Alexandria Engineering Journal 106 (2024) 560–570

Contents lists available at ScienceDirect

Alexandria Engineering Journal

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

Original article

Utilizing correlation in space and time: Anomaly detection for Industrial Internet of Things (IIoT) via spatiotemporal gated graph attention network

Yuxin Fan [a], Tingting Fu [b], Nikolai Izmailovich Listopad [c], Peng Liu [b], Sahil Garg [d,e,*], Mohammad Mehedi Hassan [f]

[a] HDU-ITMO Joint Institute, Hangzhou Dianzi University, Hangzhou, 310018, China
[b] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, 310018, China
[c] Belarusian State University of Informatics and Radioelectronics, Minsk, 220013, Belarus
[d] Electrical Engineering Department, École de technologie supérieure, Montreal, H3C 1K3, Canada
[e] Centre for Research Impact & Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, 140401, Punjab, India
[f] Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh, 11543, Saudi Arabia

ARTICLE INFO

Keywords:
Industrial Internet of Things
Graph structure learning
Gated graph attention network
Temporal convolutional network
Anomaly detection

ABSTRACT

The Industrial Internet of Things (IIoT) infrastructure is inherently complex, often involving a multitude of sensors and devices. Ensuring the secure operation and maintenance of these systems is increasingly critical, making anomaly detection a vital tool for guaranteeing the success of IIoT deployments. In light of the distinctive features of the IIoT, graph-based anomaly detection emerges as a method with great potential. However, traditional graph neural networks, such as Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs), have certain limitations and significant room for improvement. Moreover, previous anomaly detection methods based on graph neural networks have focused only on capturing dependencies in the spatial dimension, lacking the ability to capture dynamics in the temporal dimension. To address these shortcomings, we propose an anomaly detection method based on Spatio-Temporal Gated Attention Networks (STGaAN). STGaAN learns a graph structure representing the dependencies among sensors and then utilizes gated graph attention networks and temporal convolutional networks to grasp the spatio-temporal connections in time series data of sensors. Furthermore, STGaAN optimizes the results jointly based on both reconstruction and prediction loss functions. Experiments on public datasets indicate that STGaAN performs better than other advanced baselines. We also visualize the learned graph structures to provide insights into the effectiveness of graph-level anomaly detection.

1. Introduction

Many traditional industries, such as energy, healthcare, manufacturing, and water treatment, are undergoing digital transformation, with the Industrial Internet of Things (IIoT) playing a significant role [1]. The IIoT connects sensors, switches, and other IoT devices through networks, enabling data acquisition, processing, and automatic control, thereby enhancing production efficiency and economic benefits in industrial environments [2].

However, with the advancement of digital transformation, the number of devices and sensors in IIoT systems has rapidly increased, leading to a rise in overall system complexity. Traditional industrial systems, which had lower complexity and lacked security considerations, now face significant challenges in terms of security and stability in the Industry 4.0 era. For instance, water companies experience nearly 3 billion liters of water loss daily due to leaks, resulting in significant economic losses [3].

In IIoT, anomaly detection refers to the process of using IoT devices and sensors to collect data in industrial environments, then analyzing and monitoring this data to identify abnormal situations. On one hand, with the advent of IIoT, sensors and IoT devices are exposed to public networks without adequate security protection, making them vulnerable targets for malicious attacks [4]. On the other hand, anomaly detection also aids in the real-time or early detection of potential issues, preventing situations that could lead to significant economic losses [5]. Therefore, efficient anomaly detection is crucial for ensuring the security of IIoT.

* Corresponding author at: Electrical Engineering Department, École de technologie supérieure, Montreal, H3C 1K3, Canada.
E-mail address: sahil.garg@ieee.org (S. Garg).

https://doi.org/10.1016/j.aej.2024.08.048
Received 21 March 2024; Received in revised form 26 July 2024; Accepted 14 August 2024
Available online 26 August 2024

In real-world large-scale IIoT systems, numerous interconnected devices continuously collect large amount of time-series data via sensors [6]. These sensor data often exhibit complex and nonlinear interdependencies. Anomaly detection is commonly regarded as an unsupervised learning task due to the scarcity of labeled anomaly data in historical records, as well as the unpredictable and highly diverse characteristics exhibited by anomalies. In recent years, deep learning techniques, such as Autoencoders (AE) [7], have made significant advancements in detecting anomalies in high-dimensional data by utilizing reconstruction errors for anomaly scoring. Additionally, methods based on Recurrent Neural Networks (RNNs) [8] and Generative Adversarial Networks (GANs) [9] have demonstrated promising results in anomaly detection. Sometimes, deep learning can be combined with edge computing to provide more reliable performance [10]. However, most of these methods often lack explicit consideration of the structural connections among sensors, especially in cases with complex and highly nonlinear relationships, limiting their ability to detect and interpret anomalous events effectively.

Usually, interconnected devices influence each other. Introducing graph neural networks into anomaly detection in IIoT is a very promising method for detecting anomalies in various domains such as transportation, energy, factories, and dynamic networks. Graph Neural Networks (GNNs) [11] are rapidly evolving models for handling complex patterns in graph-structured data and hold great potential for anomaly detection in IIoT. If we regard a device as a node, the state of it is influenced by the states of its neighboring nodes. In GNNs, each node aggregates the features of its neighboring nodes to form its own feature representation. Different GNN variants consider different numbers of neighboring nodes in the feature aggregation process. For instance, Graph Convolutional Networks (GCNs) [12] consider the one-step adjacent neighbors of the node, while Graph Attention Networks (GATs) [13] use an attention mechanism to assign different weights to adjacent nodes during aggregation. However, these traditional GNNs have certain limitations and still have significant room for improvement. Moreover, previous anomaly detection methods based on graph neural networks have focused only on capturing dependencies in the spatial dimension, lacking the ability to capture dynamics in the temporal dimension.

In this work, we propose a new method called STGaAN, which captures the spatial and temporal dependencies of sensor time series to detect anomalies. The key achievements of our research are outlined as follows:

- We introduce STGaAN, a Spatio-Temporal Graph Attention Network that incorporates a multi-head attention mechanism along with a compact convolutional sub-network. This sub-network calculates a soft gate for each attention head, controlling its importance.
- We jointly optimize the reconstruction module based on the variational autoencoder and the prediction module based on Multilayer Perceptron (MLP) to leverage both strengths.

2. Related work

2.1. Anomaly detection in IIoT

Anomalies in IIoT can encompass various problems related to equipment, sensors, production processes, or entire systems. The applications of anomaly detection in the industrial sector span across manufacturing, energy, water treatment, transportation, and more, contributing to improved production efficiency, cost reduction, equipment safety assurance, and the prevention of potential downtime or losses. Existing anomaly detection models can be divided into two paradigms: prediction-based and reconstruction-based models [14].

Prediction-based models leverage prediction errors as indicators for anomaly detection. Predictions generated by an LSTM network can be interpreted using an unsupervised and non-parametric thresholding approach called LSTMNDT [15], thereby establishing an automated anomaly detection system to monitor the telemetry data transmitted from spacecraft. The Deep Autoencoding Gaussian Mixture Model (DAGMM) [16] is a deep learning approach for anomaly detection. It can jointly analyze the data distribution and identify anomalous patterns within the data by integrating an autoencoder with a Gaussian mixture model. DAGMM exhibits excellent anomaly detection performance, particularly for high-dimensional and complex datasets. In [17], the authors investigate how data augmentation can be applied to address these challenges and improve detection performance in an anomaly detection task using IoT datasets. However, prediction-based models attempt to deterministically predict the actual value of the next timestamp, which makes them highly sensitive to the randomness of time series data.

The reconstruction-based model acquires the representation of the entire time series by reconstructing the initial input using latent variables. The Mixed Anomaly Detection Generative Adversarial Network (MAD-GAN) [9] utilizes the capabilities of Generative Adversarial Networks (GANs) to model normal data and detect anomalies by generating synthetic data. By training GANs to capture the underlying data distribution, MAD-GAN can identify deviations from expected patterns, demonstrating high performance in anomaly detection tasks. The Long Short-Term Memory Variational Autoencoder (LSTM-VAE) [18] combines the Long Short-Term Memory (LSTM) neural network with the Variational Autoencoder (VAE). It can capture the latent distribution in data, identify abnormal patterns, and is particularly suitable for anomaly detection tasks in sequential data. Federated learning could be utilized for cooperative learning [19]. An intelligent unsupervised learning approach is used to identify anomalous data by MEUs and help to identify anomalous nodes in [20]. However, reconstruction-based models may perform inadequately when dealing with sudden perturbations in time series, especially when these perturbations still conform to a normal distribution.

2.2. Graph neural networks

As GNNs have been used to improve resource usage prediction in IIoT, such as exploring spatial and temporal information within the cellular traffic data [21], GCNs is a significant milestone in the development of GNNs. Graph and node classification tasks can benefit from GCN's efficient parameter sharing and representation learning capabilities, which are achieved by layer-by-layer aggregation of information from neighboring nodes. GATs employ self-attention mechanisms to weight the contributions of neighboring nodes, allowing different nodes to have varying influences on the target node, thereby enhancing representation learning performance.

Furthermore, researchers have considered features in the temporal dimension. Spatio-Temporal Graph Convolution (ST-GCN) is a GNN method designed specifically for handling spatio-temporal graph data, primarily applied in tasks such as video action recognition and traffic flow prediction. ST-GCNs (Spatio-Temporal Graph Convolutional Networks) [22] extend traditional GCNs to model the spatio-temporal relationships between nodes. They consider historical data across time steps and utilize convolutional operations to capture dynamic spatio-temporal patterns, which leads to improved accuracy in traffic flow prediction. Diffusion Convolutional Recurrent Neural Networks (DCRNNs) [23] combine diffusion convolution with recurrent neural networks. They employ diffusion convolutional layers to capture spatial structures in traffic networks and use recurrent neural networks to model temporal information, resulting in more accurate traffic flow predictions.

The training of GNN is inseparable from the graph structure. However, predefined graph structures often either do not exist or are not optimal, necessitating the use of graph structure learning to identify the best graph structure. In the field of traffic flow prediction, Graph WaveNets [24] first proposes using two learnable embedding matrices

to automatically construct an adaptive graph based on the input traffic data. In the domain of anomaly detection, GTA [25] employs a connection learning strategy to learn the graph structure, directly learning bidirectional links between sensors using the Gumbel-softmax sampling method.

Besides the aforementioned works, not only correlations between data streams, but also the enormous real-time data stream and variable data distribution are considered in a novel method (referred to as DLSHiForest) [26], which is based on Locality-Sensitive Hashing and the time window technique. This method is proposed to solve these problems while achieving accurate and efficient detection. In previous studies, anomaly detection using GAT only employed a single-head attention mechanism, resulting in limited model expressiveness and generalization ability, and lacking resistance to noise and outliers in the input. Additionally, previous methods only model the spatial dimension of sensors and neglect the time dimension, potentially resulting in the loss of important features. Drawing inspiration from the field of traffic flow prediction, we introduce modeling of the time dimension for sensors in our model to prevent the loss of crucial features.

3. The proposed solution: Spatiotemporal gated graph attention network

3.1. Sensor network and data collection

In a large IIoT system, data required for anomaly detection is collected from a sensor network composed of various types of sensors. It is crucial to consider the requirements for data security levels, network energy consumption, and delays occurring during data collection and transmission. To address these concerns, we adopt an RPL protocol with appropriate clustering to construct the sensor network [27].

In the RPL protocol, the DIO message contains the distance, version number, rank, and an objective function that is used to calculate the rank. In the RPL protocol, in upstream routing, the root node initiates the construction of DODAG by sending the first DIO message. All nodes, upon receiving a DIO message, select the root as their preferred parent and then calculate their own rank according to the chosen objective function. Each node then transmits its DIO message, which includes the updated rank, to all neighboring nodes. Upon receiving a DIO message, each node compares its own rank with the received one. If the received rank is higher, the message is disregarded. Conversely, if the received rank is lower, the node increments its distance by one and updates its routing table with the identifier of the sending node. In our study, we enhance the ranking calculation formula by incorporating residual energy considerations for selecting the primary cluster node as depicted in Eq. (1).

$$newRank = ETX / ResidualEnergy, \tag{1}$$

the node whose rank is lower is selected as the master node in the cluster.

In industrial networks, the number of connected devices or scalability increases rapidly. This forces IoT protocols to face routing and security issues. Classic routing protocols are poorly adapted to the IoT environment due to resource limitations and heterogeneity. In addition to the RPL protocol, we propose to use a hybrid objective function that combines more than one metric in an additive manner to select the optimal route. The metric to determine the rank of a node uses parameters such as latency D_e, and the probability of packet loss Z_e. For the convenience of further analysis, the probability of losing packets Z_e will consider such a parameter as the logarithm of the probability of passing packets $X_e = ln(1 - Z_e)$.

To find the rank of a node, we propose using the following convolution:

$$r = \frac{D_{s,j}}{D^{max}} + \frac{X_{s,j}}{X^{min}}, \tag{2}$$

where $D_{s,j}$ means the delay for each sensor node, $X_{s,j}$ stands for the probability of packet loss for each sensor node, D^{max} represents the maximum permissible delay, and X^{min} is the minimum acceptable losses.

After receiving a DIO (RPL protocol) message, each node generates a list of possible parent nodes. During the selection procedure, the parent node with the minimum rank value r (Eq. (2)) is selected. In RPL, a node's rank is based on its parent rank and base rank.

3.2. Problem statement and STGaAN architecture

We mathematically define the target problem as follows. The training data is composed of sensor values from N sensors over T_{train} time ticks and the sensor data is denoted $\mathbf{s}_{\text{train}} = \left[\mathbf{s}_{\text{train}}^{(1)}, \ldots, \mathbf{s}_{\text{train}}^{(T_{\text{train}})}\right]$. The sensor values $\mathbf{s}_{\text{train}}^{(t)} \in \mathbb{R}^N$ at each time point t constitute an N dimensional vector.

The test data, denoted as $\mathbf{s}_{\text{test}} = \left[\mathbf{s}_{\text{test}}^{(1)}, \ldots, \mathbf{s}_{\text{test}}^{(T_{\text{test}})}\right]$, consists of sensor data values from N sensors at T_{test} time points. The model outputs a collection of T_{test} binary labels representing whether each of the time point is anomalous during the inference stage, i.e. $a(t) \in \{0, 1\}$, where $a(t) = 1$ show that time t is anomalous.

STGaAN is designed to represent sensors in an Internet of things system as a graph, where sensors are treated as nodes and the learned relationships between them are represented as edges. By incorporating spatiotemporal relationship modeling into anomaly detection, STGaAN enhances the effectiveness of anomaly detection. The framework of STGaAN, illustrated in Fig. 1, consists of six components:

- **Node Embedding:** Utilize node embeddings to acquire unique features for each sensor.
- **Graph Structure Learning:** Learn a graph structure that represents the interconnections among sensors.
- **Gated Graph Attention Aggregator:** Use gated graph attention network to extract spatial dimension features [28].
- **Temporal Convolution Layer:** Use time convolutional network to extract time dimension features [29].
- **Joint Optimization:** Combined optimization with prediction-based model and reconstruction-based model.
- **Anomaly Score:** Verify if the anomaly score surpasses the predefined threshold to determine if the system is abnormal.

3.3. Node embedding

In a large IIoT system, the data gathered from various types of sensors form a multivariate time series. Different time series (sensors) may manifest highly diverse properties and exhibit complex interrelations among each other. For instance, in a water tank IoT system, decisions to activate valves or water pumps may be based on the readings of liquid level sensors observing the tank's water level. Consequently, sensors within the same water tank often exhibit high correlation.

To capture these correlations, we model the sensors and their relationships as a graph, where each sensor represents a node. We then randomly initialize node embeddings for each sensor to capture its inherent properties:

$$\mathbf{v_i} \in \mathbb{R}^{d_1}, \text{ for } i \in \{1, 2, \ldots, N\},$$

where $\mathbf{v_i}$ denotes the embedding for node i, d_1 denotes the dimensionality of the embedding $\mathbf{v_i}$, and N stands for the amount of sensor nodes. The resemblance between node embedding vectors $\mathbf{v_i}$ represents the resemblance of behaviors, hence sensors with similar node embedding vectors exhibit high correlations. The node embeddings will be used for graph structure learning of sensors in IoT systems and gated attention networks.

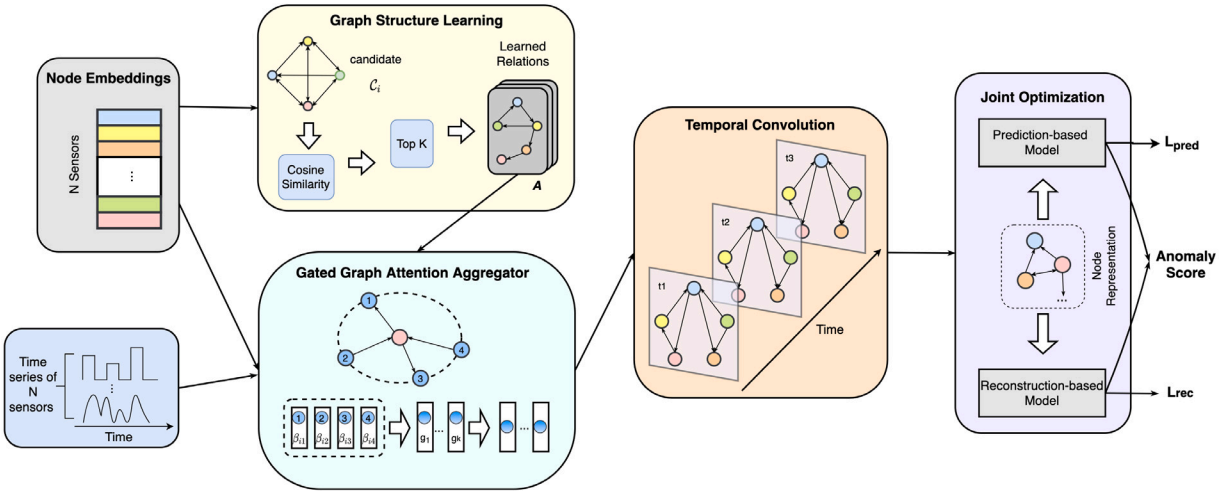


Fig. 1. The framework of STGaAN. STGaAN is composed of Node embeddings, Graph structure learning, Gated graph attention aggregator, Temporal convolution layer, Joint optimization, and anomaly score.

3.4. Graph structure learning

In this section, we will introduce how to define the edges of the sensor relationship graph.

First and foremost, it is essential to clarify that the sensors relationship graph we are modeling is a directed graph. The choice of using a directed graph stems from the fact that the dependency relationships between sensors do not need to be symmetric. The nodes in the graph represent sensors, therefore, we define edges as dependencies between sensors. If there is a directed edge from one sensor to another, it means that the state of this sensor is influenced by the state of another sensor. The adjacency matrix A is used to represent this oriented graph, where A_{ij} indicates that there is an oriented edge from node i to node j.

In an IIoT system with numerous sensors, there might be some prior information that can be used to determine node dependencies. For example, the IIoT system may be divided into multiple subsystems with minimal interaction between sensors in different parts. When such prior information exists, it can be flexibly expressed as a collection of potential relationship candidate C_i for each sensor i, representing the potential neighboring sensor set that may exhibit dependencies with sensor i as shown in Eq. (3):

$$C_i \subseteq \{1, 2, \ldots, N\} \backslash \{i\}, \tag{3}$$

when this prior information does not exist, the candidate relation of sensor i is the entire set of sensors.

To choose the dependency of node i from the set of candidate nodes, cosine similarity is used to calculate the similarity between the embedding vector of sensor i and the embedding vector of its candidate sensor $j \in C_i$ as shown in Eq. (4):

$$e_{ji} = \frac{\mathbf{v_i}^{\top} \mathbf{v_j}}{\|\mathbf{v_i}\| \cdot \|\mathbf{v_j}\|} \text{ for } j \in C_i, \tag{4}$$

where, e_{ji} is the cosine similarity of node j and node i. Then the calculated similarity is sorted in descending order and the first k nodes are selected, i.e., the first k nodes with the greatest similarity to sensor node i are connected to node i with a directed edge. The sparsity of the graph can be controlled by specifying the size of k as shown in Eq. (5):

$$A_{ji} = \mathbb{1}\left\{j \in \text{TopK}\left(\left\{e_{ki} : k \in C_i\right\}\right)\right\}, \tag{5}$$

where, $A_{ji} = 1$. It signifies that there is a directed edge between node j and node i; otherwise, $A_{ji} = 0$. TopK denotes the index set of nodes with the greatest e selected from the candidate set C_i.

3.5. Gated graph attention aggregator

3.5.1. Notion

We represent the single fully connected layer of a nonlinear activation function $\alpha(\cdot)$ as $\text{FC}_\theta^\alpha = \alpha(\text{Wx+b})$, where $\theta = \{\text{W}, \text{b}\}$ is the parameter. Different subscripts of θ indicate different transformation parameters. For the activation function, we represent the sigmoid activation function as $\sigma(\cdot)$ and the LeakyReLU activation function with a negative slope equal to 0.1 as $h(\cdot)$. If the nonlinear transformation is performed without the activation function after the linear transformation, it is expressed as $\text{FC}_\theta(\mathbf{x})$. At time tick t, we define the input of the Gated Graph Attention Aggregator on the historical time series data of dataset using a sliding window of size w as shown in Eq. (6):

$$\mathbf{x}^{(t)} := \left[\mathbf{s}^{(\mathbf{t-w})}, \mathbf{s}^{(\mathbf{t-w+1})}, \ldots, \mathbf{s}^{(\mathbf{t-1})}\right], \tag{6}$$

where, $\mathbf{s}^{(\mathbf{t-w})}$ represents all sensor data at time $t-w$. The present sensor readings, indicated as $\mathbf{s}^{(\mathbf{t})}$, is the expected output that our model needs to forecast.

3.5.2. Feature aggregator

In some of the latest studies, researchers have used multi-head GAT to explore features in different representation subspaces to provide more modeling capabilities. However, although multi-head GAT can explore multiple representation subspaces between a central node and its neighborhood, the importance of all subspaces is not the same; some subspaces may even be irrelevant for specific nodes. The model's final predictive performance may decrease because it incorporates outputs from attention heads that capture irrelevant representations.

Therefore, in this paper, we use Gated Attention Networks (GaAN) to capture the relationships between sensors. Unlike existing graph attention mechanisms, GaAN not only uses multiple attention heads to explore features in different representation subspaces to provide more modeling capabilities but also calculates a gate value between 0 and 1 through a convolutional network to control the importance of each attention head. Since only a simple and effective lightweight subnetwork is introduced when constructing the gates, the computational overhead is negligible, and the model is easy to train.

First, we concatenate the node embedding v_i and the corresponding linear transformed $x_i^{(t)}$ as shown in Eq. (7):

$$u_i^{(t)} = v_i \oplus FC_{\theta_{d_1}^{(b)}}\left(x_i^{(t)}\right), \tag{7}$$

where $\oplus$ denotes concatenation operation; $\theta_{d_1}^{(b)}$ is the parameter of the bth head, and the dimension of the linear transformation of $x_i^{(t)}$ is d_1.

In the second step, by applying softmax function to the inner product value, we calculate the bth attention coefficient β between node i and node j as shown in Eqs. (8) and (9) :

$$\varphi_{ij}^{(b)} = \left\langle u_i^{(t)}, u_j^{(t)} \right\rangle, \tag{8}$$

$$\beta_{ij}^{(b)} = \frac{\exp\left(\varphi_{ij}^{(b)}\right)}{\sum_{l\in\mathcal{N}(i)\cup\{i\}} \exp\left(\varphi_{il}^{(b)}\right)}, \tag{9}$$

where $\langle\cdot,\cdot\rangle$ is a vector inner product, $N(i)$ is the collection of adjacent nodes of node i.

In the third step, after obtaining the normalized attention coefficient between nodes, we can calculate the linear combination of their corresponding features as the aggregate representation y_i of each node as shown in Eqs. (10) and (11):

$$y_i = \mathrm{FC}_{\theta_o}\left(u_i^{(t)} \oplus \|_{b=1}^{B}\left(g_i^{(b)} \sum_{j\in\mathcal{N}(i)} \beta_{i,j}^{(b)} \mathrm{FC}_{\theta_{d_1}^{(b)}}^{h}\left(u_j^{(t)}\right)\right)\right), \tag{10}$$

$$g_i = \left[g_i^{(1)}, \ldots, g_i^{(b)}\right] = \psi_g\left(u_i^{(t)}, u_j^{(t)}\right), \tag{11}$$

where $\oplus$ denotes concatenation operation. We denote $\|_{b=1}^{B}$ as sequentially concatenation. B is the amount of attention heads and $\beta_{ij}^{(b)}$ is the bth attention coefficient between nodes i and j. $\theta_{d_1}^{(b)}$ is the parameter of the bth head, and the dimension is d_1. Concatenate these B outputs with vector u_i, and pass them through a fully connected layer parameterized by θ_o to obtain the final output y_i, which has a dimension of d_o.

$g_i^{(b)}$ is a scalar that is the gate value of the bth head of node i. To prevent the added gate from introducing an excessive number of additional parameters, a convolutional network ψ_g is employed, which utilizes the sensor node i and its neighboring sensor nodes to generate the gate value. There are many possible designs for network ψ_g. In our work, we combine maximum and average pooling to build the network as shown in Eq. (12).

$$g_i = \mathrm{FC}_{\theta_g}^{\sigma}\left(u_i^{(t)} \oplus \max_{j\in\mathcal{N}(i)}\left(\left\{\mathrm{FC}_{\theta_m}\left(u_j^{(t)}\right)\right\}\right) \oplus \frac{\sum_{j\in\mathcal{N}(i)} u_j^{(t)}}{|\mathcal{N}(i)|}\right), \tag{12}$$

where θ_m is responsible for projecting the neighboring features into a d_m dimensional vector prior to selecting the maximum element, while θ_g is tasked with mapping the concatenated features to the ultimate B gate. By setting a smaller d_m, for the subnet computing the gate, the computational overhead will be negligible. $\mathcal{N}$(i) is the set of neighboring nodes of sensor node i, $|\mathcal{N}(i)|$ represents the number of neighboring nodes of sensor node i.

3.6. Temporal convolution layer

The gated attention aggregator aggregates information from neighbor nodes in the spatial dimension for each node, and next, the time convolutional network is leveraged to aggregate the information of the time dimension. The input of the time convolutional network consists of the graph representation $\mathbf{y}^{L_{\mathrm{STGaAN}}}$ derived from the preceding aggregation step, where L_{STGaAN} is the number of layers of STGaAN. Then we make $T^0 = \mathbf{y}^{L_{\mathrm{STGaAN}}}$. The time-level representation is calculated as follows (Eq. (13)):

$$\mathbf{T}^{l+1} = \mathrm{ReLU}\left(\boldsymbol{\Phi} * \left(\mathrm{ReLU}\left(\mathbf{T}^{l}\right)\right)\right), \tag{13}$$

where T^{l+1} represents the time dimension representation of layer $l+1$, ReLU represents the activation function, $\boldsymbol{\Phi}$ represents the convolution kernel, $*$ represents the standard convolution operation, and T^l represents the time dimension representation of layer l.

Time convolutional networks update node features by merging information from neighboring time steps, thus capturing time dynamics effectively. In numerous applications, convolutional networks demonstrate superior performance compared to RNNs, circumventing common drawbacks of recursive models, such as gradient explosion/vanishing and memory retention deficiencies. Moreover, utilizing convolutional networks instead of recursive models can enhance performance by enabling parallel computation of outputs.

3.7. Joint optimization

Both prediction-based models and reconstruction-based models show their advantages in some particular cases. Our model comprises a predictive model for forecasting the next time step value and a reconstructive model for capturing the data distribution of the entire time series. During the training phase, the weights of both models are updated concurrently. The loss function is formulated as the sum of the loss functions of two optimization modules as shown in Eq. (14):

$$\mathcal{L} = \gamma_1 \mathcal{L}_{\mathrm{pred}} + \left(1-\gamma_1\right) \times \mathcal{L}_{\mathrm{rec}}, \tag{14}$$

where $\mathcal{L}_{\mathrm{pred}}$ signifies the loss function of the model based on prediction, $\mathcal{L}_{\mathrm{rec}}$ signifies the loss function of the model based on reconstruction, and γ_1 is the hyperparameter that balances the prediction and reconstruction modules.

3.7.1. Prediction-based model

Prediction-based models predict the value of the next timestamp. We pass the result of the temporal convolutional network into a multilayer perceptron (MLP) and use the root mean square error (RMSE) as the loss function as shown in Eq. (15):

$$\mathcal{L}_{\mathrm{pred}} = \sqrt{\frac{1}{M_{\mathrm{train}} - w} \sum_{t=w+1}^{M_{\mathrm{train}}} \left(\hat{s}^{(t)} - s^{(t)}\right)^2}, \tag{15}$$

where, M_{train} is the amount of samples in the training set processed by the sliding window, w represents the size of sliding Windows, $\hat{s}^{(t)}$ denotes the model's prediction at time tick t, and $s^{(t)}$ denotes the real value at time tick t.

3.7.2. Reconstruction-based model

By reconstructing the initial input s using a few latent variables z, the reconstruction-based model acquires the representation of the whole time series. We adopt variational autoencoders (VAE), which encode inputs not as single points in hidden space, but as probability distributions in hidden space. VAE can catch the data distribution of the whole time series by processing the values of time series as variate. Given an input s, it should be made a conditional distribution $p(s \mid z)$ reconstruction, $z \in R^d$ represents a potential space of vector representation, and d is latent space dimension of the VAE model. The optimization objective is to identify the optimal model parameters that can most effectively reconstruct the data distribution to approximate s. The following Eq. (16) is the real posterior density:

$$p(z \mid s) = p(s \mid z)p(z)/p(s), \tag{16}$$

where the marginal density is expressed as Eq. (17)

$$p(s) = \int p_{(z)} p_{(s|z)} dz. \tag{17}$$

And actually, due to the computational intractability of the posterior distribution $p(z \mid s)$, we approximate this posterior distribution with $q(z \mid s)$. Given the encoder $q(z \mid s)$ and decoder $p(s \mid z)$, the loss function can be calculated as Eq. (18):

$$\mathcal{L}_{\mathrm{rec}} = -\mathrm{E}_{q(z|s)}\left[\log p(s \mid z)\right] + D_{KL}\left(q(z \mid s) \,\|\, p(z)\right), \tag{18}$$

where the first term is the expected negative log-likelihood of the given input s. The second term is the Kullback–Leibler divergence between the encoders distribution $q(z \mid s)$ and $p(z)$, serves as a regularization term.

Algorithm 1 Training stage

Require: training epochs I, sensor training data $\mathbf{s}_{\text{train}} = \left[\mathbf{s}_{\text{train}}^{(1)}, \cdots, \mathbf{s}_{\text{train}}^{(T_{\text{train}})}\right]$, hyperparameters γ_1, γ_2 and batch size M.

1: Initiate the parameter W with random values, which encompasses all learnable parameters in STGaAN;
2: **for** $i = 1 \rightarrow I$ **do**
3: Compute $\mathbf{y}^{L_{\text{GaAN}}}$ in spatial dimension by GaAN;
4: Compute $\mathbf{T}^{L_{tem}}$ in temporal dimension by TCN;
5: Compute the prediction value via prediction module;
6: Compute the reconstruction probability via reconstruction module;
7: Minimize the joint loss function for optimal tuning of parameter W;
8: **end for**
9: **return** W.

3.8. Anomaly score

At each timestamp t, there are two inferred results corresponding to the joint optimization goal. The prediction-based model and the reconstruction-based model generate predicted values $s^{(t)}$ and p_i, respectively, where $s^{(t)}$ denotes the prediction of the time series, and p_i corresponds to the reconstruction probability derived from a model based on reconstruction. The final anomaly score harmonizes their respective strengths, greatly improving the effectiveness of detection effect. The anomaly score at each time tick t is the accumulation of the anomaly scores of N time series. Precisely, the formula for the anomaly score is shown in Eq. (19):

$$score = \sum_{i=1}^{N} \frac{\left(\hat{s}_i^{(t)} - s_i^{(t)}\right)^2 + \gamma_2 \left(1 - p_i\right)}{1 + \gamma_2}, \tag{19}$$

where $(\hat{s}_i^{(t)} - s_i^{(t)})^2$ is the square error between the forecast $\hat{s}_i^{(t)}$ and the observed value $s_i^{(t)}$, signifying the extent of departure of feature i from the predicted value. γ_2 is a hyperparameter used on the test set to balance the deviation from prediction and the likelihood of reconstruction. $1 - p_i$ denotes the likelihood of encountering anomalies for feature i based on the reconstruction model, and N is the overall amount of sensors.

We employ the Peak Over Threshold (POT) algorithm [30] for determining the anomaly threshold on the test set. If the anomaly score for a timestamp exceeds a pre-defined threshold, the timestamp is labeled as anomalous. Ultimately, Alg. 1 encapsulates the comprehensive training phase of STGaAN, while Alg. 2 delineates the Inference stage.

Algorithm 2 Inference stage

Require: Sensor testing data $\mathbf{s}_{\text{test}} = \left[\mathbf{s}_{\text{test}}^{(1)}, \cdots, \mathbf{s}_{\text{test}}^{(T_{\text{test}})}\right]$, model parameter W and hyperparameters γ_1, γ_2.

1: **for** each $\mathbf{s}_{\text{test}}^{(i)}$ **do**
2: Compute the anomaly score of $\mathbf{s}_{\text{test}}^{(i)}$
3: **if** anomaly score < threshold **then**
4: $\mathbf{s}_{\text{test}}^{(i)}$ = "a normal point"
5: **else**
6: $\mathbf{s}_{\text{test}}^{(i)}$ = "an anomaly"
7: **end if**
8: **end for**
9: **return** result binary label vector.

4. Experiments and analysis

To illustrate the efficiency of STGaAN, we conducted thorough experiments. We first introduced four commonly used public datasets alongside eight widely used anomaly detection methods. Subsequently, we described the evaluation metrics and experimental settings. Finally, we assessed the effectiveness of our proposed STGaAN on these datasets. The experimental results indicate that STGaAN performs comparably to or better than a range of baseline methods. We also performed ablation studies on each component of our model. Additionally, we provided insights into the interpretability of STGaAN by analyzing a case study of an IIoT system.

Table 1
The four datasets' statistics.

Datasets	Features	Train	Test	Anomalies (%)
MSL	27	58 317	73 729	10.27
SMAP	55	135 183	427 617	13.13
SWaT	51	47 520	44 991	11.96
WADI	127	102 697	17 280	6.00

4.1. Datasets

Secure Water Treatment (SWaT) [31] is a water treatment experimental platform used for research in the field of cybersecurity. The 11 days of continuous operational data that make up the SWaT dataset are divided into 4 days of data collected during attacks and 7 days under normal operational conditions were included in the SWaT dataset. The data collection process included monitoring all network traffic, sensors, and actuators. The Water Distribution Experimentation (WADI) [32] serves as a natural extension of the SWaT platform, The combination of these two testing platforms enables researchers to observe the cascading effects of network attacks across multiple testing platforms. The WADI dataset consists of 14 days of continuous normal operational data and 2 days of data collected during attack scenarios.

The Mars Science Laboratory Rover (MSL) [33] and Soil Moisture Active Passive Satellite (SMAP) [33] are real datasets collected by NASA's spacecraft. NASA experts utilize these datasets to mitigate unexpected events that may pose risks to the spacecraft during post-launch operations. The training set comprises normal data, while the testing set contains annotated anomalies.

In Table 1, we show the statistics for the four public datasets utilized.

4.2. Baselines

4.2.1. PCA

Principal Component Analysis [34] is utilized in anomaly detection by reducing the dimensionality of the data while preserving its key features. Data points that show significant departures from the principal components in this lower-dimensional space are classified as anomalies.

4.2.2. AE

Autoencoders [35] are employed in anomaly detection by training a neural network to reconstruct input data. The reconstruction error must surpass a predetermined threshold in order for anomalies to be identified, which makes AE a valuable method for finding odd patterns in a variety of data kinds.

4.2.3. DAGMM

Deep Autoencoding Gaussian Mixture Model [16] learns a probabilistic representation of the data to detect anomalies. It models normal data as a Gaussian mixture and identifies anomalies based on their deviation from this learned distribution, making it effective in detecting subtle anomalies in complex datasets.

4.2.4. LSTM-VAE

By employing LSTM to encode sequential data and learning a probabilistic latent space representation, Long Short-Term Memory Variational Autoencoder [18] is used in anomaly detection. It detects anomalies by measuring the reconstruction error or the deviation of data points from the learned probabilistic model, making it suitable for time-series and sequential data anomaly detection tasks.

4.2.5. MAD-GAN

The Multiple Anomaly Detection Generative Adversarial Network [9] is a deep learning framework employed in anomaly detection. It operates by training a GAN-like architecture to produce data that closely mimics the distribution of normal data. Anomalies are detected when the generated data differs significantly from the actual data distribution, enabling effective detection of rare and unusual instances in various datasets.

4.2.6. OmniAnomaly

OmniAnomaly [36] is an anomaly detection framework designed for multi-modal data, which leverages both temporal and non-temporal information to detect anomalies in complex and diverse datasets. It combines techniques like convolutional autoencoders, LSTM networks, and probabilistic modeling to provide robust anomaly detection capabilities across different data modalities.

4.2.7. GDN

Graph Deviation Network [6] utilizes graph attention mechanisms for structural learning in multivariate time series data as an unsupervised anomaly detection approach. It explains detected anomalies based on attention scores. The goal of the GDN approach is to learn the connections between sensors as a graph. Deviations from the learned patterns are then recognized and explained.

4.3. Evaluation metrics

The evaluation metrics that we employ are precision (Prec), recall (Rec), F1-score (F1), and the area under the ROC curve (AUC). The percentage of samples that are truly positive out of all the samples the model predicts is referred to as precision. It gauges the proportion of the sample that is truly positive among those predicted to be positive by the model. The recall rate, also called sensitivity, is the percentage of samples that are truly positive among all samples that the model correctly predicted to be positive. This metric measures how well the model can identify positive samples. An increased recall rate suggests that the model is more effective at identifying genuine positive samples. The F1-score is the weighted harmonic mean of precision and recall, providing a combined metric for precision and recall. It is appropriate for assessing the model's balance on both positive and negative samples, taking accuracy and recall rate into account in a comprehensive manner. The following are the formulas of precision, recall rate and F1-score:

$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{20}$$

$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{21}$$

$$\text{F1} = \frac{2 \times \text{Prec} \times \text{Rec}}{\text{Prec} + \text{Rec}}, \tag{22}$$

where, TP, FP, and FN represent true positives, false positives, and false negatives, respectively.

4.4. Experimental setup

The approach of STGaAN was developed in PyTorch [37] version 1.5.1 with NVIDIA Corporation GP102 [GeForce GTX 1080 Ti] graphics cards for server training, Intel(R) Xeon(R) W-2133 CPU @ 3.60 GHz x 12 and PyTorch Library version 1.5.0. The training set-up involved the Adam optimizer with a 1×10^{-3} learning rate. A total of 50 epochs and 32 batch sizes are used to train the entire network. The embedding dimension d_1 is set to 128 for all data sets. We empirically set the number of attention heads per dataset to 4, the sliding window size to 32, and the time convolution kernel size to 16. For MSL, SMAP, SWaT, and WADI, we have K set to 15, 30, 15, and 30 correspondingly. Grid search yielded the values of γ_1 and γ_2 as 0.4 and 0.8 for the model hyperparameters. On the validation data set, we established an exception threshold using the POT algorithm [30]. Any timestamp where the outlier score surpasses the threshold will be deemed "abnormal" during the inference stage.

4.5. Results and analysis

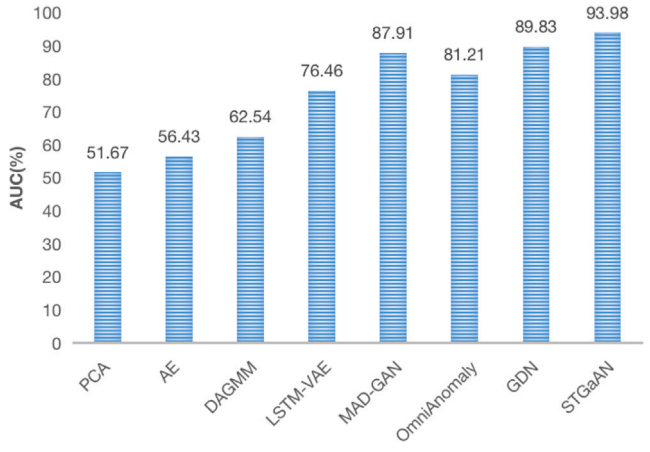
Table 2 presents the comparative results of accuracy, recall, and F1 score between STGaAN and baseline methods on four datasets. The results indicate that STGaAN demonstrates outstanding generalization ability, achieving the best performance across the datasets used in the experiments. It can be seen in Table 2 that F1-score (%) of STGaAN on the MSL and SMAP datasets is 1.36 and 1.77 higher than the best baseline, respectively. The majority of baseline methods demonstrate better performance on the MSL and SMAP datasets, since they involve relatively simple anomaly patterns and spatiotemporal dynamics. In contrast, the SWaT and WADI datasets encompass more intricate anomaly scenarios, resulting in diminished performance of most baseline methods on these two datasets. However, it is observed in Table 2, STGaAN achieves F1 scores (%) on the SWaT and WADI datasets that are 3.24 and 3.02 higher, respectively, compared to the best-performing baselines.

The limitation of traditional methods (such as PCA) and the majority of deep learning approaches lies in their failure to simultaneously consider the temporal and spatial associations of time series data. OmniAnomaly, for example, does not explicitly address spatial correlation in the model, which is a crucial factor for successful anomaly detection in multivariate time series. In this paper, the proposed GaAN layer is dedicated to solving this problem. The experiment results validate the superiority of this layer, since it significantly and consistently outperforms OmniAnomaly across all four datasets. Meanwhile, the time information is also very important for the anomaly detection of multivariate time series. For example, the DAGMM algorithm performs inadequately because it does not take time information into account. In STGaAN, TCN is utilized to grasp long-term temporal interdependencies. This design helps achieve better performance than DAGMM. Recently, GDN has achieved superior efficiency in comparison to other benchmarks. However, GDN lacks effectiveness in capturing temporal features from time series data, and GAT has limited ability to generalize attention mechanisms. Building upon the foundation of multi-head attention in our STGaAN, we allocate significance to attention heads and leverage TCN to capture temporal dependencies.

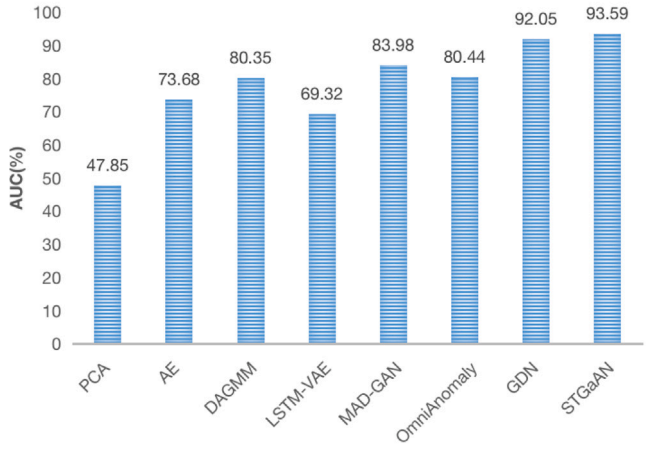
We further computed the AUC on the MSL and SMAP datasets as a performance metric, as shown in Fig. 2. Through the use of GaAN, TCN, and joint optimization techniques within STGaAN, our proposed model consistently outperforms other baseline methods. By incorporating GaAN and time convolutional networks, we consider both spatial and temporal dependencies and capture the spatiotemporal relationships between time series. Additionally, the combination of prediction-based and reconstruction-based models improves the accuracy of model parameters in STGaAN.

Table 2
The evaluation metrics of accuracy (%), recall (%), and F1-score are contrasted with those of existing methods across four datasets. The superior performance is emphasized by highlighting the finest results, while the second-finest results are underscored.

Method	MSL			SMAP			SWaT			WADI		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
PCA	28.69	24.53	26.45	27.93	20.44	23.61	25.13	22.18	23.56	38.72	5.41	9.49
AE	72.47	51.38	60.13	73.12	78.94	75.92	71.95	53.04	61.06	35.21	34.18	34.69
DAGMM	50.53	92.37	65.32	59.64	89.43	71.56	27.46	69.52	39.37	52.84	27.31	36.01
LSTM-VAE	51.48	93.72	66.46	85.51	63.66	72.98	96.24	59.91	73.85	86.34	15.58	26.40
MAD-GAN	85.17	89.91	87.48	80.49	82.14	81.31	98.97	63.74	77.54	42.19	33.54	37.37
OmniAnomaly	88.67	91.17	89.89	74.16	97.76	84.34	98.36	65.12	78.36	98.15	13.22	23.31
GDN	90.21	85.25	87.66	90.12	88.03	89.06	99.35	68.12	80.82	94.57	41.61	57.79
STGaAN	94.83	87.93	**91.25**	92.15	89.54	**90.83**	98.82	73.13	**84.06**	98.13	44.05	**60.81**



(a) MSL dataset



(b) SMAP dataset

Fig. 2. AUC (%) performance on MSL and SMAP datasets.

4.6. Ablation experiment

To investigate the necessity of the three components—GaAN, TCN, and joint optimization—we investigated the impact of these components through ablation experiments. First, we assessed the effect of replacing GaAN with a standard multi-head GAT by removing the soft gates that control the importance of different heads. Next, we evaluated the impact of removing the time convolution module in STGaAN. Lastly, we analyzed the effect of removing the reconstruction module from the joint optimization module and only using the prediction module. Table 3 summarizes the experimental findings:

Table 3
The ablation experiment on MSL.

Method	Precision	Recall	F1
STGaAN	94.83	87.93	91.25
- Soft gate	93.25	87.43	90.25
- TCN	92.61	86.15	89.26
- JO	90.18	85.52	87.79

- Substituting GaAN with a conventional multi-head GAT resulted in a decline in the model's effectiveness. This indicates that utilizing a small convolutional subnet to compute a soft gate for each attention head to regulate its significance is advantageous for enhancing the model's efficacy. Employing multiple attention heads enables exploration of features across different representational subspaces, thereby inherently providing increased modeling capacity. Yet, treating each attention head equally results in missing out on the opportunity to benefit from those attention heads that are inherently more important than others. The final predictive performance of the model might degrade due to the input containing outputs from attention heads that capture irrelevant representations.
- STGaAN with TCN outperforms the model without TCN, indicating that the time convolutional network effectively captures temporal dependencies.
- Removing the reconstruction module from the joint optimization module and using only the prediction module also results in a decline in model performance. While the prediction-based model is susceptible to the randomness inherent in time series data, making deterministic predictions about the subsequent timestamp's actual value, the reconstruction model learns the distribution of random variables, thereby mitigating this issue and improving robustness against noise and perturbations. Therefore, employing joint optimization methods is beneficial for enhancing the effectiveness of anomaly detection.

4.7. Hyperparameter γ_1 and γ_2 analysis

We conducted additional experiments on the MSL dataset to analyze the impact of balancing the values of γ_1 and γ_2, based on prediction error and reconstruction probability, respectively, which are used in the proposed loss function and anomaly score. Table 4 shows the F1-scores for all combinations of γ_1 values from 0.2 to 0.8 in increments of 0.2 and γ_2, values from 0.2 to 1 in increments of 0.2. We observed that different settings of γ_1 and γ_2 achieved similar performance across all F1-score evaluation metrics. Specifically, we achieved the best performance when $\gamma_1 = 0.4$ and $\gamma_2 = 0.8$. The results indicate that the proposed STGaAN is not sensitive to the values of γ_1 and γ_2, demonstrating robustness to different hyperparameter settings. By setting γ_1 and γ_2 values between 0.4 and 0.8, we consistently achieved

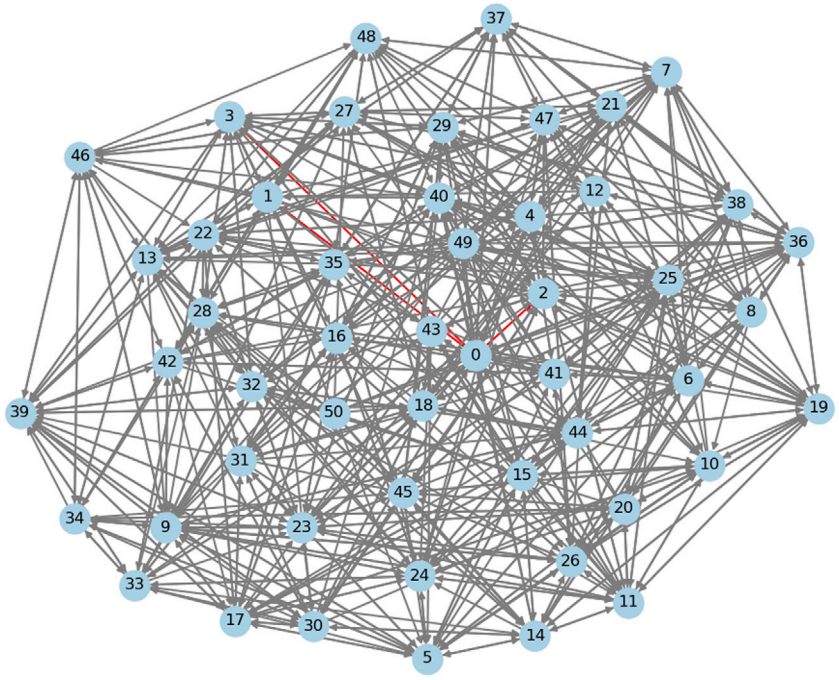
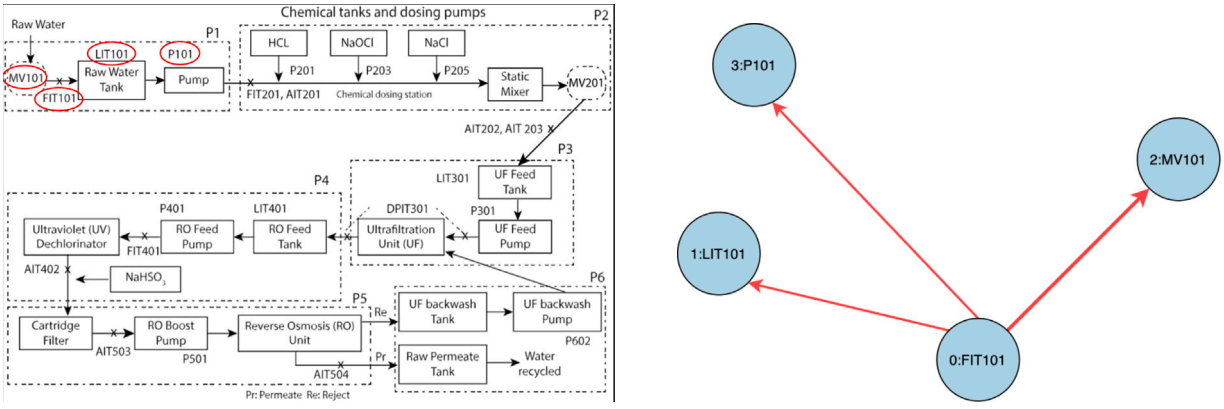


Fig. 3. Graph structure visualization of SWaT data sets. This is a directed graph with 51 nodes, each having 15 directed edges. Nodes represent sensors, and edges indicate the correlations between sensors by specifying which sensors are related to each other within the graph.



(a) SWaT's six-stage processes[38] (b) the graph structure of stage P1

Fig. 4. The left figure depicts the process diagram of the water treatment system that generated the SWaT dataset, illustrating six phases of the water treatment process and the associated sensors. The right figure represents a subgraph of the SWaT graph structure that we learned, focusing on the P1 processing phase. The thickness of the lines in the graph indicates the magnitude of the graph attention coefficients, with thicker lines representing higher attention coefficients and stronger correlations among sensors.

better results compared to other state-of-the-art anomaly detection methods listed in Table 2.

4.8. Graph structure visualization

We visualized the graph structure learnt by STGaAN on the SWaT dataset in Fig. 3. This is a directed graph containing 51 nodes, each connected by 15 directed edges (with an out-degree of 15). The nodes in the graph denote sensors, while the edges denote the correlations between them. For instance, Node 0 corresponds to the FIT101 sensor, installed on the flow indicator transmitter. Node 1 is the LIT101 sensor, installed on the liquid level indicator transmitter. Node 2 represents the MV101 sensor, attached to an electric valve. And Node 3 corresponds to the P101 sensor, which is installed on a water pump. We observed that in the learnt graph structure, these sensors are interconnected, as indicated by the red lines. Referring to the water treatment system's process diagram [38] shown in Fig. 4(a), the first four sensor nodes are all part of the first process stage, P1, which explains their strong correlations. We isolated the graph structure of stage P1 in Fig. 4(b) and further examined the attention scores calculated by GaAN for Node 0, revealing significant attention scores for Nodes 1, 2, and 3, indicating a strong correlation among these nodes. In reality, when the liquid level sensor LIT101 experiences an anomaly, it leads to anomalies in the flow sensor FIT101, valve sensor MV101, and pump sensor P101, as changes in liquid level affect flow, which in turn impacts valve and pump operations. Therefore, when a sensor malfunctions but is difficult to detect, we can utilize the sensor network obtained from the graph structure learning method. By observing the status of the sensors most relevant to the faulty one, we can infer the fault in that sensor, thereby enhancing the effectiveness of fault detection.

5. Conclusion

In this paper, we present STGaAN, a spatiotemporal graph neural network with gated attention mechanisms, designed for anomaly and

Table 4
Different values of γ_1 and γ_2 on the MSL dataset's F1-score. The superior performance is emphasized by highlighting the finest results.

γ_1	γ_2				
	0.2	0.4	0.6	0.8	1
0.2	88.87	89.42	90.69	91.08	89.42
0.4	89.52	90.75	91.04	**91.25**	90.97
0.6	89.65	90.27	90.98	91.01	90.02
0.8	88.92	90.18	90.62	90.87	89.95

fault detection. STGaAN learns the correlation between sensors and node embeddings to obtain the graph structure. It utilizes the Gated Attention Network (GaAN) to capture spatial dependencies and Temporal Convolutional Networks (TCN) to capture temporal dependencies. Finally, model parameters are jointly optimized using both prediction-based and reconstruction-based approaches. We perform comparative experiments on four public datasets. The results indicate that STGaAN outperforms other advanced baseline methods and provides good interpretability for the detection outcomes. Furthermore, we conduct ablation experiments to confirm the necessity of each module in STGaAN, demonstrating that each module is crucial for the final prediction performance. We also visualize the learned graph structure to illustrate the interpretability of STGaAN. Future work could include addressing the imbalance in heterogeneous data and anomalous data from devices. Additionally, investigating the interpretability of graph neural networks in anomaly detection for industrial IoT systems is also worthwhile.

CRediT authorship contribution statement

Yuxin Fan: Writing – original draft, Methodology. **Tingting Fu:** Writing – original draft, Validation, Methodology. **Nikolai Izmailovich Listopad:** Investigation, Formal analysis, Conceptualization. **Peng Liu:** Supervision, Resources, Formal analysis. **Sahil Garg:** Writing – review & editing, Project administration. **Mohammad Mehedi Hassan:** Writing – review & editing, Project administration.

Declaration of competing interest

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

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 62172134 and 62211530448). The authors are grateful to King Saud University, Riyadh, Saudi Arabia for funding this work through Researchers Supporting Project Number (RSP2024R18).

References

[1] J. Jiang, C. Lin, G. Han, A.M. Abu-Mahfouz, S.B.H. Shah, M. Martínez-García, How AI-enabled SDN technologies improve the security and functionality of industrial IoT network: Architectures, enabling technologies, and opportunities, Digit. Commun. Netw. 9 (6) (2023) 1351–1362.
[2] Y. Wu, Z. Wang, Y. Ma, V.C. Leung, Deep reinforcement learning for blockchain in industrial IoT: A survey, Comput. Netw. 191 (2021) 108004.
[3] N. Zhang, Y. Li, Y. Wu, Q. Zhang, Guest editorial: AI empowered communication and computing systems for industrial internet of things, IEEE Trans. Ind. Inform. (2021) 4914–4916.
[4] A. Yazdinejad, M. Kazemi, R.M. Parizi, A. Dehghantanha, H. Karimipour, An ensemble deep learning model for cyber threat hunting in industrial internet of things, Digit. Commun. Netw. 9 (1) (2023) 101–110.
[5] Y. Zuo, Y. Wu, G. Min, C. Huang, K. Pei, An intelligent anomaly detection scheme for micro-services architectures with temporal and spatial data analysis, IEEE Trans. Cogn. Commun. Netw. 6 (2) (2020) 548–561.
[6] A. Deng, B. Hooi, Graph neural network-based anomaly detection in multivariate time series, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 4027–4035.
[7] P. Shixin, C. Kai, T. Tian, C. Jingying, An autoencoder-based feature level fusion for speech emotion recognition, Digit. Commun. Netw. (2022).
[8] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, G.W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017.
[9] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, S.-K. Ng, MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks, in: Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series, Springer International Publishing, Cham, 2019, pp. 703–716.
[10] J. Mills, J. Hu, G. Min, Multi-task federated learning for personalised deep neural networks in edge computing, IEEE Trans. Parallel Distrib. Syst. 33 (3) (2022) 630–641.
[11] P. Li, L. Wang, W. Wu, F. Zhou, B. Wang, Q. Wu, Graph neural network-based scheduling for multi-UAV-enabled communications in D2D networks, Digit. Commun. Netw. (2022).
[12] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the International Conference on Learning Representations, 2017.
[13] Y. Liu, S. Yang, Y. Xu, C. Miao, M. Wu, J. Zhang, Contextualized graph attention network for recommendation with item knowledge graph, IEEE Trans. Knowl. Data Eng. 35 (1) (2023) 181–195.
[14] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, Q. Zhang, Multivariate time-series anomaly detection via graph attention network, in: 2020 IEEE International Conference on Data Mining, ICDM, IEEE, 2020, pp. 841–850.
[15] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, 2018, pp. 387–395.
[16] B. Zong, Q. Song, M. Min, W. Cheng, C. Lumezanu, D.-K. Cho, H.-F. Chen, Deep autoencoding Gaussian mixture model for unsupervised anomaly detection, in: International Conference on Learning Representations, 2018.
[17] B. Weinger, J. Kim, A. Sim, M. Nakashima, N. Moustafa, K.J. Wu, Enhancing IoT anomaly detection performance for federated learning, Digit. Commun. Netw. 8 (3) (2022) 314–323.
[18] D. Park, Y. Hoshi, C. Kemp, A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder, Cornell University - arXiv, 2017.
[19] J. Mills, J. Hu, G. Min, Communication-efficient federated learning for wireless edge intelligence in IoT, IEEE Internet Things J. 7 (7) (2020) 5986–5994.
[20] L. Wang, L. Chen, N.N. Xiong, A. Liu, T. Wang, M. Dong, An intelligent active probing and trace-back scheme for IoT anomaly detection, Digit. Commun. Netw. (2023).
[21] Z. Wang, J. Hu, G. Min, Z. Zhao, Z. Chang, Z. Wang, Spatial-temporal cellular traffic prediction for 5G and beyond: A graph neural networks-based approach, IEEE Trans. Ind. Inform. 19 (4) (2023) 5722–5731.
[22] S. Yan, Y. Xiong, D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
[23] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: Proceedings of the International Conference on Learning Representations, 2017.
[24] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph WaveNet for deep spatial-temporal graph modeling, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, 2019, pp. 1907–1913.
[25] Z. Chen, D. Chen, X. Zhang, Z. Yuan, X. Cheng, Learning graph structures with transformer for multivariate time series anomaly detection in IoT, IEEE Internet Things J. (2022) 9179–9189.
[26] Y. Yang, S. Ding, Y. Liu, S. Meng, X. Chi, R. Ma, C. Yan, Fast wireless sensor for anomaly detection based on data stream in an edge-computing-enabled smart greenhouse, Digit. Commun. Netw. 8 (4) (2022) 498–507.
[27] O.A. Lavshuk, N.I. Listopad, Routing method in iIoT networks using clustering for the RPL protocol, Probl. Fiz. Mat. Tekh. (Probl. Phys. Math. Tech.) 4 (57) (2023) 74–80.
[28] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, D.Y. Yeung, GaAN: Gated attention networks for learning on large and spatiotemporal graphs, in: Conference on Uncertainty in Artificial Intelligence, 2018.
[29] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018, ArXiv abs/1803.01271.
[30] A. Siffer, P.-A. Fouque, A. Termier, C. Largouet, Anomaly detection in streams with extreme value theory, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.
[31] J. Goh, S. Adepu, K.N. Junejo, A. Mathur, A dataset to support research in the design of secure water treatment systems, 2017, pp. 88–99.
[32] C.M. Ahmed, V.R. Palleti, A.P. Mathur, WADI, in: Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks, 2017.

[33] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 387–395.

[34] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang, A novel anomaly detection scheme based on principal component classifier, in: International Conference on Data Mining, 2003.

[35] O.I. Provotar, Y.M. Linder, M.M. Veres, Unsupervised anomaly detection in time series using LSTM-based autoencoders, in: 2019 IEEE International Conference on Advanced Trends in Information Theory, ATIT, 2020.

[36] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.

[38] A.P. Mathur, N.O. Tippenhauer, SWaT: A water treatment testbed for research and training on ICS security, in: 2016 International Workshop on Cyber-Physical Systems for Smart Water Networks, CySWater, IEEE, 2016, pp. 31–36.