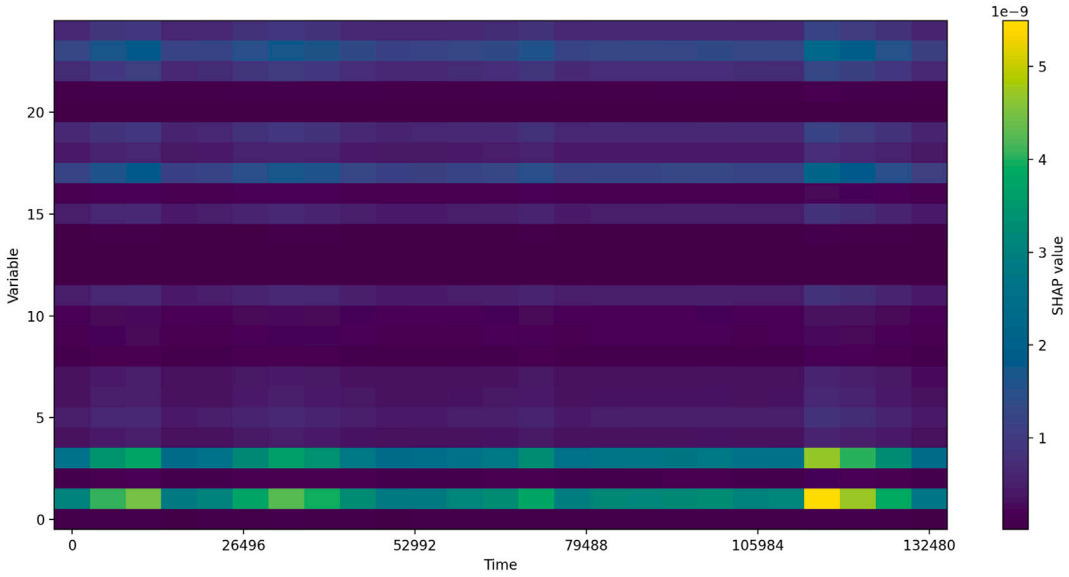


**Fig. 7.** SHAP heatmap on the PSM dataset.

diffuse changes. Moreover, the heatmap highlights that the model captures structured correlations: some variables show spikes in importance that coincide with correlated fluctuations in other dimensions, reflecting the role of graph-based modeling in focusing on interdependent signals.

This interpretability analysis reinforces that the model's architectural components, designed to learn from both variable interactions and temporal dynamics, collectively enable it to identify and attribute anomalies to their root causes in the data.

### *5.10. Limitations and practical scope*

To better characterize the practical scope of GDCMAD, we provide an analysis of its performance boundaries that complements the preceding experiments.

Although the parameter studies in Section 5.5 show that the model remains generally stable across different hyperparameter settings, performance variability on datasets such as MSL and SMAP indicates a sensitivity to configuration. Specifically, the model may fail to reliably capture anomalies that are very short in duration compared to the chosen context window, as their distinctive characteristics can be diluted within long input windows.

The noise experiments in Section 5.8 demonstrate that the method is resilient under moderate noise corruption, yet the model's performance is inevitably challenged in environments where the noise amplitude rivals or exceeds that of true anomalies. This implies that in high-noise environments, the model struggles to differentiate true anomalies from noise-induced artifacts.

In summary, GDCMAD is most effective in environments characterized by stable temporal patterns and a relatively uncontaminated normal data distribution. Conversely, it may be less effective at detecting short-lived anomalies in long sequences, or operating under conditions with substantial noise or anomaly contamination.

## 6. Conclusions

To address the challenges of anomaly detection in MTS, this paper proposes an autoencoder-based unsupervised detection method, i.e., GDCMAD. The method effectively overcomes the problem of limited data representation and insufficient discriminative ability. Experimental results across six publicly available datasets show that GDCMAD achieves superior anomaly detection performance compared to all nine baseline methods. However, with the development of society, data privacy protection is becoming increasingly important. Therefore, in future research, we will focus more on data privacy protection while steadily enhancing the model's anomaly detection.

## CRediT authorship contribution statement

**Sheng He:** Writing – original draft, Software, Methodology, Conceptualization. **Wenxuan He:** Writing – original draft, Software, Methodology, Conceptualization. **Mingjing Du:** Writing – original draft, Project administration, Funding acquisition. **Xiang Jiang:** Writing – review & editing, Validation. **Yongquan Dong:** Project administration, Funding acquisition.

## Declaration of competing interest

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

## Acknowledgments

This work is supported by the Qinglan Project of Jiangsu Province of China, the National Natural Science Foundation of China (Nos. 62006104 and 62577036), Postgraduate Research & Practice Innovation Program of Jiangsu Normal University (No. 2025XKT1452).

## Data availability

Data will be made available on request.

## References

[1] H. Xiang, X. Zhang, M. Dras, A. Beheshti, W. Dou, X. Xu, Deep optimal isolation forest with genetic algorithm for anomaly detection, in: Proceedings of the 23rd IEEE International Conference on Data Mining, 2023, pp. 678–687.
[2] H. Li, M. Chen, Time series clustering based on normal cloud model and complex network, Appl. Soft Comput. 148 (2023) 110876.
[3] M. Du, Y. Wei, Y. Tang, X. Zheng, S. Wei, C. Ji, St-tree with interpretability for multivariate time series classification, Neural Netw 183 (2025) 106951.
[4] G. He, D. Jin, W. Jiang, Z. Zhao, L. Dai, Z. Yu, C.L. Philip Chen, Efficient semi-supervised clustering with pairwise constraint propagation for multivariate time series, Inf. Sci. 681 (2024) 121233.
[5] C. Ding, S. Sun, J. Zhao, MST-GAT: a multimodal spatial–temporal graph attention network for time series anomaly detection, Inf. Fusion 89 (2023) 527–536.
[6] S. Qin, L. Chen, Y. Luo, G. Tao, Multi-view graph contrastive learning for multivariate time series anomaly detection in IOT, IEEE Internet Things J. 10 (24) (2023) 22401–22414.
[7] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T.Y. Hou, M. Tegmark, KAN: Kolmogorov-Arnold networks, arXiv preprint arXiv:2404.19756, 2024.
[8] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2828–2837.
[9] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, N.V. Chawla, A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data, in: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 1409–1416.
[10] J. Wang, S. Shao, Y. Bai, J. Deng, Y. Lin, Multiscale wavelet graph autoencoder for multivariate time-series anomaly detection, IEEE Trans. Instrum. Meas. 72 (2023) 2502911.
[11] L. Chen, X. Gao, J. Liu, Y. Zhang, X. Diao, T. Wang, J. Lu, Z. Meng, A multivariate time series anomaly detection method with multi-grain dynamic receptive field, Knowl.-Based Syst. 309 (2024) 112768.
[12] A.V.D. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, arXiv preprint arXiv:1807.03748, 2018.
[13] R. Wang, C. Liu, X. Mou, K. Gao, X. Guo, P. Liu, T. Wo, X. Liu, Deep contrastive one-class time series anomaly detection, in: Proceedings of the 23rd SIAM International Conference on Data Mining, 2023, pp. 694–702.
[14] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al., Bootstrap your own latent-a new approach to self-supervised learning, Adv. Neural Inf. Process. Syst. 33 (2020), 21271–21284.
[15] D. Chen, X. Zhao, W. Wang, Z. Tan, W. Xiao, Graph self-supervised learning with augmentation-aware contrastive learning, in: Proceedings of the 32nd ACM Web Conference, 2023, pp. 154–164.
[16] K. Hassani, A.H. Khasahmadi, Contrastive multi-view representation learning on graphs, in: Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 4116–4126.
[17] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Graph contrastive learning with adaptive augmentation, in: Proceedings of the 30th Web Conference, 2021, pp. 2069–2080.
[18] Y. Yang, C. Zhang, T. Zhou, Q. Wen, L. Sun, Dcdetector: dual attention contrastive representation learning for time series anomaly detection, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 3033–3045.
[19] H. Li, T. Du, X. Wan, Time series clustering based on relationship network and community detection, Expert Syst. Appl. 216 (2023) 119481.
[20] P. Wesołowski, K. Walasek, W. Homenda, C. Ouyang, F. Yu, Time series classification based on fuzzy cognitive maps and multi-class decomposition with ensembling, in: Proceedings of the 32nd IEEE International Conference on Fuzzy Systems, 2023, pp. 1–8.
[21] D. Fu, Z. Zhang, J. Fan, Dense projection for anomaly detection, in: Proceedings of the 38th AAAI Conference on Artificial Intelligence, 2024, pp. 8398–8408.
[22] Y. Zhang, Y. Sun, J. Cai, J. Fan, Deep orthogonal hypersphere compression for anomaly detection, in: Proceedings of the 12th International Conference on Learning Representations, 2024, pp. 7228.
[23] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 387–395.
[24] A.P. Mathur, N.O. Tippenhauer, Swat: a water treatment testbed for research and training on ICS security, in: Proceedings of the 2nd International Workshop on Cyber-Physical Systems for Smart Water Networks, 2016, pp. 31–36.

[25] A. Abdulaal, Z. Liu, T. Lancewicki, Practical approach to asynchronous multivariate time series anomaly detection and localization, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2485–2494.
[26] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, D. Pei, Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 3220–3230.
[27] J. Audibert, P. Michiardi, F. Guyard, S. Marti, M.A. Zuluaga, USAD: unsupervised anomaly detection on multivariate time series, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3395–3404.
[28] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, et al., Long short term memory networks for anomaly detection in time series, in: Proceedings of the 23rd European Symposium on Artificial Neural Networks, 2015, 89.
[29] J. An, S. Cho, Variational autoencoder based anomaly detection using reconstruction probability, Special Lecture on IE 2 (1) (2015) 1–18.
[30] D. Park, Y. Hoshi, C.C. Kemp, A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder, IEEE Robot. Autom. Lett. 3 (3) (2018) 1544–1551.
[31] B. Zhou, S. Liu, B. Hooi, X. Cheng, J. Ye, BeatGAN: anomalous rhythm detection using adversarially generated time series, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 4433–4439.
[32] G. Zhong, F. Liu, J. Jiang, B. Wang, C.L.P. Chen, Refining one-class representation: a unified transformer for unsupervised time-series anomaly detection, Inf. Sci. 656 (2024) 119914.
[33] Q. Zhou, S. He, H. Liu, J. Chen, W. Meng, Label-free multivariate time series anomaly detection, IEEE Trans. Knowl. Data Eng. 36 (7) (2024) 3166–3179.