DGFormer: An Effective Dynamic Graph Transformer Based Anomaly Detection Model for IoT Time Series

Hongxia He, Xi Li(✉), Peng Chen(✉), Juan Chen, Weijian Song, and Qinghui Xi

School of Computer and Software Engineering, Xihua University, Chengdu, China
lixi13@gmail.com, chenpeng@mail.xhu.edu.cn

Abstract. Internet of Things (IoT) is network based on information carriers such as the Internet and traditional telecommunications networks, so that all ordinary physical objects that can be independently addressed can be interconnected. In the face of the IoT produces a large of time series data, which is very necessary to detect anomaly data. Transformer has proven to be a powerful tool in several areas, but still has some limitations, such as the prediction accuracy is not high enough. As the dominant trend of multivariate time series in different scenarios becomes increasingly evident, it is particularly important to accurately capture the spatio-temporal features between them. To address these issues, we propose Dynamic Graph transFormer (DGFormer), an effective Dynamic Graph Transformer based Anomaly Detection Model for IoT Time Series. We first use Transformer with anomaly attention mechanism to extract time features. Then, a dynamic relationship embedding strategy is proposed to capture spatio-temporal features dynamically and learn the adjacency matrix adaptively. Besides, each layer of GNN is soft clustered by Diffpooling. Finally, in order to further improve the detection performance of model, we integrate the traditional autoregressive linear model with the nonlinear neural network in parallel. The experimental results show that the proposed model achieves the highest F1-score on three public IoT datasets, and the F1-score is improved by 19.3% on average.

Keywords: Internet of Things · Anomaly detection · Time series · Transformer · Graph neural network

1 Introduction

With the rapid growth of interconnecting devices and sensors in information physical systems such as autonomous vehicle, intelligent buildings, water treatment and distribution plants [1], the emergence of the IoT further promotes the application of network physical systems to various tasks, and it is increasingly necessary to monitor these devices from attacks, which is particularly important for key infrastructure such as power grids and communication networks [2]. The IoT can use blockchain, edge computing, deep learning and other methods to achieve target monitoring, positioning, recognition, user

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 562, pp. 173–188, 2024.
https://doi.org/10.1007/978-3-031-54528-3_10

privacy safe storage and other functions [3]. Nowadays, the amount of data is increasing exponentially. Faced with these massive amounts of data, the improvement of the IoT can greatly promote the future application and development of wireless sensor networks [4]. Therefore, it is very necessary to perform anomaly detection on the data in monitoring, that is, by analyzing the anomaly patterns of the target monitoring data to detect the anomaly behavior of the monitoring object. How to accurately and efficiently perform anomaly detection has also become a hot issue in the field of IoT security [5].

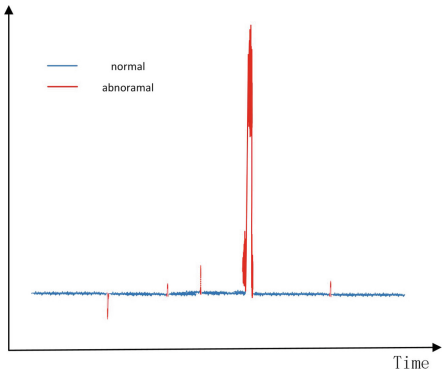

Fig. 1. Typical IoT anomaly detection, with red representing anomaly data and blue representing normal data.

Anomaly detection, a.k.a. outlier detection, has a wide range of applications in many fields, including network security, medicine, machine vision, statistics, credit card theft, and large expenditures [6]. The anomaly detection algorithm mainly learns to detect anomalies or emit danger signals when anomaly events occur by observing unlabeled datasets of normal events. As shown in Fig. 1, in real-world IoT environment, detecting anomalies from IoT sensors is essentially multivariate time series anomaly detection, as real-time IoT data collected from various sensors are processed and stored in multivariate time series. Due to the lack of anomaly labels in the data collected by sensors, and the unpredictable and diverse nature of anomalies, anomaly detection is often seen as an unsupervised learning problem. Based on above, we will focus on time series anomaly detection in an unsupervised environment.

Before machine learning, classic time series algorithms are generally used to statistical model. However, nowadays, the multivariate time series generated by cyber physics systems are highly complex and inherently nonlinear. These methods only model the relationships between sensors and can only capture linear relationships [7]. Therefore, in recent years, researchers have utilized deep learning-based techniques for anomaly detection in high-dimensional data, thereby to develop more intelligent and cost-effective methods to identify anomalies. For example, unsupervised anomaly detection algorithms such as OCSVM [34], ridge regression [8], RNN [9] and LSTM [37] are used to build models. These deep learning methods can capture long-term dependencies in time series data and are suitable for processing time series anomaly data. However, due to the relatively complex internal structure of these models, the training efficiency is very low,

and the calculation amount is also large and time-consuming. Moreover, they have weak distribution assumptions about anomaly data and easy to be affected by normal data, which may lead to the problem of false detection and missing detection. In particular, the training accuracy of these models is not high enough, and the optimization of model performance needs to be strengthened. With the further development of deep learning technology in recent years, Transformer and GNN have achieved a series of important results in many areas.

Nowadays, New applications of the Transformer self-attention network have been recognized, published, and successfully used in research areas such as computer vision, image processing, and natural language processing. Its structure includes self-attention mechanism, location coding, Add&Normalize, fully connected layer Feed Forward and other modules. These modules cooperate with each other, can achieve fast parallel operation by using self-attention mechanism, and can better process time series data and extract time features. By learning the spatio-temporal representation of graph structure, GNN can consider both spatial and temporal dimensions of data. Based on above, we consider combining Transformer with GNN for time series anomaly detection. GNN have been extensively studied in recent years and have successfully completed difficult machine learning tasks such as node classification, link prediction, and graph classification, due to high expressiveness through message passing in effective learning graph representation [10]. GNN can learn both temporal and spatial dependencies and display high-dimensional data with complex relationships, and can be widely used in the modeling of complex systems. Based on this, GNN can be a promising way to model multivariate time series data. It takes graph structure data as input. If it is applied to anomaly detection of multivariate time series, the complex relationships in time series need to be converted into graphs and learned together with the model.

We propose Dynamic Graph transFormer (DGFormer), an effective Dynamic Graph Transformer based Anomaly Detection Model for IoT Time Series. First, we design a novel anomaly attention mechanism and construct an effective Transform model to extract time features of time series data. To compute the association discrepancy, we renovate the self-attention mechanism to anomaly attention, which contains a two-branch structure to model the prior association and series association for each time point respectively. The prior association employs a learnable Gaussian kernel to present the adjacent concentration inductive bias at each time point, while a series association corresponds to the self-attention weights learned from the original sequence. The distance between the two associations at each time point is then calculated to quantify the anomaly criteria. Besides, a dynamic relational embedding strategy is proposed to capture the spatio-temporal features of the sequences to improve the timeliness of the model. And then, GNN model is used to realize the spatio-temporal dependence relationship of time series and make spatio-temporal prediction better. Finally, in order to further improve the detection performance of model, and our data have both linear and nonlinear feature parts, we consider adding an autoregressive linear model AR to extract its linear part to supplement the overall performance other than nonlinear. Experiments on real time series datasets have been proved the accuracy and effectiveness of the proposed method. These contributions are summarized as follows:

- In order to learn the spatio-temporal dependence of time series data, we propose a Transformer integrated with GNN model to dynamically capture spatio-temporal 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 a given time series sample.
- In order to extract the linear feature part of the time series data, we integrate the traditional autoregressive linear model AR with the nonlinear neural network in parallel, which further improves the robustness of the model.
- We demonstrate that DGFormer outperforms eight state-of-art baseline methods on three public IoT time series datasets, with a 19.3% improvement in the model's average F1-score.

The related work is introduced in Sect. 2. Section 3 introduces the proposed DGFormer model. Section 4 evaluates the methodology on real time series datasets. Section 5 summarizes the work.

2 Related Work

The study of anomaly detection in time series has been carried out for several decades and is an active research area that is gaining increasing attention in machine learning and data mining. At the same time, many models for time series anomaly detection are proposed. Here we mainly introduce our work from two aspects: the statistics-based method and the deep learning-based method.

2.1 Statistics-Based Method

Traditional statistical methods are mainly used on single-feature time series data, and most of them are linear methods. Kahya et al. used the statistical methods of Cumulative Sum (CUSUM) [11] to build the correlation model of time series data to strengthen the forecasting ability of the US stock exchange and retail industry. Janacek et al. use Autoregressive Integrated Moving Average (ARIMA) [12] model to predict time series data. And Chen et al. propose to use isolated forest and elliptical envelope to detect geochemical anomalies [13]. These methods are good for short-term linear time series data prediction, but not so good for long-term time series data prediction. As time series data become more and more multi-dimensional and complex, these methods can no longer meet the current needs, and deep learning methods have received widespread attention due to the powerful representation ability of deep neural networks.

2.2 Deep Learning-Based Method

Currently, two popular deep learning models, CNN and RNN, are widely used for anomaly detection. These models typically use LSTM layers and stacked CNN layers to extract features from time series, and then apply softmax layers to predict labels. For more accurate prediction, complex structures such as the recursive skip layer (LSTNet-S), the

temporal attention layer (LSTNet-A) [14], and the new temporal pattern attention mechanism have been proposed [15]. Park et al. propose the LSTM-VAE model [16], which adopts the LSTM backbone for time modeling and adopts Variational Auto Encoder (VAE) for reconstruction. MLSTM-FCN [17] uses LSTM layers and stacked CNN layers along extruding and exciting blocks to generate potential features. TapNet [18] also builds the LSTM layer and the stacked CNN layer. Bidirectional Recurrent Neural Network (BiRNN) models [19] improve the prediction accuracy of the model by adding a direction to the general RNN. Compared with LSTM, Gate Recurrent Unit (GRU) [20] has only update gates and reset gates, which greatly simplifies the running time of the model and reduces the complexity of the model, but its prediction accuracy is close to that of LSTM. In addition, some deep learning models, including THOC [21] uses recurrent neural networks (RNN) with jump connections to effectively extract multi-scale time features from time series, integrate multi-scale time features through hierarchical clustering mechanism, and then detect anomalies through multi-layer distance. GANs [22] detects anomalies by modeling nonlinear correlations between multiple time series and performing adversarial regularization. However, the limitations of the above models are obvious: they assume the same effects between time series variables, so they cannot model pairwise dependencies between variables explicitly, and the model accuracy is not high enough [23]. Recent studies show that GNN combined with Transformer can be an effective method for anomaly detection.

In traditional anomaly detection, Transformer can be used to capture global dependencies and context information of data. By introducing self-attention mechanism in Transformer, models can focus more on important nodes and edges to capture unusual patterns and features. While GNN can be used to model relationships and dependencies between data. Both have their advantages. Graph Transformer [24] provides an example of how to generalize the Transformer architecture to graphs by introducing the topological structure properties of graphs in Transformer, so that the model has prior of structural positions in a high-dimensional space. Use Laplacian eigenvectors as absolute encoding and calculate attention on the immediate region of each node, rather than on the entire graph [25]. It combines the core of Transformer (global focus) with the core of GNN (considering the topological properties of graphs). SAN [26] is similar but computes attention on the full picture, distinguishing between real edges and created edges. Mialon et al. [27] propose a way to bias self-attention calculations by relative coding via a kernel on the graph, and then incorporate the location information into Transformer by selecting a kernel function. Other recent work has attempted to incorporate structural information into graph Transformer by using GNN to integrate graph structures [28, 29]. All of them explicitly incorporate graph structures to design graph Transformer architectures that take into account both local and global information.

However, the past methods have always mined the features in time series statically, ignoring the dynamic evolution of time series. Therefore, we first use Transformer with anomaly attention to extract time features [39], and then use graph structure learning to propose a dynamic relationship embedding strategy to dynamically capture spatio-temporal features, adaptively learn the adjacency matrix, and finally improve the timeliness of the entire model.

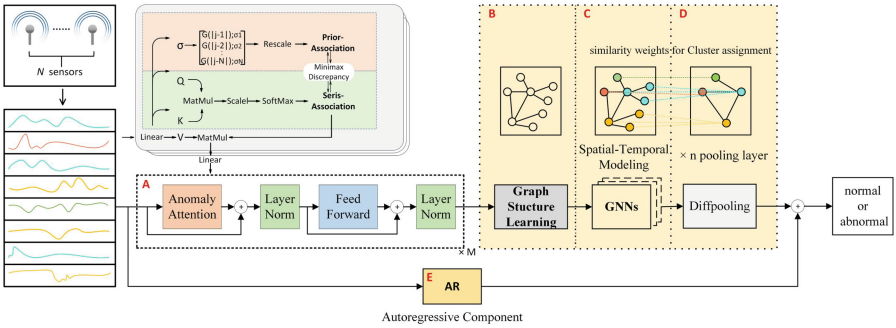

Fig. 2. DGFormer Framework.

3 Method

We'll look at the DGFormer model in detail. A diagram of DGFormer is shown in Fig. 2. The first input is a set of time series $X = \{x_1, x_2, ..., x_n\}$, which is a sequence of measurements in chronological order observed by N sensors working with each other, usually the time interval between two consecutive measurements is constant, where $x_t \in R^N$ represents the observation at time t. The whole model is a parallel structure, that is, the model is processed by nonlinear module and linear module in parallel. In the nonlinear module, we propose to use an effective Anomaly Transformer model with anomaly attention module (see Fig. 2(A)) to embed each univariable sequence in X into the representation vector of time information, and extract the time features. Then the adjacency matrix is generated adaptively by learning the spatio-temporal features of dynamically captured sequences through the graph structure (see Fig. 2(B)). Besides, soft clustering is carried out on each layer of GNN (see Fig. 2(C)) based on the node Embedding vector by Diffpooling module (see Fig. 2(D)), and deep GNN are established by Stacking repeatedly. After that, anomaly detection results of nonlinear modules are output. The linear module consists of an autoregressive model AR (see Fig. 2(E)). Finally, the results of these two parts are weighted and summed, and the result output x_t is normal or abnormal.

3.1 Transformer with Anomaly Attention Mechanism

As shown in Fig. 2(A), considering the limitations of traditional Transformer in anomaly detection, we design an effective Transformer with anomaly attention mechanism [39]. It has the anomaly attention of two branch structures (the upper part of Fig. 2(A)), and for the prior association, a learnable Gaussian kernel is used to calculate the prior with respect to the relative time distance. A learnable scaling parameter σ is also used for Gaussian kernels to adapt prior correlations to various time series patterns. The sequence association branch learns the association relationship from the original sequence, and it can adaptively find the most effective association relationship. (1) The module has shown effective results in practice. The Transformer itself consists of two main modules: anomaly attention block and fully connected layer Feed Forward, which together make

up the Transformer layer. (2) Stack multiple layers to form Transformer model. The purpose of this stage is to extract the time features and construct the feature matrix $X^{(i)}$ as well as to get anomaly attention. The feature matrix for each sequence is as follows:

$$X^{(i)} = Embed_1(x_t) \in R^{n \times N} \tag{1}$$

In the anomaly attention module, a learnable Gaussian kernel is first used to calculate the prior relative to the relative time distance, and then the 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:

$$P^m = \text{Res}cale\left(\left[\frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{|j-i|^2}{2\sigma_i^2}\right)\right]_{i,j\in\{1,\ldots,n\}}\right) \tag{2}$$

$$Z^m = S\text{oftmax}\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V \tag{3}$$

wherel $m \in \{1, \ldots, M\}$ denotes the output of the m^{th} layer with $N_{\text{mod }el}$ channels, $Q, K, V \in R^{n \times N_{\text{mod }el}}$, generates a prior association $P^m \in R^{n \times n}$ based on the learning scale $\sigma \in R^{\text{n}\times 1}$, and the i^{th} element σ_i corresponds to the i^{th} point in time. Its associated weight with the j^{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 $|j-i|$. In addition, Re$scale(\cdot)$ 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 *Soft* max$(\cdot)$ represents normalization of the attention force along the last dimension. In order to better control associative learning, a minimax strategy is proposed. Specifically, the minimization phase is one that drives a prior association to approximate the sequence association learned from the original sequence. This process will adapt the prior associations to various time patterns. In the maximization phase the sequence association is optimized to enlarge the association difference.

The module also uses a multi-head attention mechanism, and for K heads, the learning scale is $\sigma \in R^{n \times \text{K}}$. $\text{Q}_k, K_k, V_k \in R^{n \times \frac{N_{\text{mod }el}}{\text{K}}}$ represents the query, key, and value of the k^{th} head respectively. The outputs $\left\{\hat{Z}_k^m \in R_{1\le k\le \text{K}}^{n \times \frac{N_{\text{mod }el}}{K}}\right\}$ from the multiple heads is then connected and the final result $\hat{\text{Z}}^m \in R^{n \times N_{\text{mod }el}}$ 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 [30]. Its formula is as follows:

$$D\text{is}(P, \text{Z}; X) = \left[\frac{1}{\text{M}}\sum_{\text{m}=1}^{\text{M}}\left(\text{KL}\left(P_i^m, : ||\text{Z}_i^m, :\right) + \text{KL}\left(\text{Z}_i^m, : ||P_i^m, :\right)\right)\right]_{i=1,\ldots N} \tag{4}$$

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

3.2 Dynamic Graph Learning

The main feature of the latter part is to dynamically capture spatio-temporal features of time series to generate the adjacency matrix, and then transfer it to the GNN to extract the attribute information and structure information of the nodes.

In the dynamic graph learning part, a dynamic relationship embedding strategy is proposed, which considers the dynamic modeling of the spatio-temporal features information of datasets. As shown in Fig. 2(B), the time window is mainly used to deal with data that is continuous in time, and GNN model is applied to it for feature learning within each time window. Then the data of the whole time series is processed by sliding the time window to capture its dynamic evolution process. Associations between sensors have been learned through graph structures. Because undirected graphs are symmetric, they cannot represent asymmetric dependencies and causality between sensors. Therefore, this paper will use the directed graph connection feature to show the dependencies between different sensors, use the nodes of the graph to represent the sensors, and use the edges between the nodes to 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:

$$A^{(i)} = Embed_2(x_{\mathrm{t}}) \tag{5}$$

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

$$C_{ij}^{(i)} = \frac{\exp(-\sigma(distance(x_i, x_j)))}{\sum_{p=0}^{n} \exp(-\sigma(distance(x_i, x_p)))} \tag{6}$$

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

$$A^{(i)} = \sigma\left(C^{(i)} W_1\right) \tag{7}$$

where W_1 is the learnable model parameter and σ 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 c_1 to make the adjacency matrix sparse:

$$A^{(\mathrm{i})} = \begin{cases} A_{ij}^{(i)} A_{ij}^{(i)} \geq c_1 \\ 0 A_{ij}^{(i)} \leq c_1 \end{cases} \tag{8}$$

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

3.3 Graph Neural Network

As shown in Fig. 2(C), the module uses 3 GNN layers (G_1, G_2, G_3) 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:

$$X_{encode}^{(i)}, A_{encode}^{(i)} = G_3(G_2(G_1(X^{(i)}, A^{(i)}))) \tag{9}$$

where $i = 1, 2, ..., n$, $X_{encode}^{(i)} \in R^{n \times N_{\text{mod el}}}$, $A_{encode}^{(i)} \in R^{n \times n}$ is composed of graph neural network layer GNN and batch normalization layer. GNN can be such as GCN, GAT and GIN, etc. 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(D), the overall transformation of a pooling layer is shown in Eq. (9) and the following two equations show the process in the Diffpool layer, where $W_2 \in R^{N_{\text{mod el}} \times N_{\text{Diffpool}}}$ 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 $S^{(i)}$.

$$X_{Diffpool}^{(i)} = \sigma\left(S^{(i)} X_{encode}^{(i)} W_2\right) \tag{10}$$

$$A_{Diffpool}^{(i)} = \sigma\left(S^{(i)} A_{encode}^{(i)} \left(S^{(i)}\right)^T\right) \tag{11}$$

We generate centroids $K^{(i)} \in R^{N \times n_{\text{Diffpool}} \times N_{\text{mod el}}}$ based on the input graph and then compute and aggregate the relationship between every batch of centroids and the encoded graph for assignment matrix $S^{(i)}$. We can compute the relationship $S_p^{(i)} \in R^{n_{\text{Diffpool}} \times n} (p = 1, 2, ..., N)$ and $K_p^{(i)} \in R^{n_{\text{Diffpool}} \times N_{\text{model}}} (p = 1, 2, ..., N)$. 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.

$$S_p^{(i)} = \cos ine\left(K_p^{(i)}, X_{encode}^{(i)}\right) \tag{12}$$

$$S_p^{(i)} = normalize\left(S_p^{(i)}\right) \tag{13}$$

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

$$S^{(i)} = \Gamma_\varphi \left(\Big\|_{p=1}^{|N|} S_p^{(i)} \right) \tag{14}$$

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

$$x_{\text{final}}^{(i)} = P_3\left(P_2\left(P_1\left(X_{encode}^{(i)}\right)\right)\right) \tag{15}$$

3.4 Autoregressive Model

As shown in Fig. 2(E), AR model is widely used in time series analysis. As a linear model, it is easy to understand and implement. It describes the relationship between the current value and the historical value, and uses the historical time data of the variable to predict itself. It provides a simple and efficient way to model and predict time series data using only past observations as independent variables, with no other complex factors to consider. The four parts of Fig. 2(A,B,C,D), together form a nonlinear module, which mainly extracts the nonlinear feature part of the data. It makes the output scale of the neural network insensitive to the input scale, because our data set has both linear feature parts and nonlinear feature parts. A mixture of linear and nonlinear modules is used as the final result of DGFormer to enhance the recognition ability of linear features. We first use the output of the nonlinear module to get the result $x_{\text{final}} \in R^{n \times N_{\text{mod }el}}$, while the result obtained by the AR part of the linear module is expressed as $x_{AR} \in R^{n \times N_{\text{mod }el}}$. Finally, the weighted sum of the two is used to get the final result $\hat{X}_t$ of DGFormer.

The final anomaly score is as follows:

$$\mathrm{Score}(X) = soft\max(-D\mathrm{is}(P, \mathrm{Z}; X))\Theta[\|x_t - \hat{x}_t\|_2^2]_{t=1,\ldots,n} \tag{16}$$

where Θ is element-by-element multiplication.

4 Experiment

The three main points we try to verify in our experimental study are as follows:

(1) Does the DGFormer framework allow us to find anomaly more efficientlythan we would otherwise? Yes.
(2) What is the influence of dynamic graph structure learning on anomaly detection? The dynamic adjacency matrix used in our model finally achieves the best result compared to other adjacency matrices.
(3) Does a hybrid anomaly score provide more information than an anomaly score using Transformer or GNN alone? Yes.

Table 1. Details of the experimental baseline datasets. #App represents the application of the data set, #AR represents the proportion of truth value anomalies in the entire data set, and d represents the dimension.

Datasets	#App	#Train	#Test	#AR	d
MSL	Space	58317	73729	0.105	55
SMAP	Space	135183	427617	0.128	25
PSM	Server	132481	87841	0.278	25

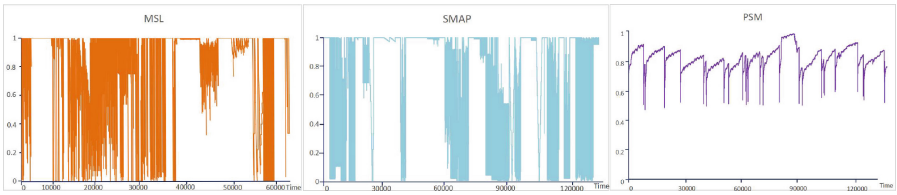

Fig. 3. The feature representation for one dimension of MSL, SMAP and PSM datasets.

4.1 Datasets and Experimental Setup

Datasets. Since this paper is based on time series data in IoT, we mainly choose the practical datasets related to IoT to evaluate our propose DGFormer model: The first is the two public datasets on the IoT, MSL (Mars Science Laboratory Rover) and SMAP (Soil Moisture Active and Passive Satellite) [31], contain remote sensing anomaly data obtained in the Spacecraft Monitoring System Event Surprise Anomaly Emergency Anomaly (ISA) report. And the dataset PSM(Pool Server Metrics) [32] collected within multiple application server nodes is also used as a supplementary dataset. Table 1 reports the statistics for these datasets. Figure 3 describes the one-dimensional feature representation of the three datasets. It can be seen that there are significant differences in feature distribution among them, and also shows that datasets we select have diversity distribution.

Baseline Model. To fully demonstrate the strength of our model, we compare DGFormer to the following eight baselines, these include several classic models such as ALAD, OC-SVM, LSTM, SO_GAAL and USAD, several recent new models such as TRANAD, Anomaly Transformer and MTAD_GAT.

ALAD: Adversarially Learned Anomaly Detection [33], is implemented using the PYOD[1] library, with hyperparameters set to batch_size = 32, dec_layers = 10, and dropout_size = 0.2 as a rule of experience.

OC-SVM: One-Class Support Vector Machines [34], is implemented using PYOD_1 library, and the hyperparameter is set to kernel = 'rbf', degree = 3, coef = 0.0.

SO_GAAL: Single-Objective Generative Adversarial Active Learning [35], uses a mini-max game between a generator and a discriminator that generates adversarial learning to directly generate information-rich potential outliers.

USAD: UnSupervised Anomaly Detection on Multivariate Time Series [36], 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: Long short-term memory [37], 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.

[1] http://github.com/yzhao062/pyod.

MTAD_GAT: MTAD_GAT [38], 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: Anomaly Transformer [39], 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.

TRANAD: TRANAD [40], consist 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. Experimental details follow Shen et al. [21]. All neural network models are optimized by using the Adam optimizer, with the initial learning rate set to 10^{-4}. If the anomaly scores of a point in time (Eq. (13)) are greater than some threshold δ, then we mark the point in time as an anomaly. A threshold of δ is determined so that r proportion of the data in validation datasets are marked as anomaly. Specifically, non-overlapping sliding Windows are mainly used to obtain a set of subsequences, and the size of sliding Windows is fixed at 100. The Transformer with the anomaly attention have 3 layers, 512 channels to set hidden state, and 8 digits of h. GNN have three layers with an output dimension of 128, and the number of nodes in the pooling layer is 1. For the experiment, set r to equal 1%. The hyperparameter λ is set to 3 to weigh the two parts of the loss function, and the training process is stopped early in 10 periods with a batch size of 32. All experiments were implemented in Pytorch3.8 using a single NVIDIA GeForce 930MX GPU.

4.2 Main Result

DGFormer Achieves a Consistent Up-to-Date Level Across All Baseline Model Tests. In order to measure the effectiveness of various anomaly detection methods, we use Precision, Recall and the harmonic average of precision and recall (F1-score) as evaluation indicators. As shown in Table 2, DGFormer achieve 94.92%、98.11% and 96.53% F1-scores on datasets MSL, PSM and SMAP, respectively, which are 20.34%, 18.2% and 19.48% higher on average than other methods. Precision and Recall are consistently up to date across all benchmark models, and we observe that it is compelling to consider the advantages of transformer's integration with GNN in time series anomaly detection. In addition, we plot the F1-score bar chart in Fig. 4 for a complete comparison. DGFormer has the highest F1-score on all three datasets. This means that is important for real-world applications.

4.3 Ablation Study

To demonstrate the efficiency of our architecture design, careful ablation studies are conducted, and the test results measured using F1-score(%) are shown in Table 3.

Table 2. DGFormer's quantitative results across three real world datasets. P, R and F1 indicate Precision, Recall, and F1-score (expressed in %), respectively. The F1-score is a harmonic average of precision and recall. For these three metrics, a higher value indicates better performance, where the highest score is highlighted in bold.

Datasets	MSL			SMAP			PSM		
Metric	P	R	F1	P	R	F1	P	R	F1
ALAD	52.58	95.31	68.06	53.34	59.07	56.17	61.15	93.95	74.08
OCSVM	59.96	90.11	65.41	53.91	59.07	56.37	78.52	90.21	83.96
SO_GAAL	89.94	90.34	61.78	67.28	53.30	59.48	46.25	49.59	47.86
LSTM	85.45	82.50	83.95	89.41	78.13	83.39	76.93	89.64	82.80
USAD	97.95	99.12	88.57	81.39	96.27	89.74	79.62	97.29	76.53
TRANAD	96.15	99.99	94.64	80.43	98.72	89.15	81.50	98.99	95.97
MTAD_GAT	76.23	98.24	86.78	75.16	99.91	85.83	76.28	98.33	81.09
Anomaly Transformer	98.46	98.33	94.19	93.54	98.18	96.27	95.20	96.89	97.01
DGFormer	98.85	97.59	**94.92**	94.32	98.89	**96.53**	97.64	98.58	**98.11**

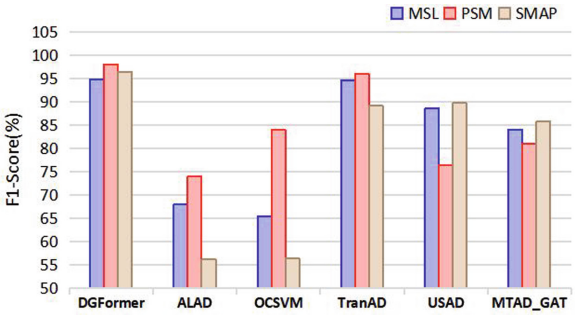

Fig. 4. Comparison of F1-score (%) results between DGFormer and partial baseline models on three datasets using bar charts. The MSL, SMPA and PSM datasets are represented in blue, gray, and pink columns, respectively. (Color figure online)

The Impact of Embedding Strategies on Anomaly Detection Using Dynamic Graph. In Table 3, DGFormer-one is a DGFormer framework with an all-in-one adjacency matrix. DGFormer-corr is a DGFormer framework with adjacency matrix of correlation coefficients. DGFormer framework has the dynamic adjacency matrix proposed by us. You can see that different adjacency matrices can be used in our DGFormer framework. However, the performance of the all-one matrix is slightly worse than that of the correlation coefficient matrix, and our dynamic matrix achieves the best.

Hybrid Transformer and GNN Have a Higher Average F1-Score Than Other Combinations. In Table 3, DGFormer-woAR means that the AR component is removed

Table 3. Ablation results of DGFormer (F1-score (%)). Where, DGFormer-one, DGFormer-corr, DGFormer-woAR and DGFormer-woDG represent neural network modules with full adjacency matrix, adjacency matrix with correlation coefficient, no autoregressive module and no dynamic embedded graph, highest scores are highlighted in bold.

Methods	MSL	SMAP	PSM	Avg F1(as %)
DGFormer-one	92.40	96.19	97.68	95.42
DGFormer-corr	92.69	96.50	97.65	95.61
DGFormer-woAR	92.35	95.48	97.29	95.04
DGFormer-woDG	92.82	96.47	97.33	95.54
DGFormer	**94.92**	**96.53**	**98.11**	**96.52**

from the DGFormer model, and DGFormer-woDG means that the dynamic graph embedding and graph neural network segments are removed from the DGFormer model. The complete DGFormer obtains the best results in different batchsizes. It shows that all components contribute to detection performance of the overall model. The performance of DGFormer-woDG has decreased, which indicates that adding GNN to dynamically capture temporal features can improve the timeliness of the model. DGFormer-woAR's performance degradation is even more pronounced, indicating that AR components play a crucial role. The reason is that AR is generally robust to scale changes in the data [32].

5 Summary

In this paper, we propose a deep learning framework, Dynamic Graph transFormer (DGFormer), An Effective Dynamic Graph Transformer based Anomaly Detection Model for IoT Time Series. It overcomes the defects of traditional Transformer and GNN, and proposes an effective model to obtain GNN parameters by using Transformer with anomaly attention mechanism, and dynamically capture timing features by learning the graph structure. Finally, by parallelizing an autoregressive model AR, a model with strong interpretability was obtained. DGFormer has achieved state-of-art results on a detailed set of empirical studies. For future research, there is hope to explore and design more powerful graph Transformer that can be incorporated into our DGFormer framework to obtain more expressive performance and further improve the usefulness of our method.

References

1. Renjie, W., Eamonn, J.K.: Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. IEEE Trans. Knowl. Data Eng. **35**(3), 2421–2429 (2021)
2. Liang, W., Huang, W., Long, J., et al.: Deep reinforcement learning for resource protection and real-time detection in IoT environment. IEEE Internet Things J. **7**(7), 6392–6401 (2020)
3. Muhammad, S.: Fog computing and its role in the internet of things: concept, security and privacy issues. Int. J. Comput. Appl. **180**(32), 7–9 (2018)

4. Xin, R., Chen, P., Zhao, Z.: CausalRCA: causal inference based precise fine-grained root cause localization for microservice applications. J. Syst. Softw. **203**, 111724 (2023). https://doi.org/10.1016/j.jss.2023.111724
5. Peng, C., et al.: Effectively detecting operational anomalies in large-scale IoT data infrastructures by using a GAN-based predictive model. Comput. J. **65**(11), 2909–2925 (2022)
6. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 1–58 (2009)
7. Zhang, R., Chen, J., Song, Y., Shan, W., Chen, P., Xia, Y.: An effective transformation-encoding-attention framework for multivariate time series anomaly detection in IoT environment. Mob. Netw. Appl. 1–13 (2023). https://doi.org/10.1007/s11036-023-02204-9
8. Tang, M., Fu, X., Wu, H., Huang, Q., Zhao, Q.: Traffic flow anomaly detection based on robust ridge regression with particle swarm optimization algorithm. Math. Prob. Eng. **2020**, 1–10 (2020)
9. Venkatesan, R., et al.: Hyperspectral image features classification using deep learning recurrent neural networks. J. Med. Syst. (2019). https://doi.org/10.1007/s10916-019-1347-9
10. Wu, Y., Dai, H.N., Tang, H.: Graph neural networks for anomaly detection in industrial internet of things. IEEE Internet Things J. **9**(12), 9214–9231 (2021). https://doi.org/10.1109/JIOT.2021.3094295
11. Kahya, E., Theodossiou, P.: Predicting corporate finacial distress: a time-series CUSUM methodology'. Rev. Quant. Finan. Account. **13**(4), 323–345 (1996)
12. Janacek, G.: Time series analysis forecasting and control. J. Time **31**(4), 303 (2010)
13. Chen, Y., Wang, S., Zhao, Q., Sun, G.: Detection of multivariate geochemical anomalies using the bat-optimized isolation forest and bat-optimized elliptic envelope models. J. Earth Sci. **32**(2), 415–426 (2021)
14. Lai, G., Chang, W.C., Yang, Y., Liu, H.: Modeling long-and short-term temporal patterns with deep neural networks. In: International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 95–104. ACM (2018)
15. Song, Y., Xin, R., Chen, P., Zhang, R., Chen, J., Zhao, Z.: Identifying performance anomalies in fluctuating cloud environments: a robust correlative-GNN-based explainable approach. Future Gener. Comput. Syst. **145**, 77–86 (2023)
16. Park, D., Hoshi, Y., Kemp, C.C.: A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. IEEE Rob. Autom. Lett. **3**(3), 1544–1551 (2018)
17. Fazle, K., Somshubra, M., Houshang, D.: Insights into lstm fully convolutional networks for time series classification. IEEE Access **7**, 67718–67725 (2019)
18. Zhang, X., Gao, Y., Lin, J., et al.: TapNet: multivariate time series classification with attentional prototypical network. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 6845–6852 (2020)
19. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
20. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation (2014). https://doi.org/10.3115/v1/D14-1179
21. Shen, L., Li, Z., Kwok, J.: Timeseries anomaly detection using temporal hierarchical one-class network. Adv. Neural Inf. Process. Syst. **33**, 13016–13026 (2020)
22. Mehdi, M., Bing, X., et al.: Generative adversarial networks. Commun. ACM **63**, 139–144 (2020)
23. Qi, S., Chen, J., Chen, P., Wen, P., Niu, X., Xu, L.: An efficient GAN-based predictive framework for multivariate time series anomaly prediction in cloud data centers. J. Supercomput. 1–26 (2023). https://doi.org/10.1007/s11227-023-05534-3
24. Xavier, B., et al.: A generalization of Transformer networks to graphs. DLG-AAAI (2020). https://doi.org/10.48550/arXiv.2012.09699

25. Shao, P., He, J., Li, G., Zhang, D., Tao, J.: Hierarchical graph attention network for temporal knowledge graph reasoning. Neurocomputing **550**, 126390 (2023)
26. Devin, K., et al.: Rethinking graph transformers with spectral attention. In: NeurIPS (2021). https://doi.org/10.48550/arXiv.2106.03893
27. Chen, D., et al.: A trainable optimal transport embedding for feature aggregation and its relationship to attention. In: ICLR (2021). https://doi.org/10.48550/arXiv.2006.12065
28. Pan, Y., et al.: A novel approach to scheduling workflows upon cloud resources with fluctuating performance. MONET **25**(2), 690–700 (2020)
29. Chen, P., Xia, Y., Pang, S., Li, J.: A probabilistic model for performance analysis of cloud infrastructures. Concurr. Comput. Pract. Exp. **27**(17), 4784–4796 (2015)
30. Christopher, M.B., et al.: Pattern Recognition and Machine Learning. Springer, New York (2006)
31. Ahmed, A., Zhuanghua, L., Tomer, L.: Practical approach to asynchronous multivariate time series anomaly detection and localization. In: KDD, pp. 2485–2494 (2021)
32. Ya, S., Wei, S., et al.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: SIGKDD Explorations, pp. 2828–2837 (2019)
33. Houssam, Z., Manon, R., Bruno, L., et al.: Adversarially learned anomaly detection. In: IEEE International Conference on Data Mining (ICDM) (2018). https://doi.org/10.1109/ICDM.2018.00088
34. Bernhard, S., et al.: Support vector method for novelty detection. Adv. Neural Inf. Process. Syst. (1999)
35. Liu, Y., Li, Z., Zhou, C., et al.: Generative adversarial active learning for unsupervised outlier detection. IEEE Trans. Knowl. Data Eng. **32**(8), 1517–1528 (2019). https://doi.org/10.1109/TKDE.2019.2905606
36. Julien, A., Pietro, M., Frédéric, G., Sébastien, M., Maria A.Z.: USAD: unsupervised anomaly detection on multivariate time series. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3395–3404 (2020)
37. Martin, S., Ralf, S., Hermann, N.: LSTM neural networks for language modeling. In: Interspeech (2012). https://doi.org/10.1016/0165-6074(89)90269-X
38. Zhao, H., Wang, Y., Duan, J., et al.: Multivariate time-series anomaly detection via graph attention network. In: ICDM (2020). https://doi.org/10.1109/ICDM50108.2020.00093
39. Xu, J., Wu, H., Wang, J., Long, M.: Anomaly transformer: time series anomaly detection with association discrepancy. In: ICLR (2021). arXiv preprint arXiv:2110.02642
40. Giuliano, C.: TranAD: deep Transformer networks for anomaly detection in multivariate time series data. In: Proceedings of the VLDB Endowment (2022). https://doi.org/10.48550/arXiv.2201.07284