

RESEARCH

Open Access



# Efficiently localizing system anomalies for cloud infrastructures: a novel Dynamic Graph Transformer based Parallel Framework

Hongxia He<sup>1</sup>, Xi Li<sup>1</sup>, Peng Chen<sup>1\*</sup>, Juan Chen<sup>1</sup>, Ming Liu<sup>2,3</sup> and Lei Wu<sup>2,3</sup>

## Abstract

Cloud environment is a virtual, online, and distributed computing environment that provides users with large-scale services. And cloud monitoring plays an integral role in protecting infrastructures in the cloud environment. Cloud monitoring systems need to closely monitor various KPIs of cloud resources, to accurately detect anomalies. However, due to the complexity and highly dynamic nature of the cloud environment, anomaly detection for these KPIs with various patterns and data quality is a huge challenge, especially those massive unlabeled data. Besides, it's also difficult to improve the accuracy of the existing anomaly detection methods. To solve these problems, we propose a novel Dynamic Graph Transformer based Parallel Framework (DGT-PF) for efficiently detect system anomalies in cloud infrastructures, which utilizes Transformer with anomaly attention mechanism and Graph Neural Network (GNN) to learn the spatio-temporal features of KPIs to improve the accuracy and timeliness of model anomaly detection. Specifically, we propose an effective dynamic relationship embedding strategy to dynamically learn spatio-temporal features and adaptively generate adjacency matrices, and soft cluster each GNN layer through Diffpooling module. In addition, we also use nonlinear neural network model and AR-MLP model in parallel to obtain better detection accuracy and improve detection performance. The experiment shows that the DGT-PF framework have achieved the highest *F1-Score* on 5 public datasets, with an average improvement of 21.6% compared to 11 anomaly detection models.

**Keywords** Cloud infrastructures, Anomaly detection, Transformer, Graph neural network, Spatio-temporal feature

## Introduction

The cloud monitoring systems can monitor the status and performance of various resources. These include servers, networks, databases and storage. They also provide real-time alarms to help users find and fix problems quickly [1]. It can also closely monitor various KPIs (key

performance indicators) of cloud resources to accurately detect anomalies. These KPIs are heavily influenced by user behavior, schedules, etc., and have roughly seasonal patterns that occur at regular intervals. Time dependence is an important feature of KPIs, it has a big impact on predicting and analyzing data [2]. On the other hand, since user behavior may change on different days, the shape of the KPIs curve is not exactly the same for each repeat cycle. These KPIs instances as shown in Fig. 1, with anomalies in red color. Applications and services on cloud computing platforms are often large-scale and highly concurrent [3]. So, detecting anomalies usually requires processing many KPIs [4, 5]. In addition, the complex and changeable interaction between cloud infrastructures also makes abnormal behaviors more

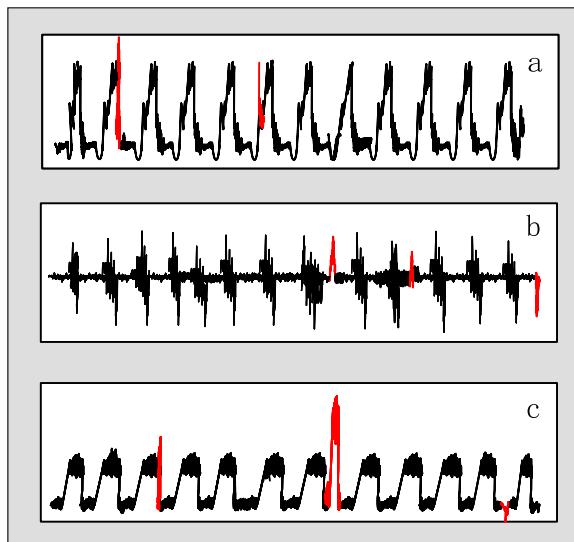
\*Correspondence:

Peng Chen  
chenpeng@mail.xhu.edu.cn

<sup>1</sup> School of Computer and Software Engineering, Xihua University, Chengdu, China

<sup>2</sup> School of Mathematical Sciences, University of Electronic Science and Technology of China, Chengdu, China

<sup>3</sup> Yangtze Delta Region Institute (Qzhou), University of Electronic Science and Technology of China, Qzhou, China



**Fig. 1** An example of KPIs, with anomalies in red color

elusive [6]. How to accurately and efficiently detect abnormal behaviors in cloud infrastructures has also become the focus of enterprises and related fields [7]. Therefore, it is necessary to detect the anomaly behaviors that may exist in cloud infrastructures in time.

Anomaly detection (a.k.a outlier detection) is a hot research in the real world. It can be used in various fields, like financial fraud detection, network intrusion detection, medical diagnosis, etc. Time dependence is an important feature of KPIs. It has a big impact on predicting and analyzing data [8]. According to different learning styles, anomaly detection can be divided into supervised anomaly detection, semi-supervised anomaly detection and unsupervised anomaly detection. Considering it is tedious and time-consuming to label data manually because various anomalies exist, unsupervised learning involves picking up interesting structures in the data, and learning features without labels is popular [9]. Based on this, we focus on unsupervised anomaly detection for KPIs with various patterns and data quality.

To address the challenge of anomaly detection in cloud, researchers and engineers have proposed many new methods. These include machine learning, data mining, and statistical analysis [10]. Before machine learning, traditional time series algorithms are usually used to build statistical models, such as ARMA [11], ARIMA [12], GARCH [13], etc. Remarkably, the relationship presented by KPIs is highly complex, and the intrinsic correlation is often nonlinear. Some of the traditional statistical models just simply model the relationships between KPIs in the cloud environment, capturing only linear relationships. That's why researchers need to design smarter and

better-performing models to detect system anomalies in cloud infrastructures [14]. In recent years, deep learning is one of the latest trends in machine learning and artificial intelligence research, and many major breakthroughs have been made in this field around the world. Examples are CNN [15] and LSTM [16, 17]. These deep learning methods are better able to learn data representation and feature extraction through a combination of multi-layer neural networks and nonlinear activation functions, as well as increasing the depth and width of the network. They are better at handling nonlinear features than traditional machine learning models. However, most of the existing anomaly detection methods focus on either linear, e.g., statistical models including ARMA or nonlinear feature representation, e.g., classic machine learning and deep learning models including KNN [18], while the cloud environment is highly dynamic and complex, resulting in KPIs in the cloud environment has both complicated complex linear and nonlinear dependencies. Moreover, the existing methods focus more on the temporal dependence and inadequate on the spatial features. In the cloud environment, there are complex connections and dependencies between nodes, making space dependency unavoidable. The user node communicates with the cloud service node through the client node, which in turn relies on the network node for data transmission. Cloud service nodes rely on infrastructure support provided by data centers, but also need to communicate with network nodes. Such connections and dependencies between each node form a complex cloud environment system. When performing anomaly detection, effectively extracting the spatial dependency between nodes could contribute to better anomaly identification. Therefore, as for anomaly detection in dynamic runtime cloud environment, existing models are still insufficient in terms of detection accuracy and real-time performance.

In the latest research, GNN is proved to be suitable for spatial dependency learning. It is good at processing the graph structure data and mining the dependency between nodes, and can pass the real high expressiveness achieved through message passing in effective learning graph representation [19]. Transformer stacks multiple layers of attention for fast parallel operation. As a result, it can better distinguish normal and abnormal fluctuations and better capture long-term time correlations. Inspired by combination of GNN and Transformer, we leverage GNN, Transformer and attention to extract the features of nonlinear parts of KPIs data, use the attention mechanism and graph convolution operation to effectively process the complex relationship between nodes [20]. We also fuse the relationship between nodes in the graph and the dependence relationship between positions in the sequence. At the same time, the model captures

the spatial and temporal KPIs data and displays high-dimensional data with complex relationships to improve the performance of the model. Using AR-MLP to extract the features of the linear part of KPIs data, the AutoRegressive (AR) model can maintain the linear relationship between the KPIs, and the nonlinear feature extraction capability of Multi-Layer Perceptron (MLP) can improve the feature extraction performance of AR model. At the same time, it is spatially sensitive, allowing communication between different spatial locations and operating independently on each channel. Therefore, it can better adapt to complex anomaly detection tasks. Based on these ideas, we propose a novel Dynamic Graph Transformer based Parallel Framework (DGT-PF) for efficiently detect system anomalies in cloud infrastructures, which get the highest *F1-Score* on 5 public datasets. It has achieved high-precision and real-time anomaly detection in a cloud environment.

We design an anomaly attention mechanism to replace the self-attention mechanism, and construct an effective Transformer with anomaly attention mechanism model to extract the time features of KPIs, which contains a two-branch structure to model the prior-association and series-association of each time point respectively. The prior-association employs the learnable Gaussian kernel to present the adjacent-concentration inductive bias of each time point, while the series-association corresponds to the self-attention weights learned from raw series. This design can better distinguish between normal and abnormal fluctuations and derive temporal correlations between sequences. In addition, in order to further extract the spatio-temporal features of the context and improve the timeliness of the model, we propose an effective dynamic relationship embedding strategy, which integrate GNN module and Diffpool module to realize the spatio-temporal dependence of the time series and make better spatio-temporal prediction. Meanwhile, we integrate the AR-MLP model and the above nonlinear model to output the final anomaly detection results. Finally, experiments on 5 public datasets verify the accuracy and timeliness of our Framework. The main contributions of this paper are as follows:

- In order to learn the spatio-temporal dependence of KPIs, we propose a Transformer integrated with GNN model to dynamically capture time series features to improve the timeliness of the model.
- We propose a dynamic relationship embedding strategy based on graph structure learning to adaptively learn the adjacency matrix to simulate potential relationships in given KPIs.
- In order to better extract all the feature information of data and adapt to complex prediction tasks, we

use AR-MLP model and nonlinear neural network to integrate in parallel, and further improve the accuracy and robustness of the model.

- We demonstrate that DGT-PF outperforms 11 state-of-art baseline methods on 5 public datasets, with a 21.6% improvement in the average *F1-score*.

The rest of the paper is organized as follows. In the related work, we review the existing statistical and deep learning methods. In the model part, we introduce the DGT-PF model in detail. In the [Experimental result](#) section, we evaluate the accuracy and timeliness of anomaly detection. Finally, the conclusion and future work are summarized.

## Related works

The research on anomaly detection is being carried out rapidly, and methods based on deep learning are widely used [21]. This section explores existing methods for anomaly detection. They are based on statistics and deep learning, with a focus on GNN and Transformer methods.

### Statistics-based method

Traditional statistical methods are mainly used on single-feature KPIs, and most of them are linear methods. ARMA, ARIMA, GARCH and so on are used for time series analysis and prediction. The ARMA model [11] is an autoregressive moving average model, which assumes that KPIs are composed of a linear combination of past observations and random error terms. In literature, Autoregressive Integrated Moving Average (ARIMA) model [12] is used to predict KPIs. Differential operation can change non-stationary time series into stationary time series, so it is easier to establish models. GARCH model [13] is a Generalized AutoRegressive Conditional Heteroscedasticity model, which assumes that the variance of time series changes with time. Chen et al. [22] propose the use of isolated forest and elliptical envelope to detect geochemical anomalies. These methods are good for short-term linear KPIs prediction, but not so good for long-term KPIs prediction. Moreover, it is usually necessary to manually select the features related to anomaly detection, which may not cover all the relevant features, resulting in inaccurate anomaly detection results. In addition, KPIs tend to be multi-dimensional and complicated, and these traditional statistical methods can no longer meet the current needs [23]. Due to the strong expression ability of deep neural networks and their better performance in anomaly detection accuracy, deep learning methods have received extensive attention in the industry in recent years.

### Deep learning-based method

With the advent of the big data era, anomaly detection has become an important task in many fields. As a powerful machine learning technology, deep learning provides many new methods and ideas for anomaly detection. Therefore, for high-dimensional and unlabeled data, more and more researchers use deep learning to detect anomalies in KPIs [24]. There are many models that are based on the LSTM layer and the stacked CNN layer to extract features from time series, and then apply the softmax layer to predict label. Complex structures such as the recursive skip layer (LSTNet-S), the temporal attention layer (LSTNet-A) [25]. LSTM-VAE model [26] integrates LSTM into VAE for anomaly detection, in which LSTM backbone is used for time modeling and VAE is used for reconstruction. VAE-GAN [27] based on LSTM uses LSTM as encoder, generator and discriminator to detect anomalies by reconstructing differences and discriminating results. TapNet [28] also builds the LSTM layer and the stacked CNN layer. Bidirectional molecule generation with RNN model [29] improves the prediction accuracy of the model by adding a direction to the RNN. ELBD (Ensemble Learning-Based Detection) framework [30] integrates four existing detection methods for performance anomaly detection and prediction of cloud applications. USAD [31] uses an autoencoder with two decoders and an adversarial training framework to classify normal and abnormal data. In addition, some deep learning models, including THOC [32] uses RNN with jump connections to effectively extract multi-scale time features from time series, integrates multi-scale time features through hierarchical clustering mechanism, and then detect anomalies through multi-layer distance. MAD-GAN [33] uses a new anomaly score, *DR-score*, to identify and reconstruct the output of GAN's generator and discriminator, so as to detect anomalies. Although anomaly detection models based on deep learning have achieved remarkable results in many scenarios, there are still some limitations, such as data quality, interpretability, and computing resources [34, 35]. They also assume the same effects between time series variables, so they cannot model pair-dependent relationships between variables unambiguously, and anomaly detection accuracy is not high enough. However, data in cloud systems is highly volatile. So, the needed anomaly detection algorithms require more precise results. Recent studies show that GNN combined with Transformer for anomaly detection can effectively improve detection accuracy.

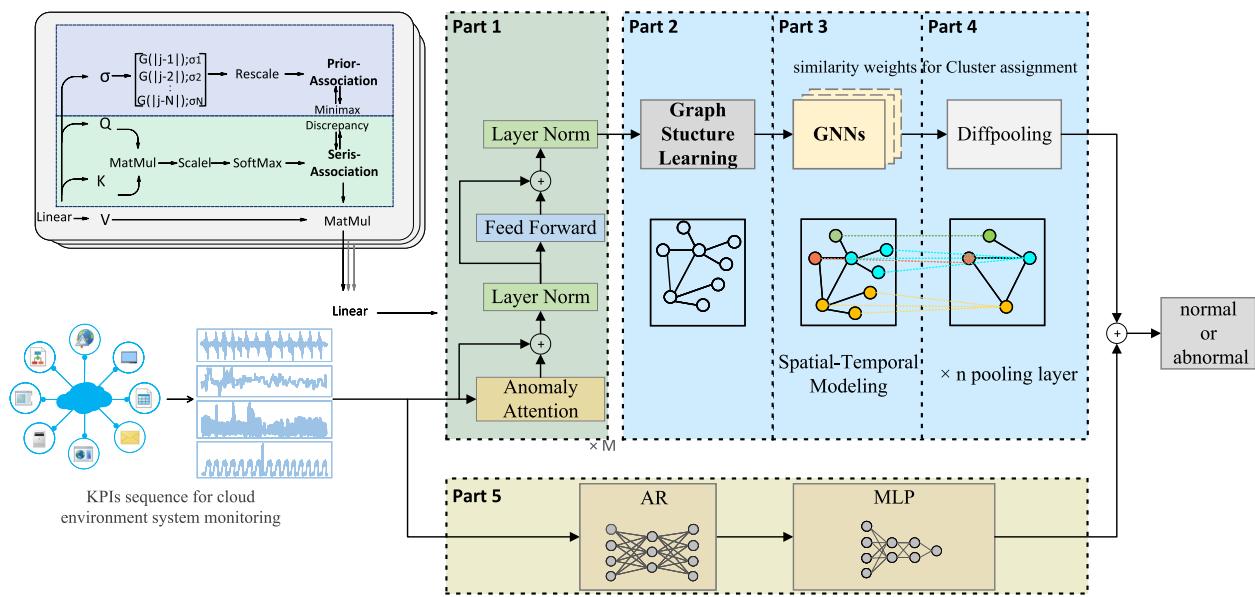
The combination of GNN and Transformer architectures offers several advantages. In traditional anomaly detection, Transformer can use a self-attention mechanism to capture global dependencies and context information of the data, enabling the model to focus more

on important nodes and edges to capture abnormal features. GNN can be used to model relationships and dependencies between data. With the advent of Transformer in natural language processing and computer vision, Embedding the graph structure into the Transformer architecture can overcome the limitations of local neighborhood aggregation while avoiding strict structural induction bias. Graph Transformer [36] introduces the topological structure attribute of graph into Transformer, so that the model has a prior of the structure position in the high-dimensional space. The model combines the heart of Transformer with that of GNN. It does so by considering global and topological properties of the graph. It calculates attention near each node, not on the whole graph. Whereas the SAN model [37] is similar but calculates attention on the entire graph, distinguishing between real edges and created edges. The Graph Attention Network(GAT) [38] is a GNN that uses a self-attention mechanism to capture more complex relationships between nodes. GAT avoids the problem of exploding number of parameters in traditional GCN [39] by aggregating the features of nodes into a shared global space and using a shared weight matrix to calculate the correlation of each node with its neighbors. These models combine GNN with Transformer to take full advantage of both to design a graph Transformer architecture that takes local and global information into account.

While the cloud environment is highly dynamic and complex, resulting in KPIs in the cloud environment has both complicated complex linear and nonlinear dependencies. And the existing methods focus more on the temporal dependence and inadequate on the spatial features. Therefore, our proposed method can effectively extract nonlinear and linear features of dynamically changing KPIs. Firstly, we use a Transformer with anomaly attention to get time features [40]. Then, we propose a strategy for embedding dynamic relationships. It is based on learning graph structure to capture spatio-temporal features and learn the adjacency matrix. In addition, the node embedding vector based on Diffpooling module performs soft clustering for each GNN layer, and then builds the results of the deep GNN output nonlinear module through repeated stacking. Finally, we use the AR-MLP model to better extract all the feature information of KPIs and adapt the complex anomaly detection task. That is, we integrate nonlinear modules and linear modules in parallel to improve the accuracy and timeliness of the entire model.

### Methodology

Next we will introduce the overall framework and detailed modules in detail. The framework diagram of the model is shown in Fig. 2. Suppose monitoring a



**Fig. 2** The overall architecture of DGT-PF

successive system of  $N$  measurements and recording the equally spaced observations over time. Our input is a set of KPIs, defined as  $X = \{x_1, x_2, \dots, x_n\}$ , in general, a constant time interval between two successive measurements, where  $x_t \in R^N$  represents the observed value at time  $t$ .

Actually, to efficiently capture both nonlinear and linear features, the overall framework of DGT-PF is a parallel structure, which is integrated by nonlinear modules and linear modules in parallel. Since the monitoring data in the dynamic cloud includes both nonlinear and linear features, we consider to extract these two types of features independently, then integrate them to achieve better detection accuracy.

The upper channel is the nonlinear feature extraction module of Transformer integrating GNN, as can be seen in Fig. 2(Part 1-4). GNN is able to fuse global and local information through multi-layer messaging, and Transformer is able to capture temporal dependencies of the sequence through attention mechanisms. Combining the two parts can fuse the links between nodes in the graph. It also fuses the ties between positions in the sequence and helps us understand the whole graph structure. At the same time, Transformer integrating GNN model can capture the spatio-temporal features of KPIs, and display high-dimensional data with complex relationships to improve model performance. Among it, we also propose an effective dynamic relationship embedding strategy based on Graph Structure Learning to carry out feature learning within each time window. Then, the whole KPIs data is processed by sliding time window to capture its

dynamic evolution process, so as to better spatio-temporal prediction and effectively mining the dynamic changing KPIs features, as shown in Fig. 2(Part 2). In addition, the node embedding vector based on Diffpooling module performs soft clustering for each GNN layer, as shown in Fig. 2(Part 3-4), and then builds the results of the deep GNN output nonlinear module through repeated stacking.

The lower channel is to capture linear regularities, consisting of an AR model and a MLP model, as demonstrated in Fig. 2(Part 5). Adding AR model can maintain the linear relationship of KPIs, and the nonlinear feature extraction capability of MLP can enhance the feature extraction performance of AR model. At the same time, it is spatially sensitive, allowing communication between different spatial locations, and operates independently on each channel. Thus, it can better adapt to the complex anomaly detection task. Finally, the output of the nonlinear module and the output of the linear module are added by weight to obtain the final anomaly detection results  $x_t$ , which is represented as normal or abnormal.

#### Transformer with anomaly attention for feature representation

As shown in Fig. 2(Part1), considering the limitations of traditional Transformer in anomaly detection, we design an effective Transformer with anomaly attention mechanism. It is characterized by stacking the Anomaly-Attention blocks and feed-forward layers alternately. This stacking structure is conducive to learning underlying associations from deep multi-level features. And

it has two branch structures (on the left of Fig. 2(Part1)). For the prior association, we use learnable Gaussian kernel to calculate the prior relative time distance. Due to the unimodal nature of its Gaussian kernel, its neighborhood can be better focused and the prior association can be adapted to various time series patterns. And sequence association is learning association from original sequence, which can find the most effective association relationship. This section aims to extract temporal features and construct the feature matrix  $X^{(i)}$  as well as to get anomaly attention. The feature matrix for each sequence is as follows:

$$X^{(i)} = \text{Embed}_1(x_t) \in R^{n \times N} \quad (1)$$

In the anomaly attention module, a learnable Gaussian kernel is first used to calculate the prior relative to the relative time distance, and then input node feature  $X$  is projected onto the query ( $Q$ ), key ( $K$ ) and value ( $V$ ) matrix by linear projection. Assume that the model contains  $M$  layers with length  $n$  and input time series  $X \in R^{n \times N}$ . The anomaly attention in layer  $m$  is:

$$P^m = \text{Rescale}\left(\left[\frac{1}{\sqrt{2\pi\sigma_i}} \exp\left(-\frac{\|j-i\|^2}{2\sigma_i^2}\right)\right]\right) \quad (2)$$

$$Z^m = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right).V \quad (3)$$

where  $m \in \{1, \dots, M\}$  denotes the output of the  $m^{\text{th}}$  layer with  $N_{\text{model}}$  channels,  $Q, K, V \in R^{n \times N_{\text{model}}}$ , generates a prior association  $P^m \in R^{n \times n}$  based on the learning scale  $\sigma \in R^{n \times 1}$ , and the  $i^{\text{th}}$  element  $\sigma_i$  corresponds to the  $i^{\text{th}}$  point in time. Its associated weight with the  $j^{\text{th}}$  point is calculated by the Gaussian kernel  $G(\|j-i\|; \sigma_i) = \frac{1}{\sqrt{2\pi\sigma_i}} \exp\left(-\frac{\|j-i\|^2}{2\sigma_i^2}\right)$  w.r.t. the distance  $\|j-i\|$ , where  $i, j \in \{1, \dots, n\}$ . In addition,  $\text{Rescale}(\cdot)$  is used to transform the associated weights into discrete distributions  $P^m$  by partitioning rows.  $Z^m \in R^{n \times n}$  represents sequence association, and  $\text{Softmax}(\cdot)$  represents normalization of the attention force along the last dimension.

The module also uses a multi-head attention mechanism, and for  $H$  heads, the learning scale is  $\sigma \in R^{n \times H}$ .  $Q_h, K_h, V_h \in R^{n \times \frac{N_{\text{model}}}{H}}$  represents the query, key and value of the  $h^{\text{th}}$  head respectively. The outputs  $\left\{ Z_h^m \in R^{n \times \frac{N_{\text{model}}}{H}} \mid 1 \leq h \leq H \right\}$  from the multiple-head is then connected and the final result  $Z^m \in R^{n \times N_{\text{model}}}$  is obtained. and the symmetric  $KL$  difference between prior association and sequence association is used for anomaly differences, which represents the information gain between these two distributions [41]. Its formula is as follows:

$$Y = KL(P_i^m, : || Z_i^m, :) + KL(Z_i^m, : || P_i^m, :) \quad (4)$$

$$\text{Dis}(P, Z; X) = \left[ \frac{1}{M} \sum_{m=1}^M Y \right]_{i=1, \dots, N} \quad (5)$$

where  $KL(\cdot || \cdot)$  is the  $KL$  divergence calculated between two discrete distributions corresponding to each row of  $P^m$  and  $Z^m$ .  $\text{Dis}(P, Z; X) \in R^{n \times 1}$  is the point-by-point association difference of  $X$  with respect to a prior association  $P$  and sequence association  $Z$  from multiple layers.

### Dynamic graph learning

This paper focuses on the problem of system anomaly detection in the cloud environment, so we need to consider the dynamic changes and trends of KPIs. Therefore, we propose an effective dynamic relationship embedding strategy, considering the dynamic modeling and prediction of time sequence information of KPIs. As shown in Fig. 2(Part2), the time window is mainly used to process the data that is continuous in time, and the GNN model is applied to it for feature learning within each time window. Then the data of the whole KPIs is processed by sliding the time window to capture its dynamic evolution process, so as to better spatio-temporal prediction and effectively mining the dynamic changing KPIs features. By using a graph structure to learn the temporal correlation between cloud monitoring system data, we adopt directed graph connection features to show the dependencies between different measurement systems. The nodes of the graph represent the measurement systems, and the edges between nodes represent their dependencies. The layer adaptively learns the adjacency matrix  $A^{(i)} \in R^{N \times N}$  for sequences passing through the Transformer module to simulate potential relationships in a given time series sample  $x_t$ . The learned graph structure (adjacency matrix)  $A^{(i)}$  is defined as:

$$A^{(i)} = \text{Embed}_2(x_t) \quad (6)$$

We first calculate the similarity matrix between the sample time series, the formula is as follows:

$$C_{ij}^{(i)} = \frac{\exp(-f(\text{distance}(x_i, x_j)))}{\sum_{p=0}^n \exp(-f(\text{distance}(x_i, x_p)))} \quad (7)$$

where distance represents distance measurements, such as Euclidean distance, absolute distance, dynamic time warping, etc. The dynamic adjacency matrix  $A^{(i)}$  then be calculated as:

$$A^{(i)} = f(C^{(i)} W_1) \quad (8)$$

where  $W_1$  is the learnable model parameter and  $f$  is the activation function. In addition, in order to improve training efficiency, reduce noise effects, and make the model more robust, set the threshold value  $b$  to make the adjacency matrix sparse:

$$A^{(i)} = \begin{cases} A_{ij}^{(i)} & A_{ij}^{(i)} \geq b \\ 0 & A_{ij}^{(i)} < b \end{cases} \quad (9)$$

Finally, normalization is applied to  $A^{(i)}$ .

#### Repeated stack GNNs for spatio-temporal modeling

As shown in Fig. 2(Part3), the module uses 3 GNN layers ( $G1, G2, G3$ ) on the input graph (expressed as  $X^{(i)}, A^{(i)}$ ) to model the spatio-temporal relationship. The GNN layer can integrate spatial dependence and time patterns to embed the features of nodes, and transform the feature dimensions of nodes into decoding. The formula is as follows:

$$X_{\text{encode}}^{(i)}, A_{\text{encode}}^{(i)} = \text{GNNs}\left(X^{(i)}, A^{(i)}\right) \quad (10)$$

where  $X_{\text{encode}}^{(i)} \in R^{n \times N_{\text{model}}}$ ,  $A_{\text{encode}}^{(i)} \in R^{n \times n}$  is composed of graph neural network layer layer and batch normalization layer, and  $i = 1, 2, \dots, n$ . Then, during the pooling phase, GNN is trained using classical Diffpool and the soft cluster allocation of nodes at each layer of deep GNN is learned. As shown in Fig. 2(Part4), the overall transformation of a pooling layer is shown in equation (10) and the following two equations show the process in the Diffpool layer, where  $W_2 \in R^{N_{\text{model}} \times N_{\text{Diffpool}}}$  is the trainable parameter matrix representing the linear transformation and  $S^{(i)} \in R^{n_{\text{Diffpool}} \times n}$  is the distribution matrix representing the projection from the original node to the pooled node (cluster).  $X_{\text{Diffpool}}^{(i)} \in R^{n_{\text{Diffpool}} \times N_{\text{Diffpool}}}$  and  $A_{\text{Diffpool}}^{(i)} \in R^{n_{\text{Diffpool}} \times n_{\text{Diffpool}}}$  which has less nodes than the input graph, the parameter  $T$  represents inverting the matrix  $S^{(i)}$ .

$$X_{\text{Diffpool}}^{(i)} = \sigma\left(S^{(i)} X_{\text{encode}}^{(i)} W_2\right) \quad (11)$$

$$A_{\text{Diffpool}}^{(i)} = \sigma\left(S^{(i)} A_{\text{encode}}^{(i)} \left(S^{(i)}\right)^T\right) \quad (12)$$

We generate centroids  $K^{(i)} \in R^{N \times n_{\text{Diffpool}} \times N_{\text{model}}}$  based on the input graph and then compute and aggregate the relationship between every batch of centroids and the encoded graph for assignment matrix  $S^{(i)}$ . The relationship  $S_p^{(i)} \in R^{n_{\text{Diffpool}} \times n}$  ( $p = 1, 2, \dots, N$ ) and  $K_p^{(i)} \in R^{n_{\text{Diffpool}} \times N_{\text{model}}}$  ( $p = 1, 2, \dots, N$ ) can be computed. We use cosine similarity to evaluate the relationship between input node embeddings and centroids, followed by a row normalization deployed in the resulting assignment matrix.

$$S_p^{(i)} = \text{cosine}\left(K_p^{(i)}, X_{\text{encode}}^{(i)}\right) \quad (13)$$

$$S_p^{(i)} = \text{normalize}\left(S_p^{(i)}\right) \quad (14)$$

Then we concatenate  $S_p^{(i)}$  ( $p = 1, 2, \dots, N$ ) and perform a trainable weighted sum  $\Gamma_\phi$  to the concatenated matrix, leading to the final assignment matrix  $S^{(i)}$ .

$$S^{(i)} = \Gamma_\phi \left( \prod_{p=1}^{|N|} S_p^{(i)} \right) \quad (15)$$

After stacking several Diffpool, we can pool the original graph to a single node and get its graph-level representation vector  $x_{\text{final}}$ , as follows:

$$x_{\text{final}}^{(i)} = P_3\left(P_2\left(P_1\left(X_{\text{encode}}^{(i)}\right)\right)\right) \quad (16)$$

#### AR-MLP model ensembling

Since the monitoring data in the dynamic cloud contains nonlinear and linear features, we consider extracting the two features separately, and output the anomaly detection results through two lines into the AR-MLP model and the above nonlinear module in parallel, so as to better extract all the feature information of the key indicators. AutoRegressive (AR) model is a predictive model based on a time series, which assumes that the value of the current moment is only related to the value of several previous moments [42]. Adding AR model can maintain the linear relationship of KPIs, and the nonlinear feature extraction capability of MLP can enhance the feature extraction performance of AR model. At the same time, it is spatially sensitive, allowing communication between different spatial locations, and operating independently on each channel. Therefore, we propose a model combining AR model and MLP model to output anomaly detection results in parallel with the above nonlinear module, as shown in Fig. 2(Part5). AR-MLP can better adapt to complex anomaly detection tasks and further improve the accuracy and robustness of the model.

We first use the output of the above module to get the result  $x_{\text{final}} \in R^{n \times N_{\text{model}}}$ , and at the same time, the result obtained by the AR-MLP model is expressed as  $x_{\text{AM}} \in R^{n \times N_{\text{model}}}$ . Finally, the weighted sum of the two is used to get the final result  $\hat{x}_t$  of DGT-PF. The final anomaly score is as follows:

$$x^* = \left[ \|x_t - \hat{x}_t\|_2^2 \right]_{t=1, \dots, n} \quad (17)$$

$$\text{Score}(X) = \text{Softmax}(-\text{Dis}(P, Z; X)) \Delta(x^*) \quad (18)$$

where  $\Delta$  is element-by-element multiplication.

## Experiments and results

In this section we conduct synthetic experiments on 5 public datasets. The accuracy and timeliness of anomaly detection are evaluated by experiments. We first compare the detection performance of DGT-PF with baseline methods on 5 public datasets. Then we provide ablation experiments to analyze the importance of modules in DGT-PF. Then we plot two-part loss to analysis the convergence of DGT-PF model on 5 datasets during training. Finally, the sensitivity test of the experimental parameters is carried out.

### Datasets

Firstly, to deal with the complex and time-varying scenarios in the cloud environment, the experimental datasets should be more complex and diverse to be practical. Secondly, this paper is designed based on the KPIs data in cloud monitoring system, we mainly select public datasets related to cloud services to evaluate our propose DGT-PF model:

Server Machine Dataset (SMD) [43] is a 5-week real-time dataset of 28 cloud platform servers, collected from a large Internet company, it has 38 dimensions. Pool Server Metrics (PSM) [44] is a dataset collected from within eBay's multiple application server nodes and has 25 dimensions. Mars Science Laboratory Rover (MSL) and Soil Moisture Active and Passive Satellite (SMAP) [45], are spacecraft datasets provided by NASA in 55 and 25 dimensions, respectively. These contain remote sensing anomaly data obtained in the Spacecraft monitoring System Event Sudden Anomaly Emergency Anomaly (ISA) report. Secure Water Treatment (SWAT) [46] is obtained from 51 sensors of the critical infrastructure system under continuous operations. These three datasets are also used as supplementary datasets. Table 1 shows the statistics details for the experimental datasets. Figure 3 describes the one-dimensional feature representation of the 5 datasets. And it can be seen that there are significant differences in feature distribution among them, and also indicates that selected datasets have diverse distribution.

### Evaluation metrics

In order to measure the accuracy and effectiveness of various anomaly detection methods, precision, recall and *F1-score* are used as evaluation indicators to verify the anomaly detection performance of the model. Precision is about how much of the data detected as anomalies is true anomalies, while recall is about how much of the real anomaly data is detected as anomalies. The *F1-score* is a function of both precision and recall.

**Table 1** Details of the experimental baseline datasets. # App represents the application of the data sets, ab represents the truth abnormal proportion of the whole data sets, and d represents the dimension

Dataset	#App	#Train	#Test	ab(Truth)	d
SMD	Server	708405	708420	0.042	38
PSM	Server	132481	87841	0.278	25
MSL	Space	58317	73729	0.105	55
SMAP	Space	135183	427617	0.128	25
SWAT	Water	396000	449919	0.121	51

$$F1 = 2 * \frac{precision * recall}{precision + recall} \quad (19)$$

Therefore, we mainly focus on the *F1-score* for detection accuracy.

### Baseline model

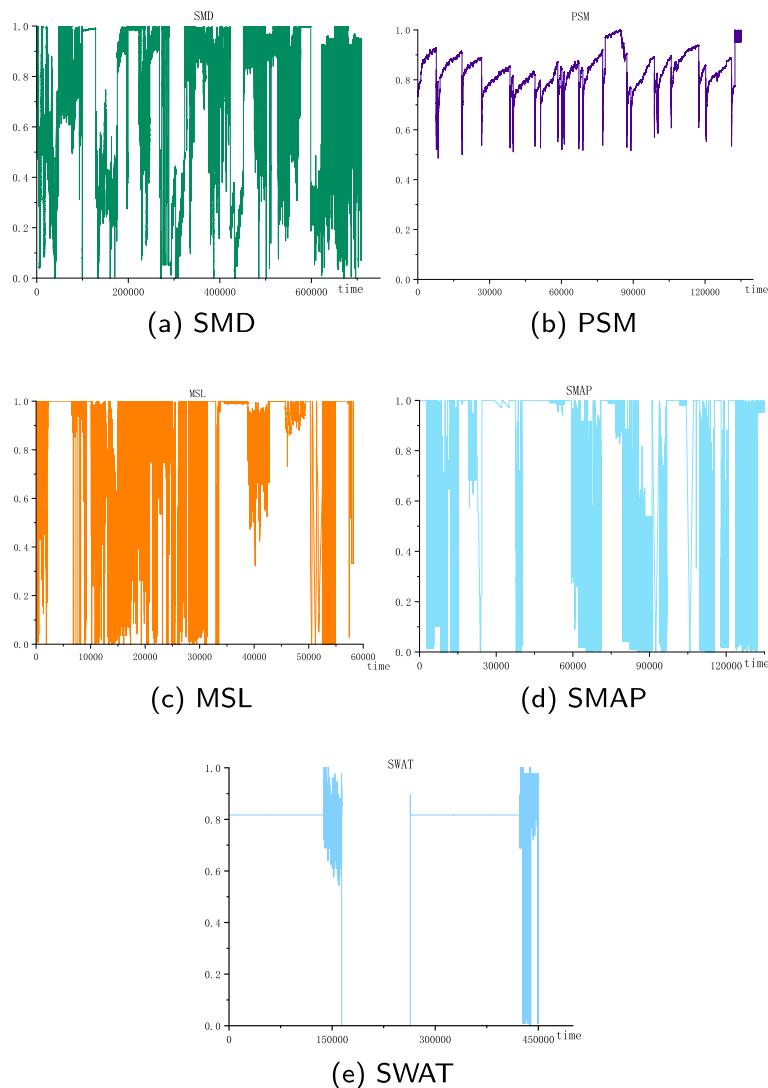
To fully demonstrate the strength of our model, we compare DGT-PF to the following 11 baselines, and we choose the first 3 statistical models and the last 8 deep learning models as baseline methods to prove that our model is due to linear and nonlinear models. Specific methods include 3 statistical methods **CBLOF**, **IsolationForest**, **ARIMA** and 3 classical deep learning models **ALAD**, **OCSVM**, **SO\_GAAL**, all of which are derived from pyod [47] except ARIMA model [12]. In addition, there are 5 recent deep learning models.

**USAD** [31]: UnSupervised Anomaly Detection on Multivariate Time Series, Combine autoencoder and adversarial training, the ordinary autoencoder is divided into one encoder and two decoders. One decoder produces fake data and trains the other decoder against it to improve its ability to recognize fake data.

**LSTM** [48]: Long Short-Term Memor, is a neural network model used to process sequence data. It captures long-term dependencies in sequence through gating mechanism and memory unit, and can solve problems such as gradient disappearance and gradient explosion.

**MTAD\_GAT** [49]: uses two parallel graph attention layers to learn timing and feature dependencies between multiple time series, and a reconstruction-based approach to learn normal data from historical data, in which (VAE) models are used to detect anomalies by reconstructing probabilities.

**Anomaly Transformer** [40]: consists of multiple layers overlapping anomaly attention modules and Feed Forward neural networks, in which anomaly attention has two branches: a prior association branch and a sequence association branch. Their correlation differences are then calculated to create the final outlier score.



**Fig. 3** The feature representation for one dimension of SMD, PSM, MSL, SMAP and SWAT datasets

**TRANAD** [50]: consists of Transformer and GAN, uses score based adaptation to achieve multi-modal feature extraction and stability through adversarial training, and introduces the idea of adversarial training.

#### Experimental setup

All experiments are implemented using a single NVIDIA GeForce 930MX GPU in python 3.8.15 and Pytorch 1.9.0+cu111. Experimental details follow Shen et al. [32]. All neural network models are optimized by using the Adam optimizer. We set the initial learning rate to  $10^{-4}$ , and we mark the point in time as an anomaly if the anomaly score is greater than some threshold  $\delta$ . A threshold of  $\delta$  is determined so that  $r$  proportion of the data in the validation dataset is marked as abnormal. Specifically,

non-overlapping sliding Windows are mainly used to obtain a set of sequences, and the size of sliding Windows is set to 100. Transformer with anomaly attention has 3 layers, 512 channels to set hidden state, and H-heads attention mechanism. GNN has 3 layers and the output dimension of the GNN layer is 128. To train the DGT-PF model, we set the epoch to 32, the batch size to 32,  $r$  to 1 %, and the training process to stop early in 10 periods.

#### Experimental result

##### Performance of anomaly detection

DGT-PF achieves a consistent up-to-date level across all baseline model tests. We present the results of DGT-PF compared with 11 baseline methods on 5 public datasets and analyze their detection performance. We

calculate the precision, recall, and *F1-score* for each model, and compare the results in Table 2. Table 2 shows that DGT-PF outperforms all other methods on these 5 datasets. The *F1-score* on SMD, PSM, MSL, SMAP and SWAT are 92.26%, 98.17%, 95.02%, 96.68% and 93.88% respectively, with an average increase of 23.29%, 21.84%, 21.24%, 20.47% and 20.95% compared with other methods. Some recall in the baseline model is higher than ours. For example, the recall of TRANAD in the 5 datasets has reached 99.99%. This may be due to the overfitting caused by the overly complex design of the model itself, or the lack of rigorous feature selection, resulting in the model's inability to adapt to the data well. The same is true for the USAD and MTAD\_GAT methods. However, precision has consistently achieved the highest levels across all benchmark models. Overall, it is compelling to consider the advantages of transformer combined with GNN in unsupervised anomaly detection. In addition, in terms of average ranking of *F1-score*, DGT-PF performed best across all 5 datasets. As you can see in Fig. 4, this means that this is important for real world applications.

#### Ablation study

The effects of dynamic graph embedding strategy, dynamic graph learning and AR-MLP model on DGT-PF are analyzed through ablation study. The experimental results are shown in Table 3, and the results of the two parts of ablation are shown separately in Figs. 5 and 6.

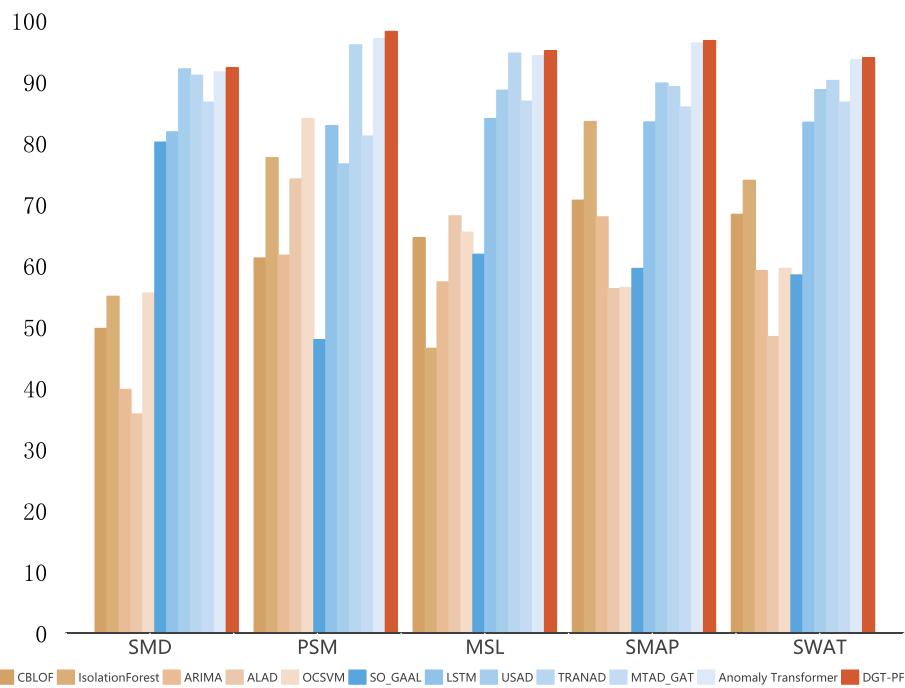
The influence of embedding strategies on anomaly detection using dynamic graph. Traditional deep learning methods mostly focused on static mining of features in KPIs, ignoring the dynamic evolution of KPIs data, and it

is necessary to consider the dynamic changes and trends of KPIs. So, we propose an effective dynamic relationship embedding strategy. It is based on learning graph structure. It can better extract dynamic features from the KPIs data. From DGT-PF-one and DGT-PF-corr in Fig. 5, we can see the influence of dynamic graph embedding strategy on the anomaly detection capability of our proposed model. DGT-PF-one is a DGT-PF model with an all-in-one adjacency matrix. DGT-PF-corr is a DGT-PF model with adjacency matrix of correlation coefficients. DGT-PF model has the dynamic adjacency matrix proposed by us. As can be seen in Table 3, DGT-PF-one has an average *F1-score* of 94.29 %, DGT-PF-corr has an average *F1-score* of 94.47 % and ours is 95.20 %. It can be seen that different adjacency matrices can be used in our DGT-PF model. Also, the all-one matrix performs slightly worse than the correlation matrix. Our dynamic matrix is the best.

Hybrid Transformer and GNN have higher average *F1-score* than other combinations. It can be seen from Fig. 6 that we verify the influence of this module on DGT-PF model by removing dynamic graph learning and AR-MLP model. DGT-PF-woAM indicates that AR-MLP components are removed from the DGT-PF model, and DGT-PF-woDG indicates that dynamic graph embedding and graph neural network segments are removed from the DGT-PF model. As can be seen in Table 3, With an average *F1-score* of 93.95 % for DGT-PF-woAM and 94.44 % for DGT-PF-woDG, the complete DGT-PF has the best results across different batch size, indicating that all components contribute to the effectiveness and robustness of the entire model. The performance of

**Table 2** DGT-PF compares the anomaly detection performance of 11 detection methods on 4 public datasets. P, R, and *F1-score* stand for Precision, Recall, and *F1-score* (expressed in %). For all 3 metrics, higher values indicate better performance, with the best *f1-score* highlighted in bold

Method	SMD			PSM			MSL			SMAP			SWAT		
	P	R	F1												
CBLOF	57.20	76.12	49.68	82.91	85.25	61.18	48.92	96.33	64.48	57.29	90.49	70.61	54.29	89.61	65.18
IsolationForest	62.31	73.29	54.94	66.94	92.54	77.55	52.39	89.07	46.43	76.09	92.45	83.48	74.96	91.25	84.15
ARIMA	45.44	90.49	39.72	80.56	60.25	61.64	67.39	79.67	57.30	74.68	81.42	67.90	62.05	74.92	70.11
ALAD	51.99	79.67	35.73	61.15	93.95	74.08	52.58	95.31	68.06	53.34	59.07	56.17	50.49	64.28	51.28
OCSVM	34.34	76.68	55.45	78.52	90.21	83.96	59.96	90.11	65.41	53.91	59.07	56.37	49.73	57.26	54.39
SO_GAAL	78.54	79.66	80.10	46.25	49.59	47.86	89.94	90.34	61.78	67.28	53.30	59.48	60.63	55.07	53.82
LSTM	78.55	85.28	81.78	76.93	89.64	82.80	85.45	82.50	83.95	89.41	78.13	83.39	82.15	73.18	79.64
USAD	80.59	99.99	92.05	79.62	97.29	76.53	97.95	99.12	88.57	81.39	96.27	89.74	80.79	93.82	79.87
TRANAD	88.72	99.99	91.02	81.50	98.99	95.97	96.15	99.99	94.64	80.43	98.72	89.15	81.28	97.72	86.19
MTAD_GAT	76.57	77.37	86.63	76.28	98.33	81.09	76.23	98.24	86.78	75.16	99.91	85.83	73.58	99.99	85.07
Anomaly Transformer	89.32	93.94	91.57	95.20	96.89	97.01	98.46	98.33	94.19	93.54	98.18	96.27	90.31	97.96	92.58
<b>DGT-PF</b>	<b>89.59</b>	<b>95.85</b>	<b>92.26</b>	<b>97.66</b>	<b>98.72</b>	<b>98.17</b>	<b>98.84</b>	<b>97.60</b>	<b>95.02</b>	<b>94.33</b>	<b>98.90</b>	<b>96.68</b>	<b>91.18</b>	<b>96.41</b>	<b>93.88</b>



**Fig. 4** F1-score of DGT-PF and all baseline models

**Table 3** Ablation results of DGT-PF (F1-score (%)). Where, DGT-PF-one, DGT-PF-corr, DGformer-woAM and DGT-PF-woDG represent neural network modules with full adjacency matrix, adjacency matrix with correlation coefficient, no AR-MLP module and no dynamic embedded graph, highest scores are highlighted in bold

Method	SMD	PSM	MSL	SMAP	SWAT	Avg.F1(as%)
DGT-PF-one	92.10	97.68	92.40	96.19	93.08	94.29
DGT-PF-corr	91.69	97.65	92.69	96.50	93.83	94.47
DGT-PF-woAM	91.01	97.29	92.35	95.48	93.62	93.95
DGT-PF-woDG	91.82	97.33	92.82	96.47	93.76	94.44
<b>DGT-PF</b>	<b>92.26</b>	<b>98.17</b>	<b>95.02</b>	<b>96.68</b>	<b>93.88</b>	<b>95.20</b>

DGT-PF-woDG has decreased, which indicates that adding GNN to dynamically capture temporal features can improve the timeliness of the model. DGT-PF-woAM's performance degradation is even more pronounced, indicating that AR-MLP components play a crucial role. The reason is that AR model can be used to maintain the linear relationship of KPIs, and the nonlinear feature extraction capability of MLP model can enhance the feature extraction performance of AR model. Meanwhile, it is spatially sensitive and can better adapt the complex anomaly detection task. However, if our framework only has nonlinear modules, it can not extract all the characteristic information of KPIs well and is not sensitive to the input. so it is necessary to add the AR-MLP model [51]. The experimental results show that both dynamic

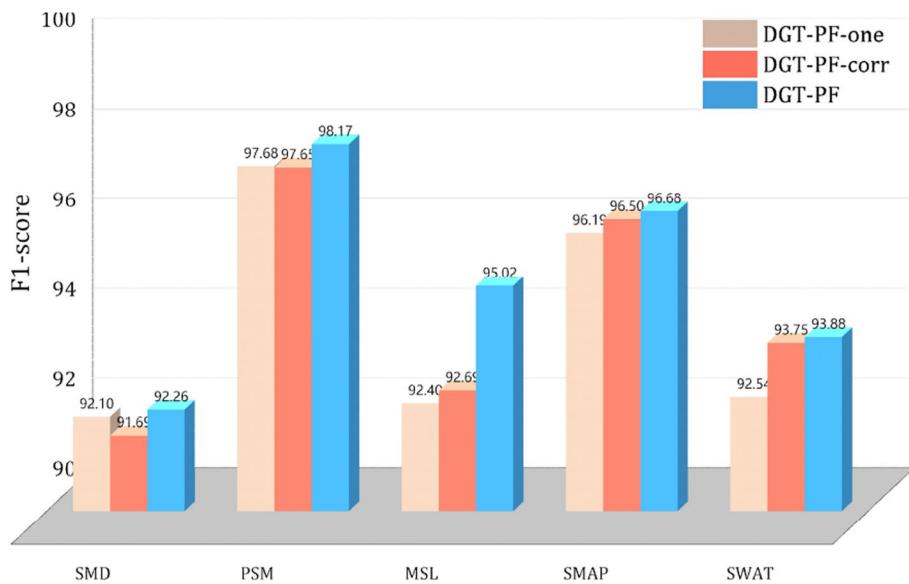
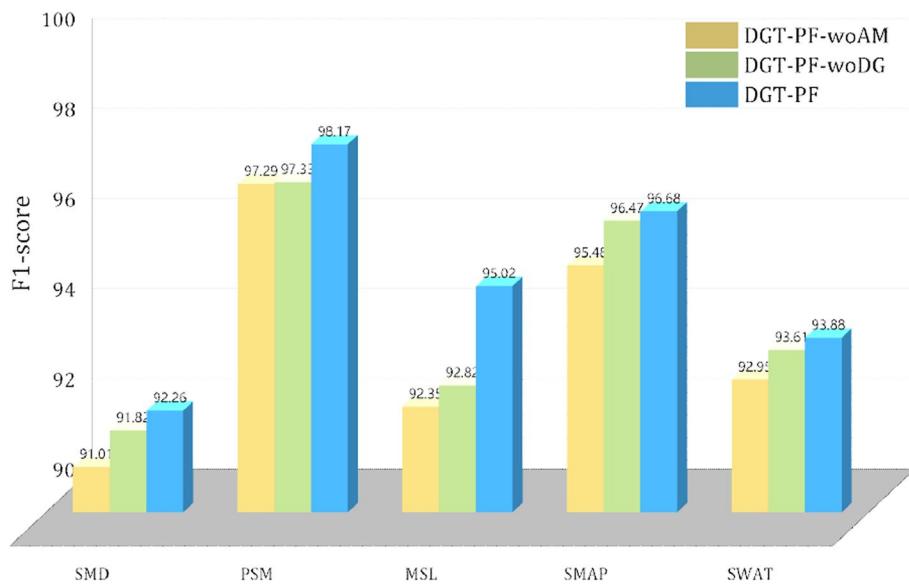
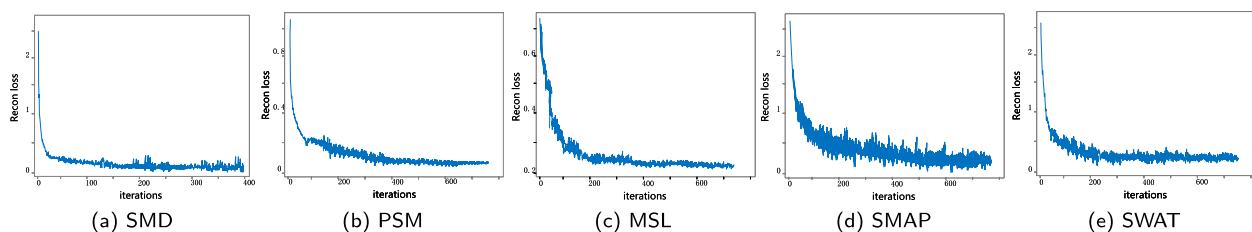
graph learning and AR-MLP model improve the anomaly detection performance of the model.

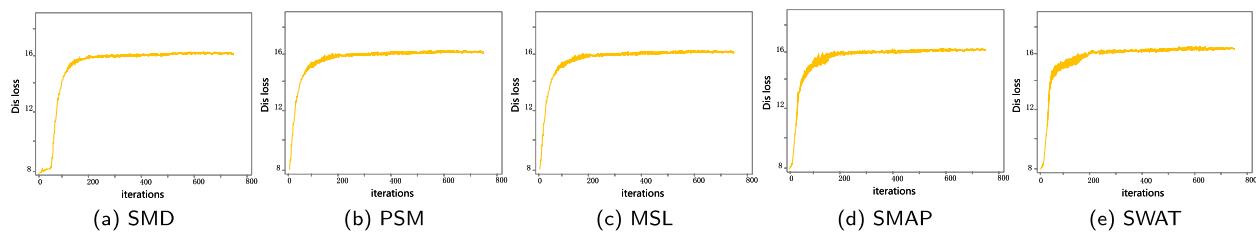
#### Convergence analysis

We plot the two-part loss of the model during training, as shown in Figs. 7 and 8. The two-part loss can be converged within acceptable iterations on all five real-world datasets, and this good convergence is necessary for the optimization of the model.

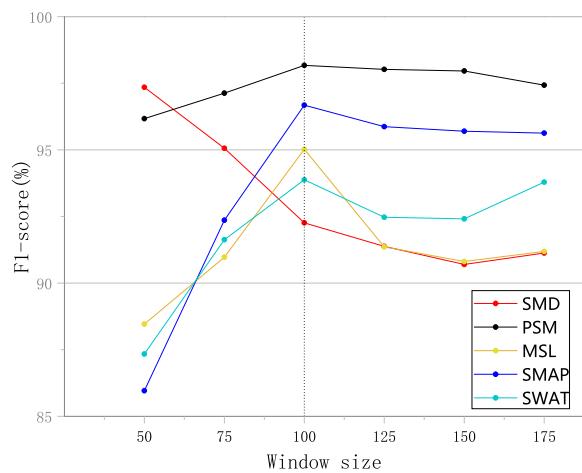
#### Effect of parameters

We study the effects of window size and batch size on the model performance. In the experiment, the window size is 100 and batch size is 32, which takes into account the time information, memory and computing

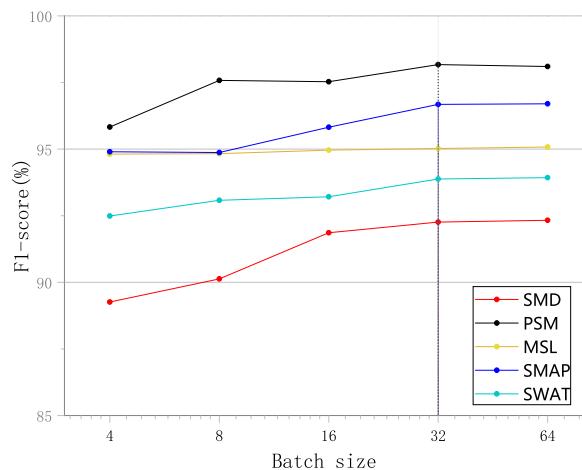
**Fig. 5** Ablation results of dynamic graph embedding strategy**Fig. 6** Ablation results of Hybrid Transformer and GNN**Fig. 7** Change curve of  $\|x_t - \hat{x}_t\|_2^2$  in five datasets during the training process



**Fig. 8** Change curve of  $\|Dis(P, S; X)\|_1$  in five datasets during the training process



**Fig. 9** Parameter sensitivity for sliding window size



**Fig. 10** Parameter sensitivity for batch size

efficiency. Figure 9 shows the effects of different window size choices on *F1-score*. Figure 10 shows the influence of batch size of different sizes on *F1-score*. In the task of anomaly detection, window size is an important parameter, which determines how long the scope of the anomaly detection. Different window sizes may have an impact on the anomaly detection results, as smaller

**Table 4** DGT-PF model selects different layers of performance with anomaly attention. Highest scores are highlighted in bold

Dataset	SMD	PSM	MSL	SMAP	SWAT
layer 1	89.95	97.90	93.19	96.33	96.26
layer 2	91.25	95.14	93.66	96.19	93.39
layer 3	90.88	96.93	94.90	96.08	93.41
Multiple-layer	<b>92.26</b>	<b>98.17</b>	<b>95.02</b>	<b>96.68</b>	<b>93.88</b>

windows may result in higher false positives, while larger windows may result in higher false positives. Larger window sizes represent larger memory overhead and smaller slippage numbers, and our model is relatively stable for window sizes across a wide range of datasets. In particular, only performance is considered, and its relationship to window size can be determined by the data patterns. Therefore, the accuracy of anomaly detection model can be improved by selecting a reasonable window size. For example, when the window size of MSL, SMAP, PSM and SWAT datasets is 100, the *F1-score* of anomaly detection is the highest, and when the window size of SMD dataset is 50, the *F1-score* of anomaly detection is the highest. Considering the performance of the model comprehensively, we chose the window size of 100 in the experiment. In addition, we find in Fig. 10 that the larger batch size is, the better it is. However, with the increase of batch size, the performance requirements of computers will also increase. Therefore, based on the above considerations, we choose batch size to be 32.

We investigate the performance of the model using different anomaly attention layers. We average the outlier scores from multiple layers to get the final result. As shown in Table 4, the best multi-layer design is achieved. This also proves the value of multi-layer quantization. Also, we investigate the model's performance and efficiency. This was under different choices of layer number  $M$  and hidden channel  $D$ . The results are in Tables 5 and 6. Generally, the use of more computation (e.g. increasing model size, datasets size, or training steps) can obtain better results but with larger memory and computation costs. Our goal is to achieve

**Table 5** Model performance under different choices of the number of layers M. Highest scores are highlighted in bold

Dataset	SMD	PSM	MSL	SMAP	SWAT
M=1	91.35	98.02	<b>95.46</b>	96.24	92.76
M=2	91.71	98.13	94.88	96.21	92.91
M=3	92.26	<b>98.17</b>	95.02	<b>96.68</b>	<b>93.88</b>
M=4	<b>92.45</b>	98.15	95.03	96.33	92.95

high accuracy and real-time performance. We evaluate the model performance under different choices of hidden channel numbers. The detection time is the average running time to determine whether the incoming sample is abnormal or not. As shown in Table 6, the detection of the model is about 2 milliseconds. GPU memory utilization for training is also within acceptable range of 5.3GB. In general, increasing the size of the model yields better results, but requires more memory and computational costs. With this in mind, we choose the number of layers M to be 3 and the number of hidden channels D to be 512. The results above validate our model's sensitivity. This quality is key for applications.

### Case study: visualizations

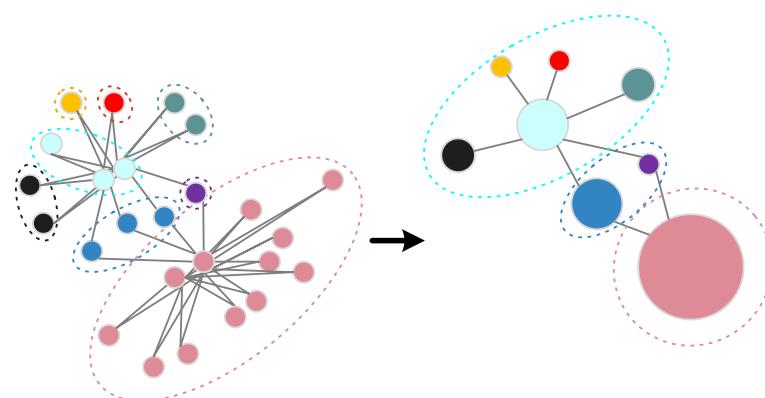
In this section, we'll use the PSM dataset to visualize the Diffpooling process. Figure 11 shows a visualization of node assignments in the first and second layers on the graph built from the PSM dataset. We use the same color to represent nodes in the same cluster. It is worth mentioning that the allocation matrix may not assign nodes to a specific cluster. The column corresponding to the unused cluster has a lower value for all nodes. For example, we set the expected number of clusters in the first layer to be 15 (more than 8). But, in fact, we get 8 clusters. This reminds us that even if the expected number of clusters is defined in advance, our model automatically performs clustering to get the best rough result for this graph. These features can be adjusted for different inputs, so they have a strong ability to generalize.

### Experimental summary

Our model DGT-PF is compared with 11 baseline methods on 5 public datasets, and its *F1-score* reaches state-of-the-art results with the highest reaching 98.17%, and converges within the acceptable iteration range, which indicates the validity of our model. In addition, the ablation experiments prove that all five parts of our model play a certain role, and we also do a series of parameter

**Table 6** Model performance under different choices of the number of hidden channels D. Mem is GPU memory utilization for training, time is the average running time to determine whether the incoming sample is abnormal or not. Highest scores are highlighted in bold

Dataset	SMD	PSM	MSL	SMAP	SWAT	Mem(GB)	time(ms)
D=256	91.72	98.05	<b>95.45</b>	96.31	92.44	4.7	2.5
D=512	92.26	<b>98.17</b>	95.02	<b>96.68</b>	<b>93.88</b>	5.3	2.0
D=1024	<b>97.26</b>	97.29	94.37	96.05	92.99	6.8	1.8



**Fig. 11** Visualize the Diffpool process, using a sample graph from the PSM dataset that has 25 variables, so the original graph has 25 nodes. The nodes at Layer 2 correspond to the cluster at Layer 1. We use the same colors represent nodes in the same cluster, and the dashed lines represent different clusters

sensitivity experiments to optimize our model and make the model achieve the best results.

## Conclusion

Anomaly detection is the key to ensure the reliability of cloud monitoring system. In order to detect system anomalies in cloud infrastructures, we propose a novel Dynamic Graph Transformer based Parallel Framework(DGT-PF). In order to improve the accuracy and timeliness of the anomaly detection model, this paper proposes several innovations, which are related as follows:

In this paper, we propose a new deep learning framework, a novel Dynamic Graph Transformer based Parallel Framework (DGT-PF) for efficiently detect system anomalies in cloud infrastructures. It overcomes the defects of traditional Transformer and GNN, and proposes an effective model that uses Transformer with anomaly attention mechanism to obtain GNN parameters and dynamically capture spatio-temporal features through graph structure learning. Finally, a parallel AR-MLP model is obtained with strong interpretability. DGT-PF has achieved state-of-the-art results on a detailed set of empirical studies. In the experiment, we use 5 public datasets to evaluate our model. In terms of *F1-score*, DGT-PF outperform all baseline methods on 5 public datasets.

Our DGT-PF model has been proved to be effective in detecting system anomalies in cloud infrastructure by experiments. However, our model have also some limitations. First of all, we only do anomaly detection experiments in this paper without anomaly diagnosis and root cause localization, and the effective fault diagnosis needs to add these parts of content, only doing the anomaly detection is not enough, we will intensify these parts of our work in the future. Secondly, the basic model of this paper is GNN, Transformer and attention mechanism. Compared with other models, this model is more complex, which may lead to the increase of time complexity and parameters, thus affecting model training. For some high real-time applications that require microsecond level detection time, the real-time performance of the proposed model is still inadequate and needs further improvement.

In future work, we also consider three aspects to improve the DGT-PF model. Firstly, we consider adding experiments for anomaly diagnosis and root cause localization in the next part of our work. Secondly, to reduce the training time and improve the real-time performance in the optimization of neural network is also one of the focuses of our research. Finally, we hope to explore and design a more powerful graph Transformer that can be

incorporated into our DGT-PF model for better presentation performance.

## Acknowledgements

This research is partially supported by the National Natural Science Foundation of China under Grant No.62376043 and Science and Technology Program of Sichuan Province under Grant No.2020JDRC0067, No.2023JDRC0087, and No.24NSFTD0025.

## Authors' contributions

Hongxia He put forward the main ideas, designed and carried out the experiment, and wrote the manuscript. Xi Li guided research and provided comments for writing. Peng Chen guided ideas and experiments and provided comments for writing. Juan Chen, Ming Liu, and Lei Wu assisted in the experiment. All authors reviewed and approved the final manuscript.

## Funding

This research is supported by the National Natural Science Foundation under Grant No.62376043 and Science and Technology Program of Sichuan Province under Grant No.2024ZHCG0016, No.2023JDRC0087 and No.2024NSFTD0008.

## Availability of data and materials

No datasets were generated or analysed during the current study.

## Declarations

### Competing interests

The authors declare no competing interests.

Received: 13 December 2023 Accepted: 24 May 2024

Published online: 04 June 2024

## References

- Cid-Fuentes JA, Szabo C, Falkner K (2018) Adaptive performance anomaly detection in distributed systems using online svms. IEEE Trans Dependable Secure Comput 17(5):928–941
- Xu H, Chen W, Zhao N, Li Z, Bu J, Li Z, Liu Y, Zhao Y, Pei D, Feng Y, Chen J, Wang Z, Qiao H (2018) Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. The Web Conference 2018-Proceedings of the World Wide Web Conference, WWW 2018, France, p 187–196. <https://doi.org/10.1145/3178876.3185996>.
- Long T, Chen P, Xia Y, Ma Y, Sun X, Zhao J, Lyu Y (2024) A deep deterministic policy gradient-based method for enforcing service fault-tolerance in mec. Chin J Electron 34:1–11
- Li Z, Lu Q, Zhu L, Xu X, Liu Y, Zhang W (2018) An empirical study of cloud api issues. IEEE Cloud Comput 5(2):58–72
- Chen J, Chen P, Niu X, Wu Z, Xiong L, Shi C (2022) Task offloading in hybrid-decision-based multi-cloud computing network: a cooperative multi-agent deep reinforcement learning. J Cloud Comput 11(1):1–17
- Chen P, Xia Y, Pang S, Li J (2015) A probabilistic model for performance analysis of cloud infrastructures. Concurr Comput Pract Experience 27(17):4784–4796
- Yu M, Zhang X (2023) Anomaly detection for cloud systems with dynamic spatiotemporal learning. Intell Autom Soft Comput 37(2). <https://doi.org/10.32604/iasc.2023.038798>
- Chang CI (2022) Target-to-anomaly conversion for hyperspectral anomaly detection. IEEE Trans Geosci Remote Sens 60:1–28
- Ba NG, Selvakumar S (2020) Anomaly detection framework for internet of things traffic using vector convolutional deep learning approach in fog environment. Futur Gener Comput Syst 113:255–265
- Erhan L, Ndubaku M, Di Mauro M, Song W, Chen M, Fortino G, Bagdasar O, Liotta A (2021) Smart anomaly detection in sensor systems: A multi-perspective review. Inf Fusion 67:64–79
- Ives AR, Abbott KC, Ziebarth NL (2010) Analysis of ecological time series with arma (p, q) models. Ecology 91(3):858–871

12. Shumway RH, Stoffer DS, Shumway RH, Stoffer DS (2017) Arima models. Time series analysis and its applications: with R examples. pp 75–163
13. Bauwens L, Laurent S, Rombouts JV (2006) Multivariate garch models: a survey. *J Appl Econ* 21(1):79–109
14. Xin R, Chen P, Zhao Z (2023) Causalrca: Causal inference based precise fine-grained root cause localization for microservice applications. *J Syst Softw* 203:111724
15. Sharif Razavian A, Azizpour H, Sullivan J, Carlsson S (2014) Cnn features off-the-shelf: an astounding baseline for recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, United states. p 512–519. <https://doi.org/10.1109/CVPRW.2014.131>
16. Yu Y, Si X, Hu C, Zhang J (2019) A review of recurrent neural networks: Lstm cells and network architectures. *Neural Comput* 31(7):1235–1270
17. Liu H, Xin R, Chen P, Gao H, Grossi P, Zhao Z (2023) Robust-pac time-critical workflow offloading in edge-to-cloud continuum among heterogeneous resources. *J Cloud Comput* 12(1):1–17
18. Zhang ML, Zhou ZH (2007) Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recog* 40(7):2038–2048
19. Wu Y, Dai HN, Tang H (2021) Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet Things J* 9(12):9214–9231
20. Zhang R, Chen J, Song Y, Shan W, Chen P, Xia Y (2023) An effective transformation-encoding-attention framework for multivariate time series anomaly detection in iot environment. *Mobile Networks Appl* 1–13. <https://doi.org/10.1007/s11036-023-02204-9>
21. Ibdunmoye O, Hernández-Rodriguez F, Elmroth E (2015) Performance anomaly detection and bottleneck identification. *ACM Comput Surv (CSUR)* 48(1):1–35
22. Chen Y, Wang S, Zhao Q, Sun G (2021) Detection of multivariate geochemical anomalies using the bat-optimized isolation forest and bat-optimized elliptic envelope models. *J Earth Sci* 32(2):415–426
23. Zhang Z, Xu J, Zhou X (2019) Mapping time series into complex networks based on equal probability division. *AIP Adv* 9(1):015017
24. Song Y, Xin R, Chen P, Zhang R, Chen J, Zhao Z (2024) Autonomous selection of the fault classification models for diagnosing microservice applications. *Futur Gener Comput Syst* 153:326–339
25. Lai G, Chang WC, Yang Y, Liu H (2018) Modeling long-and short-term temporal patterns with deep neural networks. In: The 41st international ACM SIGIR conference on research & development in information retrieval. pp 95–104. <https://doi.org/10.1145/3209978.3210006>
26. Park D, Hoshi Y, Kemp CC (2018) A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robot Autom Lett* 3(3):1544–1551
27. Niu Z, Yu K, Wu X (2020) Lstm-based vae-gan for time-series anomaly detection. *Sensors* 20(13):3738
28. Zhang X, Gao Y, Lin J, Lu CT (2020) Tapnet: Multivariate time series classification with attentional prototypical network. AAAI 2020-34th AAAI Conference on Artificial Intelligence, United states, vol 34. p 6845–6852. <https://doi.org/10.1609/aaai.v34i04.6165>
29. Grisoni F, Moret M, Lingwood R, Schneider G (2020) Bidirectional molecule generation with recurrent neural networks. *J Chem Inf Model* 60(3):1175–1183
30. Xin R, Liu H, Chen P, Zhao Z (2023) Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework. *J Cloud Comput* 12(1):1–16
31. Audibert J, Michiardi P, Guyard F, Marti S, Zuluaga MA (2020) Usad: Unsupervised anomaly detection on multivariate time series. Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, United states, p 3395–3404. <https://doi.org/10.1145/3394486.3403392>
32. Shen L, Li Z, Kwok J (2020) Timeseries anomaly detection using temporal hierarchical one-class network. *Adv Neural Inf Process Syst* 33:13016–13026
33. Li D, Chen D, Jin B, Shi L, Goh J, Ng SK (2019) Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In: International conference on artificial neural networks. Springer, Cham, p 703–716. [https://doi.org/10.1007/978-3-030-30490-4\\_56](https://doi.org/10.1007/978-3-030-30490-4_56)
34. Qi S, Chen J, Chen P, Wen P, Niu X, Xu L (2023) An efficient gan-based predictive framework for multivariate time series anomaly prediction in cloud data centers. *J Supercomput* 1–26. <https://doi.org/10.1007/s11227-023-05534-3>
35. Chen P, Liu H, Xin R, Carval T, Zhao J, Xia Y, Zhao Z (2022) Effectively detecting operational anomalies in large-scale iot data infrastructures by using a gan-based predictive model. *Comput J* 65(11):2909–2925
36. Dwivedi VP, Bresson X (2020) A generalization of transformer networks to graphs. arXiv preprint arXiv:201209699. <https://arxiv.org/abs/2012.09699>
37. Kreuzer D, Beaini D, Hamilton W, Létourneau V, Tossou P (2021) Rethinking graph transformers with spectral attention. *Adv Neural Inf Process Syst* 34:21618–21629
38. Yun S, Jeong M, Kim R, Kang J, Kim HJ (2019) Graph transformer networks. *Adv Neural Inf Process Syst* 32:11983–11993
39. Zhang S, Tong H, Xu J, Maciejewski R (2019) Graph convolutional networks: a comprehensive review. *Comput Soc Networks* 6(1):1–23
40. Xu J, Wu H, Wang J, Long M (2022) Anomaly transformer: Time series anomaly detection with association discrepancy. ICLR 2022-10th International Conference on Learning Representations
41. Zhou F, Du X, Li W, Lu Z, Wu J (2022) Nidd: An intelligent network intrusion detection model for nursing homes. *J Cloud Comput* 11(1):1–17
42. Song Y, Xin R, Chen P, Zhang R, Chen J, Zhao Z (2023) Identifying performance anomalies in fluctuating cloud environments: A robust correlative-gnn-based explainable approach. *Futur Gener Comput Syst* 145:77–86
43. Su Y, Zhao Y, Niu C, Liu R, Sun W, Pei D (2019) Robust anomaly detection for multivariate time series through stochastic recurrent neural network. Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, United states, p 2828–2837. <https://doi.org/10.1145/3292500.3330672>
44. Abdulaal A, Liu Z, Lancewicki T (2021) Practical approach to asynchronous multivariate time series anomaly detection and localization. Proceedings of the 27th ACM SIGKDD international conference on knowledge discovery & data mining, Singapore, p 2485–2494. <https://doi.org/10.1145/3447548.3467174>
45. Hundman K, Constantinou V, Laporte C, Colwell I, Soderstrom T (2018) Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. pp 387–395
46. Mathur AP, Tippenhauer NO (2016) Swat: a water treatment testbed for research and training on ics security. pp 31–36. <https://doi.org/10.1109/CyWater.2016.7469060>
47. Zhao Y, Nasrullah Z, Li Z (2019) Pyod: A python toolbox for scalable outlier detection. *J Mach Learn Res (JMLR)* 20(96):1–7
48. Sundermeyer M, Schlüter R, Ney H (2015) From feedforward to recurrent LSTM neural networks for language modeling. *IEEE Transactions on Audio, Speech and Language Processing, Institute of Electrical and Electronics Engineers Inc.*, United States, 23(3):517–529. <https://doi.org/10.1109/TASLP.2015.2400218>
49. Zhao H, Wang Y, Duan J, Huang C, Cao D, Tong Y, Xu B, Bai J, Tong J, Zhang Q (2020) Multivariate time-series anomaly detection via graph attention network. In: 2020 IEEE International Conference on Data Mining (ICDM). Sorrento, Italy, p 841–850. <https://doi.org/10.1109/ICDM50108.2020.00093>
50. Tuli S, Casale G, Jennings NR (2022) Tranad: Deep transformer networks for anomaly detection in multivariate time series data. arXiv preprint arXiv:220107284. <https://doi.org/10.48550/arXiv.2201.07284>
51. Mohammadi B, Mehdizadeh S, Ahmadi F, Lien NTT, Linh NTT, Pham QB (2021) Developing hybrid time series and artificial intelligence models for estimating air temperatures. *Stoch Env Res Risk A* 35:1189–1204

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.