International Journal of Information Security and Privacy
Volume 18 • Issue 1

An IIoT Temporal Data Anomaly Detection Method Combining Transformer and Adversarial Training

Yuan Tian, Yan'an University, China
Wendong Wang, Yan'an University, China*
Jingyuan He, Yan'an University, China

ABSTRACT

The existing Industrial Internet of Things (IIoT) temporal data analysis methods often suffer from issues such as information loss, difficulty balancing spatial and temporal features, and being affected by training data noise, which can lead to varying degrees of reduced model accuracy. Therefore, a new anomaly detection method was proposed, which integrated Transformer and adversarial training. Firstly, a bidirectional spatiotemporal feature extraction module was constructed by combining Graph Attention Networks (GAT) and Bidirectional Gated Recurrent Unit (BiGRU), which can simultaneously extract spatial and temporal features. Then, by combining multi-scale convolution with Long Short-Term Memory (LSTM), multi-scale contextual information was captured. Finally, an improved Transformer was used to fuse multi-dimensional features, combined with an adversarial-trained variational autoencoder to calculate the anomalies of the input data. This method outperforms other comparison models by conducting experiments on four publicly available datasets.

KEYWORDS

BiGRU, Graph Attention Networks, IIoT, Residual Network, Temporal Data Anomaly Detection, Transformer

Time series data refers to a sequence of data points that are indexed or graphed in chronological order. This type of data typically reflects the developmental patterns and changing characteristics of things over time. Temporal data is recorded in human life everywhere, like stock prices in the financial sector, temperature in specific regions during a certain period, electrocardiogram trends, real-time monitoring data collected by sensors in the industrial sector, etc., (Abdelrahman & Keikhosrokiani, 2020; Chatterjee & Ahmed, 2022). The core of temporal data analysis focuses on discovering the patterns from data and predicting future value with historical observations, providing the reference and basis for decision-making. Therefore, more and more researchers are starting to study how to design a model to analyze the temporal data (Chen et al., 2021a).

The temporal data in industrial production is becoming increasingly widespread and IIoT can be seen as a collaborative work that provides a collection of technologies for businesses and

DOI: 10.4018/IJISP.343306 *Corresponding Author

applications using the Internet as a carrier (Ennaji et al., 2023; Fang et al., 2021; Garg et al., 2022). It can utilize electronic devices connected to the physical object, and heterogeneous sensors can collect process control data. These devices include industrial automation systems, medical instruments, and personal computers (Lai et al., 2021; Li & Jung, 2022; Lu et al., 2021). The data between these sensors is highly correlated, and this correlation has complex topological structures and nonlinear characteristics. However, there may be anomalies in the data in IIoT, which could have adverse effects. Thus, it is necessary to design a model to effectively detect anomalies in IIoT temporal data (Song et al., 2023). As deep learning is increasingly used in various fields, Wu et al. (2024) introduced a website link security detection algorithm that leverages multi-modal fusion to enhance prediction accuracy. Meanwhile, Guendouz et al. (2023) devised a novel feature selection approach based on the Dragonfly algorithm, aiming to enhance Android malware detection performance. The method combines different characteristics of the data to build a classification model of the results with machine learning algorithms.

Transformer is a powerful model structure that effectively captures the features of input data through self-attention and multi-head attention mechanisms (Balaji & Sankaranarayanan, 2022). At the same time, Transformer also has parallel computing capabilities and can handle large-scale datasets. Therefore, applying Transformer to anomaly detection in IIoT temporal data can improve detection efficiency and accuracy (Su et al., 2019).

However, simple Transformer models are often susceptible to overfitting and adversarial attacks. To address these issues, consider incorporating adversarial training into Deep Learning (DL) (Wang et al., 2019). Adversarial training adds some noise or interference during the training process, making it more suitable for noise and anomalies in input data. Meanwhile, adversarial training can also improve the model's generalization ability, making it perform better on unprecedented data (Xia et al., 2022; Zhong et al., 2022).

An IIoT temporal data anomaly detection method was proposed, which integrated Transformer and adversarial training to solve the problem of traditional anomaly detection methods unable to handle complex industrial temporal data with high data dimensions, high noise interference, and fast pattern changes. The contribution of the proposed method is as follows:

Combining GAT and BiGRU to construct a bidirectional spatiotemporal feature extraction module that balances spatial and temporal features.

Combining multi-scale convolution and LSTM networks to capture multi-scale contextual information, implementing deep feature extraction based on residual networks, while preventing phenomena such as vanishing gradients, exploding gradients, overfitting, and network degradation.

Adopting an improved Transformer to achieve multi-dimensional feature fusion, combining pooling to balance global and local features to avoid issues such as information loss.

Combining adversarial training with a variational autoencoder to amplify abnormal reconstruction errors effectively solves the problem of low model performance caused by training data noise in traditional autoencoder models.

RELATED WORK

Abnormal data in IIoT may lead to damage or malfunction of industrial equipment, affecting the quality and efficiency of industrial production, and reducing the safety and reliability of IIoT systems. Therefore, the detection and management of abnormal data is a very important aspect of IIoT. By using effective anomaly detection methods, abnormal data can be detected and processed in a timely manner; thereby, ensuring the normal operation of the IIoT system.

Prediction Based Methods

When detecting temporal anomaly data, using the Recurrent Neural Network (RNN) and the variants network was a common approach. For the simplification and lightweight consideration of the model

structure, Tran et al. (2022) introduced an anomaly detection model based on self-supervised learning (SSL) to analyze the temporal data. This model has a certain effect on detecting the degree of data anomalies. In addition, a convolutional encoder was applied to capture the correlation of input data. Shen et al. (2020) implemented the detection of abnormal data based on LSTM and GRU. By introducing multiple gating units, long-term dependency relationships have been learned to better achieve long-sequence data processing and anomaly detection. Zhang et al. (2022) proposed an architecture with time-frequency analysis for anomaly detection (TFAD). This model contains modules for data decomposition and expansion, which can improve accuracy to a certain extent. Meng et al. (2020) applied Transformer to data streams by masking temporal data modeling and proposed a masking strategy for anomaly detection using an attention mechanism that includes parallel update time steps. Deng & Hooi (2021) proposed a model based on graph neural networks, which can represent temporal data as graph data and then perform anomaly detection by learning feature vectors of nodes and edges. Zhou et al. (2021) introduced a model based on Transformer: Informer. The computational complexity has been improved because of the attention network, and the prediction progress and long sequence prediction efficiency have been improved.

The above prediction-based data anomaly detection methods can predict future data points by establishing mathematical models or algorithms and comparing actual data with predicted data to detect outliers. These methods have a certain degree of predictability, adaptability, and interpretability. But at the same time, there are also obvious drawbacks, such as high requirements for data volume and high computational costs. In addition, if the data distribution changes, the model may need to be adjusted or retrained. Therefore, it is necessary to consider issues such as data quality, computational cost, and model drift during use. When selecting and using this anomaly detection method, it is necessary to weigh the specific application scenarios and requirements.

Refactoring Based Methods

The reconstruction-based method can define clear topological structures and learn causal relationships between variables. Li et al. (2022) proposed a model called Anomaly PTG, which can achieve good detection performance in different scenarios. This model can detect relatively rare abnormal data. Zhao et al. (2020) used attention to graph neural networks (GNNs) to improve the performance of identifying root causes. Chen et al. (2021b) used automatic learning of graph structures, graph convolution, and the Transformer to capture temporal correlations. Audibert et al. (2020) introduced an unsupervised anomaly detection model with a reverse training autoencoder (USAD). By conducting adversarial training on the encoder-decoder architecture, it was possible to learn how to adapt to reconstruction errors containing abnormal inputs. Wu et al. (2020) proposed an improved Transformer consisting of an encoder and a decoder, which obtained global information from the input through the self-attention mechanism. However, due to its quadratic complexity with the input data, the efficiency was low. Zangrando et al. (2023) designed an out-of-distribution neural networks model for anomaly detection (ODIN AD), which can detect temporal data anomalies at multiple stages, such as data preparation and evaluation. Li et al. (2019) proposed an improved GAN-based anomaly detection model (MAD-GAN). Under the GAN framework, LSTM was used as the basic model, and then a new discriminant and reconstruction anomaly score was introduced to detect anomalies using discriminators and generators.

The current temporal data anomaly detection methods in IIoT have the characteristic of high automation, which can automatically learn data features and perform anomaly detection, reducing the need for manual intervention. However, there are also problems such as high false alarm rates, difficulty in real-time detection, lack of interpretability, and inability to process large-scale and low-quality data. It is necessary to study anomaly detection methods with high accuracy, real-time performance, interpretability, and scalability to adapt to the challenges of the IIoT environment.

PROBLEM DEFINITION

The IIoT data is generally collected and generated by multiple IIoT sensors. In this paper, the multi-sensor data inputs sampled at equal time intervals are represented as follows:

$$X = \left\{x_1, x_2, ..., x_T\right\} \tag{1}$$

In eq (1), $X \in R^{T \times m}$ is the input of the MSTSAD model, T is the maximum length of timestamp, m is the dimension of features of the data, and $x_t \in R^m$ is the value at the time of t. For the input data X, the anomaly label sequence $Y = \left\{y_1, y_2, ..., y_T\right\}$ needs to be provided. If $y_t \in \left\{0,1\right\}$ is 0, it demonstrates that the situation at time t is normal, and if $y_t \in \left\{0,1\right\}$ is 1, it demonstrates that the situation at time t is abnormal.

To eliminate the adverse effects caused by significant numerical differences between feature dimensions in temporal data, the normalization of the minimum maximum method is adopted to normalize each dimension of input data. The calculation method is as follows:

$$x_i = \frac{x_i - \min\left(x_{tr,i}\right)}{\max\left(x_{tr,i}\right) - \min\left(x_{tr,i}\right)} \tag{2}$$

In eq (2), $\max\left(x_{tr,i}\right)$ and $\min\left(x_{tr,i}\right)$ are the max and min values on the i-th dimension of the input data, respectively. Due to the time correlation of temporal data, which means there is a time dependence between different time points, considering the dependency relationship between observation points and historical points. This article generates the input of the MSTSAD model through the sliding window, and it can be represented as:

$$W_t = \left\{x_{t-n+1}, x_{t-n}, ..., x_{t-1}, x_t\right\} \tag{3}$$

X is not directly used as the input result, but rather the multi-dimensional temporal data X is divided into a sliding window W as the input of the MSTSAD model. This makes the model's anomaly detection of time observation points not only focus on itself but also combine historical time dependence information to score the anomalies at time points.

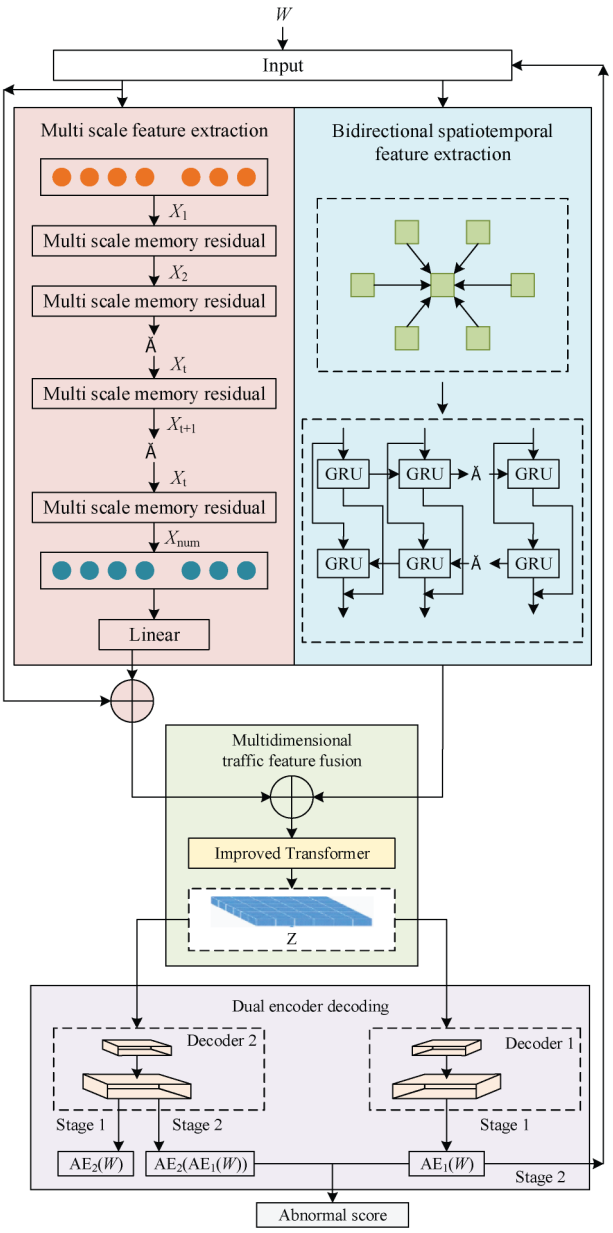
PROPOSED ANOMALY DETECTION METHOD

Fig.1 shows the framework of the proposed MSTSAD model (Multiscale time series prediction for anomaly detection). In Fig. 1, is the input of the MSTSAD model. First, each feature of the original data is normalized, and then all the features of the normalized data are concatenated into a tensor. The tensor is fed into the multi-scale feature extraction and bidirectional spatiotemporal feature extraction module. In the multi-scale feature extraction module, a multi-scale memory residual network is adopted to extract the temporal features of the tensor. In the bidirectional spatiotemporal feature extraction module, the GAT and the GRU network are adopted to extract the spatial features of the tensor. Then the multi-dimensional traffic feature fusion module concatenates the output of the multi-scale feature extraction and bidirectional spatiotemporal feature extraction module to a new tensor that can represent the spatial feature and temporal feature.

The new tensor is fed into the improved transformer layer. In the improved transformer layer, a multi-head attention mechanism is used to further extract the key features of the tensor. And then the output result is fed into the dual decoder module. In the dual decoder module, GRU is used to stack two fully connected layers decoder 1 and decoder 2. Finally, the abnormal score of the current time node is obtained by calculating the difference between the reconstruction value and the true value of the current. We will describe each module below in detail.

In Fig. 1, the abnormal score is as follows:

Figure 1. The Framework of The MSTSAD Model (This framework includes feature extraction, improved Transformer and bidirectional decoding modules, and finally calculates the anomaly score)



$$a\left(W\right)=\left(1-\alpha\right)\left\|W-AE_{1}\left(W\right)\right\|_{2}+\alpha\left\|W-AE_{2}\left(AE_{1}\left(W\right)\right)\right\|_{2} \tag{4}$$

Bidirectional Spatiotemporal Feature Extraction Module

Space feature extraction based on GAT: Due to the difficulty in obtaining the correlation between IIoT temporal data through prior methods, this paper considers the multivariate temporal data after sliding window partitioning. Each node represents a certain feature. The attention GNN is used to capture the relationship between adjacent nodes. Each node is a sequence: $s_{i}=\left\{s_{i,t}\left|t\in\left[o,n\right)\right.\right\}$, with a total of K nodes. And n represents the total number of timestamps, i.e. sliding window size, and also is the total number of multivariate temporal data features. The GAT layer calculates the feature representation of each node as:

$$s_{i}{}'=\sigma\left(\sum_{j=1}^{L}\alpha_{ij}s_{j}\right) \tag{5}$$

Among them, $s_{i}{}'$ is the output of i; σ is the sigmoid; α_{ij} is the weight, which measures the direct relations between the node i and j; L is adjacent node's amount of i. Attention score is expressed as:

$$e_{ij}=LR\left[w^{\mathrm{T}}\cdot\left(v_{i}\oplus v_{j}\right)\right] \tag{6}$$

$$a_{ij}=\frac{\exp\left(e_{ij}\right)}{\sum_{n=1}^{N}\exp\left(e_{in}\right)} \tag{7}$$

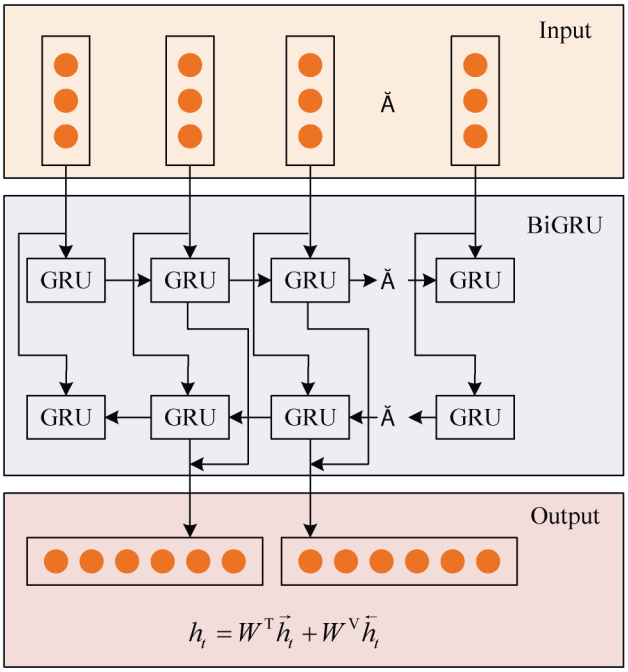
In eq (6) and (7), $w\in R^{2n}$ is a learnable column vector, N represents the dimension of the data, which is the total number of timestamps; LR is the nonlinear activation function LeakyReLU.

Fig. 2 shows the BiGRU processing process for the input data. And this process can be described as follows:

$$\begin{cases}\vec{h}_{t}=GRU\left(x_{t},\vec{h}_{t-1}\right)\\ \overleftarrow{h}_{t}=GRU\left(x_{t},\overleftarrow{h}_{t-1}\right)\\ h_{t}=W^{\mathrm{T}}\vec{h}_{t}+W^{\mathrm{V}}\overleftarrow{h}_{t}\end{cases} \tag{8}$$

In eq (8), the GRU is the nonlinear transformation of the input. W^{T} and W^{V} are the weight coefficients of $\vec{h}_{t}$ and $\overleftarrow{h}_{t}$.

Figure 2. The Processing of BiGRU (the BiGRU can simultaneously better capture the semantics and information in the text to improve the expression ability and performance of the model)



Multi Scale Feature Extraction

Fig. 3 shows details of the multi-scale memory residual. It is the core module in the temporal feature extraction of the input data. First, the input data is fed into three convolution layers with different convolution kernels, and then the results of each convolution module are fused with corresponding weights. The BN (Batch Normalization) layer is mainly used to reduce the time of training the model, and is the activation function. The convolution operation performs a sliding window on the input and calculates the weighted sum of the data within the window to obtain time-related local area characteristics of the data. LSTM network is used to capture the connection between the current moment of data and other moments. This allows the model to selectively ignore important information, which is useful for predicting and analyzing the future of temporal data.

Fig. 4 shows the structure after adding identity mapping to the multi-scale memory residual module. Adding identity mapping to each multi-scale memory residual module on the basis of the multi-scale memory module can prevent the occurrence of deep network gradient disappearance and achieve deep representation of the model.

Multidimensional Feature Fusion

The proposed model adopts an improved Transformer to fuse multidimensional features, allowing for mutual transmission and establishment of correlation relationships among multiple dimensional features, in order to adaptively strengthen key features and enhance the model's global representation ability for network traffic. The structure of the improved Transformer is shown in Fig. 5.

Firstly, the low-level sentence embedding representation $P = pooler\_out$ is obtained. Then, bidirectional spatiotemporal features and multi-scale features are extracted for word embedding. The

Figure 3. Multi-scale Memory Module (Residual connections add the output of the previous layer directly to the output of the current layer. This residual can be further learned by subsequent layers, allowing the network to better adapt to complex data distributions.)

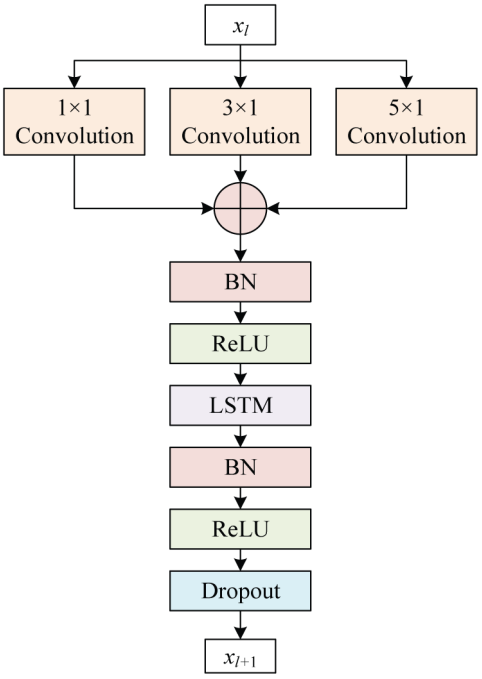


Figure 4. Multi-scale Memory Residual Module with Added Identity Mapping (It can prevent the occurrence of deep network gradient disappearance and achieve deep representation of the model)

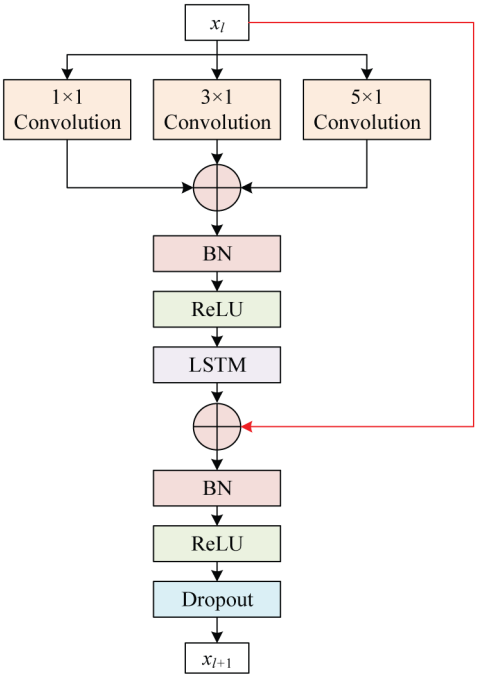
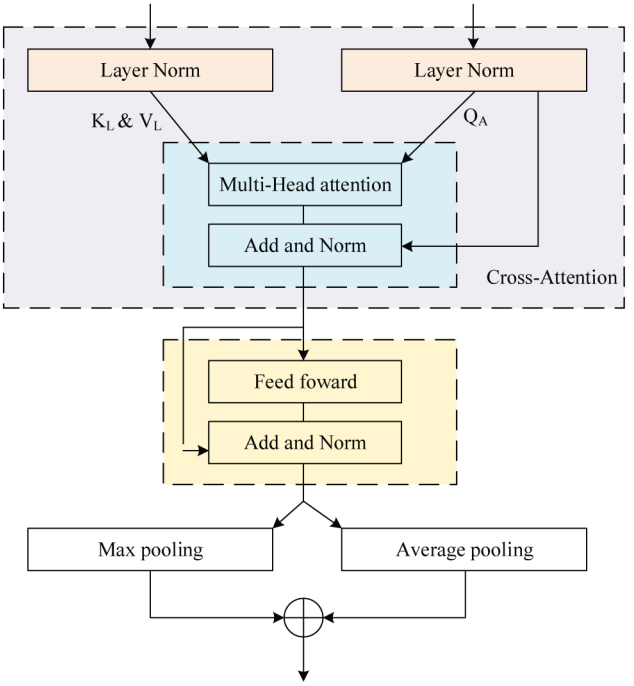


Figure 5. The Structure of an Improved Transformer (In the improved Transformer, multi-head attention mechanism is used to further extract the key features of the input data)



bidirectional spatiotemporal feature Table obtained is H_n, and the multi-scale features are represented as M_s. The obtained features from various dimensions are concatenated together to obtain the concatenated feature Z:

$$Z = \left[P, M_s, H_n\right]^{\mathrm{T}} \tag{9}$$

Next, an improved Transformer is used to adaptively fuse multi-dimensional features P, M_s, and H_n. The improved Transformer layer uses a multi-head attention mechanism to further extract the key features of the tensor. Next, we pass the output to the dual decoder module. Taking the operation process of the i-th head of the input P of the first channel as an example, the process of multi-head self-attention can be illustrated as: Initialize the relevant parameter matrix, perform self-attention operations on the row vectors P, M_s, and H_n of the fusion matrix Z, establish relevant connections, and obtain the output of the i-th head. Splice k heads to obtain the output o_1 of P. The details are as follows.

$$\begin{cases} Q_i = PW_i^q \\ K_i = PW_i^k \\ V_i = PW_i^v \end{cases} \tag{10}$$

$$H_i = Att\left(Q_i, K_i, V_i\right) = softmax\left(\frac{Q_i\left(K_i\right)^T}{\sqrt{d_{K_i}}}\right) \tag{11}$$

$$o_1 = Con\left(H_1, H_2, ..., H_k\right) \tag{12}$$

In eq (10)-(12), W_i^q, W_i^k, and W_i^v are the linear mapping weight matrices of P, and o_1 is the final obtained feature.

The fusion outputs o_2 and o_3 of M_s and H_n are obtained after the multi head self-attention fusion module, and the overall fusion output is $O = \left[o_1, o_2, o_3\right]$. Pooling operations have the advantages of suppressing noise. The results of pooling are combined as the final output, and the calculation process is:

$$\begin{cases} X_{i_{\max}}^{L\to A} = maxpooling\left(X_i^{L\to A}\right) \\ X_{i_{avg}}^{L\to A} = averagepooling\left(X_i^{L\to A}\right) \\ X_i^{L\to A} = Concat\left(X_{i_{\max}}^{L\to A}, X_{i_{avg}}^{L\to A}\right) \end{cases} \tag{13}$$

Dual Decoder Module

In the reconstruction stage, the decoder obtains the reconstruction value of the current timestamp by reconstructing latent variables, and performs anomaly diagnosis by calculating the difference between the reconstruction and true of the current.

$$\begin{cases} Dec\left(z\right) = Lin_2\left[Lin_1\left(TranEnc\left(z\right)\right)\right] \\ x_{recon} = Dec\left(z_t\right) \end{cases} \tag{14}$$

In eq (14), Lin is the fully connected layer. The dual decoder module in the overall framework mainly includes autoencoder AE1 and AE2. AE1 consists of Encoder and Decoder1, while AE2 consists of Encoder and Decoder2. Decoder1 and Decoder2 have the same network structure, and AE1 and AE2 share the Encoder. The encoding-decoding form is as follows.

$$\begin{cases} AE_1\left(x_t\right) = Decoder1\left(Encoder\left(x_t\right)\right) \\ AE_2\left(x_t\right) = Decoder2\left(Encoder\left(x_t\right)\right) \end{cases} \tag{15}$$

Here we use GRU (Gated Recurrent Unit) to stack these two fully connected layer decoders. Specific steps are as follows:

Decoder AE1: Decoder AE1 is responsible for processing the output from the improved Transformer layer. It maps these features to higher dimensional representations and generates intermediate states. The intermediate state is passed to decoder AE2.

Decoder AE2: Decoder AE2 further processes the intermediate states of Decoder AE1. It can be understood as a further decoding of a specific task.

Finally, decoder AE2 calculates the predicted value of the current time node and compares it with the true value. And then we can get the anomaly score of the current time node by calculating the difference between them.

Abnormal Score

In anomaly detection, we usually use temporal data to identify anomalies. The approach is to train a model to reconstruct predicted values, and then determine whether a point in the test temporal data is an anomaly by calculating the reconstruction error. Specific steps are as follows:

(1) Model training: We use historical temporal data to train a model to learn the characteristics and patterns of temporal data.
(2) Reconstruct predicted values: Using the trained model, we predict the test data and obtain the reconstructed temporal data.
(3) Calculate reconstruction error: Compare the reconstructed sequence to the original test data and calculate the error at each time point. Large errors may indicate anomalies.
(4) Setting thresholds: We use a non-parametric dynamic thresholding method to determine the threshold for anomalies. This means that the threshold adjusts adaptively based on changes in the data.
(5) Anomaly score: For each timestamp of the test data, we obtain its anomaly score, which is the reconstruction error. If the score exceeds the threshold, it is considered that there is an anomaly at that time point.

In summary, the MSTSAD model is trained to reconstruct the predicted value, and then the possibility of a point in the test temporal data being an anomaly is obtained by calculating reconstruction error. For timestamp of test data x_t, obtain the anomaly score a_t for that timestamp as an exception. This article adopts a nonparametric dynamic threshold method to determine the threshold η. When $a_t > \eta$, the corresponding anomaly label $y_t = 1$, otherwise $y_t = 0$.

Model Training

Adopting a two-stage training approach, AE1 and AE2 undergo self-training in the first stage to learn the periodic pattern of the input data. AE1 and AE2 are trained in adversarial training by re-inputting the reconstructed output of AE1 into AE2 for adversarial training. AE1 aims to deceive AE2 by reconstructing the data, while AE2 aims to correctly distinguish whether the data is reconstructed.

Autoencoder training: Autoencoder training takes place for Encoder, Decoder1, and Decoder2. In order to enable them to reconstruct normal data, the normal data is input into Decoder1 and Decoder2 simultaneously, and the result is reconstructed through their respective decoder networks. After the iterative training is completed, Encoders, Decoder1, and Decoder2 that can reconstruct normal data are obtained. Therefore, the main purpose of this stage is to enable AE1 and AE2 to learn the feature of normal data, minimizing the reconstruction loss on normal data, where lossAE1 and lossAE2 represent the reconstruction loss of AE1 and AE2 in self-training:

$$\begin{cases} LOSS_{AE_1} = \sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_1\left(x_{n,i}\right)\right]^2} \\ LOSS_{AE_2} = \sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_2\left(x_{n,i}\right)\right]^2} \end{cases} \tag{16}$$

Among them, $x_{n,i}$ is the value of the i-th feature in n timestamp x_n. $AE_1\left(x_{n,i}\right)$ and $AE_2\left(x_{n,i}\right)$ represent the values of input data $x_{n,i}$ after reconstruction by AE1 and AE2, respectively.

Adversarial training: The goal of training is for AE2 and AE1 to generate the predicted value. The reconstructed data generated from AE1 is compressed by Encoder to z, and then reconstructed by AE2 using adversarial training mechanisms. The goal of AE1 is to reduce the error between the result of AE1 and AE2, which indicates that the results of AE1 deceive AE2, causing AE2 to treat the reconstructed data of AE1 as real data, resulting in smaller reconstruction errors. The goal of AE2 is to maximize this difference, indicating that AE2 can correctly distinguish between real data and reconstructed data, resulting in significant reconstruction errors. The objectives of adversarial training are:

$$LOSS_A = \min_{AE_1}\max_{AE_2}\sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_2\left[AE_1\left(x_{n,i}\right)\right]\right]^2} \tag{17}$$

Based on the above analysis, both AE1 and AE2 aim to reduce the error between $x_{n,i}$ and the reconstruction values $AE_1\left(x_{n,i}\right)$ and $AE_2\left(x_{n,i}\right)$ to fully learn the potential features. In the adversarial training phase, the goal of AE1 is to reduce the final error $LOSS_{AE_2}$ between $x_{n,i}$ and the secondary reconstruction data $AE_2\left[AE_1\left(x_{n,i}\right)\right]$ after passing through the AE1 and AE2 modules. On the contrary, the goal of AE2 is to amplify this error as much as possible to achieve recognition purposes. For the training of the first and second stages, this article sets the weight ratio of the reconstruction error in the two stages, which will change with the increase of training iterations. The proportion of training losses $LOSS_{AE_1}$ and $LOSS_{AE_2}$ in the early stage is relatively large, but as the iterations increases, it will increase the proportion of adversarial training losses. The total training loss after combining the two stages is as follows for AE1 and AE2:

$$\begin{cases} LOSS_{AE_1} = \dfrac{1}{n}\sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_1\left(x_{n,i}\right)\right]^2} \\ \qquad\qquad + \left(1-\dfrac{1}{n}\right)\sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_2\left[AE_1\left(x_{n,i}\right)\right]\right]^2} \\ LOSS_{AE_2} = \dfrac{1}{n}\sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_2\left(x_{n,i}\right)\right]^2} \\ \qquad\qquad - \left(1-\dfrac{1}{n}\right)\sqrt{\sum_{i=1}^{k}\left[x_{\boldsymbol{n},i} - AE_2\left[AE_1\left(x_{n,i}\right)\right]\right]^2} \end{cases} \tag{18}$$

To alleviate the overfitting of the model to noise, this article introduces the VAE model. The encoder network adaptively generates the mean variance of the fitted data distribution, and then samples Gaussian noise to generate latent variables. This achieves robustness in reconstructing industrial temporal data and alleviates the overfitting to noise in the training data. Adding the VAE regularization term to the loss function during training is:

$$LOSS_{KL}\left(\delta,\gamma,x\right) = -D_{KL}\left[q_\gamma\left(z\middle|x\right)\middle\|p_\delta\left(z\right)\right] \tag{19}$$

In eq(19), δ and γ are parameters of the prior and posterior distribution p and q, respectively, while x and z are input and latent feature representations of the variational autoencoder.

The process is as follows:

Input: Training data sliding window $W_t = \left\{x_{t-n+1}, x_{t-n}, ..., x_{t-1}, x_t\right\}$
Test data sliding window $\hat{W}_t = \left\{\hat{x}_{t-n+1}, \hat{x}_{t-n}, ..., \hat{x}_{t-1}, \hat{x}_t\right\}$
Abnormal score weight α, number of iterations N
Encoder: input
W_t, $x^{ms} = MSTCN\left(W_t\right)$, $x^{ts} = BiGRU\left(GAT\left(W_t\right)\right)$, $x^{'} = Concat\left(\mathrm{Att}\left(x^{ms}, x^{ts}\right)\right)$
Decoder1:
Input z, $x_{recon}^{AE_1} = Linear_2\left(Linear_1\left(TranEnc\left(z\right)\right)\right)$
Decoder2:
Input z, $x_{recon}^{AE_2} = Linear_2\left(Linear_1\left(TranEnc\left(z\right)\right)\right)$
Output: Abnormal score of test data
1. Initialize network parameters in Encoder, Decoder1, and Decoder2
2. for n=1 to N do
3. for t=1 to T_1 do
4. $Encoder\left(x_t\right) \to z_t$
5. $Decoder1\left(z_t\right) \to AE_1\left(x_t\right)$
6. $Decoder2\left(z_t\right) \to AE_2\left(x_t\right)$
7. $Decoder2\left(Encoder\left(AE_1\left(x_t\right)\right)\right) \to AE_2\left(AE_1\left(x_t\right)\right)$
8. $$LOSS_{AE_1} = \frac{1}{n}\sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_1\left(x_{n,i}\right)\right]^2} + \left(1-\frac{1}{n}\right)\sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_2\left[AE_1\left(x_{n,i}\right)\right]\right]^2}$$
9. $$LOSS_{AE_2} = \frac{1}{n}\sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_2\left(x_{n,i}\right)\right]^2} - \left(1-\frac{1}{n}\right)\sqrt{\sum_{i=1}^{k}\left[x_{n,i} - AE_2\left[AE_1\left(x_{n,i}\right)\right]\right]^2}$$
10. $LOSS_{KL}\left(\delta, \gamma, x\right) = -D_{KL}\left[q_\gamma\left(z\middle|x\right)\middle\|p_\delta\left(z\right)\right]$, add VAE regularization term.
11. Update network parameters in Encoder, Decoder1, and Decoder2
12. end for
13. end for
14. for t=1 to T_2 do
15. $Decoder1\left(Encoder\left(\hat{x}_t\right)\right) \to AE_1\left(\hat{x}_t\right)$
16. $Decoder2\left(Encoder\left(AE_1\left(\hat{x}_t\right)\right)\right) \to AE_2\left(AE_1\left(\hat{x}_t\right)\right)$
17. $$a\left(\hat{x}_n\right) = \left(1-\alpha\right)\sqrt{\sum_{i=1}^{k}\left[\hat{x}_{n,i} - AE_1\left(\hat{x}_{n,i}\right)\right]^2} + \alpha\sqrt{\sum_{i=1}^{k}\left[\hat{x}_{n,i} - AE_2\left[AE_1\left(\hat{x}_{n,i}\right)\right]\right]^2}$$
18. end for

Anomaly Detection

After completing the construction and training, the network weights of AE1 and AE2 converged, and the test data containing anomalies was the input for detection. The final result is calculated by the predicted values and true labels. This article calculates the anomaly score of the sliding window is:

$$a\left(\hat{x}_n\right) = \left(1-\alpha\right)\sqrt{\sum_{i=1}^{k}\left[\hat{x}_{n,i} - AE_1\left(\hat{x}_{n,i}\right)\right]^2} + \alpha\sqrt{\sum_{i=1}^{k}\left[\hat{x}_{n,i} - AE_2\left[AE_1\left(\hat{x}_{n,i}\right)\right]\right]^2} \tag{20}$$

In eq (20), α is used to measure the proportion of reconstruction errors between AE1 and AE2; $\hat{x}_n$ represents the n-th sample in the test set. Meanwhile, this article adopts a nonparametric dynamic threshold method to determine the threshold η. When $a\left(\hat{x}_n\right) > \eta$, the corresponding anomaly label $y_n = 1$, otherwise $y_n = 0$.

EXPERIMENT

Experimental Environment

The information regarding the experimental environment during the model training process is shown in Table 1.

Data Set

This experiment was conducted on four real public datasets, and Table 2 shows the details of the datasets:

The ECG5000 dataset (Chen et al., 2015) was obtained from a patient in the MIT-BIH arrhythmia database and recorded their heartbeat. Each heartbeat is independently annotated by two cardiologists. The patient has severe congestive heart failure, and the grading value is automatically annotated, so it can be used as a benchmark for evaluating the proposed method. The duration of the ECG5000 dataset is 20 hours. Its data preprocessing is divided into two steps:

Table 1. Experimental Environment (This is the environment in which we conduct experiments)

Parameters	Configuration
CPU	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.80 GHz
GPU	NVIDIA GeForce GTX 3060
Memory	16G
OS	Windows 10
Programming Language	Python 3.8
Data process	Pandas 1.1.2
Machin learning	sklearn 0.23.2
DL	TensorFlow-GPU 2.5.0 keras

(1) Extract each heartbeat.
(2) Interpolate to make each heartbeat length equal.

ECG5000 dataset has 4500 training samples, and 500 testing samples. The dataset interpolates each heartbeat to the same length, which equals to 140 time steps, and each time step is 1 millisecond. According to the type of heartbeat, it is divided into 5 categories.

The GPW dataset (Lu et al., 2021) was created by Wei Gao and Tommy Morris as a network attack database from natural gas pipelines and water storage tanks. The dataset has higher dimensions and more attack types. Due to its collection and recording in real industrial control environments, the use of this dataset can better reflect the intrusion situation in actual industrial control systems, and so it is favored by researchers and industry insiders. The dataset consists of seven attack types in total.

The Occupancy dataset (Luis et al., 2016) uses statistical learning models to accurately detect office room occupancy, and it uses digital cameras to establish ground occupancy rates for model training. Occupancy contains 20560 instances and is commonly used in IIoT anomaly detection.

The SWaT dataset (Mathur, & Tippenhauer, 2016) was collected from a scaled-down water treatment test bench with 51 sensors. The SWaT dataset includes the data of 11 days. The data of 7 days was collected with the normal situation and 4 days under the situation of being attacked.

Setup

For the training and detection, a 1:1 ratio was used to divide the dataset into training and testing. Sliding window length selected for the experiment is $k = 25$; the sliding step size is 1. The iterations are $N = 100$, and the batch size is $batch = 128$, $\alpha = 0.9$. The latent variables and hidden state units of the model are adjusted through backpropagation algorithms and Adam optimizers (Chatterjee & Ahmed, 2022; Chen, et al., 2015; Deng & Hooi, 2021; Fang et al., 2021; Lai et al., 2021). Adam optimizer sets the learning rate as 0.001.

The hyperparameters of the comparative model follow the author's original settings. The experimental process includes comparative experiments, parameter sensitivity analysis experiments, ablation experiments, and case analysis, which will be described in the subsequent sections.

Evaluating Indicator

The performance evaluation of the MSTSAD model adopts Precision (P), Recall (R), and F1 score as indicators to measure the detection effect.

$$P = \frac{T_P}{T_P + F_P} \tag{21}$$

Table 2. The Information of Dataset

Dataset	Train Size	Test Size	Dimensions	Anomalies
ECG5000	4500	500	1	4.68%
GPW	135183	427617	8	13.30%
Occupancy	8143	9752	7	10.52%
SWaT	568031	378688	51	11.98%

Table 3. Confusion Matrix (Table 3 shows the results of confusion Matrix of the MSTSAD model)

		Prediction	
		1	**0**
Actual	**1**	T_P	F_N
	0	F_P	T_N

$$R = \frac{T_P}{T_P + F_N} \tag{22}$$

$$F1 = \frac{2PR}{P + R} \tag{23}$$

The meanings of the symbols above are shown in Table 3.

Comparison With the State-of-the-Art Methods

To measure the overall performance of the MSTSAD model, advanced methods in IIoT temporal data anomaly detection were selected as baseline comparison methods: Informer (Zhou et al., 2021), DAGMM (Deep Autoencoding Gaussian Mixture Model, Bo et al., 2018), ODIN AD (Out-of-Distribution Neural networks for Anomaly Detection, Zangrando et al., 2023), USAD (Unsupervised Anomaly Detection for Multivariate Temporal data, Audibert et al., 2020), Anomaly-PTG (Anomaly Parallel Transformer GRU, Li et al., 2022), TFAD (Temporal data Architecture With Time-frequency Analysis for Anomaly Detection, Zhang et al., 2022), MAD-GAN (Multivariate Anomaly Detection for Temporal data Data with Generative Adversarial Networks, Li et al., 2019).

Compare and analyze the above methods using the ECG 5000, GPW, Occupancy, and SWaT datasets. Under the same experimental conditions, the results calculated using different models on the ECG 5000 dataset are listed in Fig. 6 and Table 4. It demonstrates the MSTSAD model achieves the highest F1 score with 99.91% among all the models on the ECG 5000 dataset. Compared to Informer, the F1 score increases by about 1.10%. The results on the GPW dataset are listed in Fig. 7 and Table 5. It demonstrates that the F1 score of the MSTSAD model achieves the best result with 80.03% and increases by 1.37% compared to Informer. The results of the Occupancy dataset are listed in Fig. 8 and Table 6. It demonstrates that The F1 score of the MSTSAD model increases by 2.43% compared to Informer. The results on the SWaT dataset are listed in Fig. 9 and Table 7. It demonstrates that the F1 score of the MSTSAD model increases by 2.53% compared to MAD-GAN model. Overall, the proposed MSTSAD model achieved the best results on all datasets.

Since Informer adopts an innovative self-attention mechanism and generative decoder, which can capture long-term dependencies between the temporal data, it can greatly improve the prediction performance. However, the Informer does not have a designed structure to extract the spatial features of the data, so there is a bottleneck in its anomaly detection accuracy. The proposed MATSAD model introduces a GAT and a GRU network to further extract the spatial features of the data and adopts an improved Transformer to achieve multi-dimensional feature fusion. To balance global and local features to avoid issues such as information loss, we designed a combined pooling structure. And the MSTSAD model also combines adversarial training with a variational autoencoder to amplify abnormal reconstruction errors effectively and solves the problem of low model performance caused by training data noise in traditional autoencoder models.

Figure 6. Results on The ECG5000 Dataset (shows the results of all the models on the ECG5000 dataset)

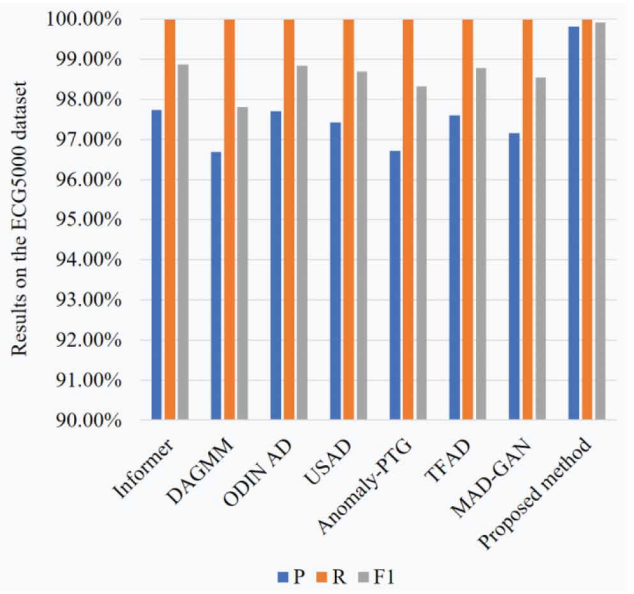


Table 4. Results on The ECG5000 Dataset (Table 4 shows the results of all the models on the ECG5000 dataset)

Model	Indicator		
	P	R	F1
Informer	97.74%	99.99%	98.87%
DAGMM	96.69%	99.99%	97.81%
ODIN AD	97.71%	99.99%	98.84%
USAD	97.42%	99.99%	98.69%
Anomaly-PTG	96.71%	99.99%	98.33%
TFAD	97.60%	99.99%	98.78%
MAD-GAN	97.16%	99.99%	98.55%
MSTSAD(ours)	**99.81%**	**99.99%**	**99.91%**

Ablation Experiment

To assess the effectiveness of individual modules in the MSTSAD model, the ablation experiment was conducted to compare the impact of different modules on the overall performance of the model as follows.

Model 1: Only considers multi-scale temporal features and does not consider bidirectional spatiotemporal features.

Model 2: Only considers bidirectional spatiotemporal features and does not consider multi-scale temporal features.

Model 3: The spatiotemporal module only retains the spatial feature module GAT, while removing the temporal feature module BiGRU.

Figure 7. Results on The GPW Dataset (shows the results of all the models on the GPW dataset)

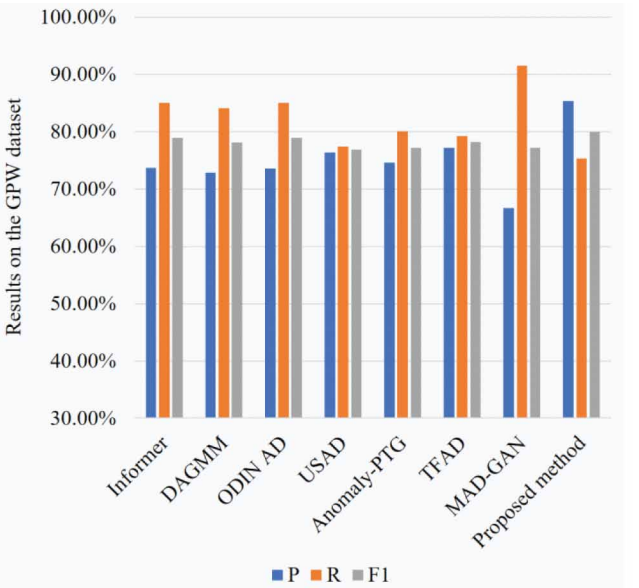


Table 5. Results on The GPW Dataset (Table 5 shows the results of all the models on the GPW dataset)

Model	Indicator		
	P	R	F1
Informer	73.66%	85.07%	78.95%
DAGMM	72.87%	84.16%	78.11%
ODIN AD	73.64%	85.04%	78.93%
USAD	76.36%	77.47%	76.91%
Anomaly-PTG	74.62%	80.09%	77.25%
TFAD	77.19%	79.25%	78.21%
MAD-GAN	66.72%	91.60%	77.20%
MSTSAD(ours)	**85.35%**	**75.34%**	**80.03%**

Model 4: The spatiotemporal module only retains the temporal feature module BiGRU, while removing the spatial feature module GAT.
Model 5: Only adversarial training is used, and the variational autoencoder architecture is not used.
Model 6: Only performs simple concatenation of multi-dimensional features without using Transformer for encoding.

Fig. 10 and Table 8 show the results of ablation experiments on the GPW dataset. Fig. 11 and Table 9 show the results of ablation experiments on the Occupancy dataset. The results indicate that eliminating any module from the model leads to a reduction in the precision of P, R, and F1-score. Among all modules, the index values of Model 2, Model 1, and Model 3 have relatively large decreases, indicating that considering multi-scale temporal features has the greatest improvement on the model.

Figure 8. Results on The Occupancy Dataset (shows the results of all the models on the Occupancy dataset)

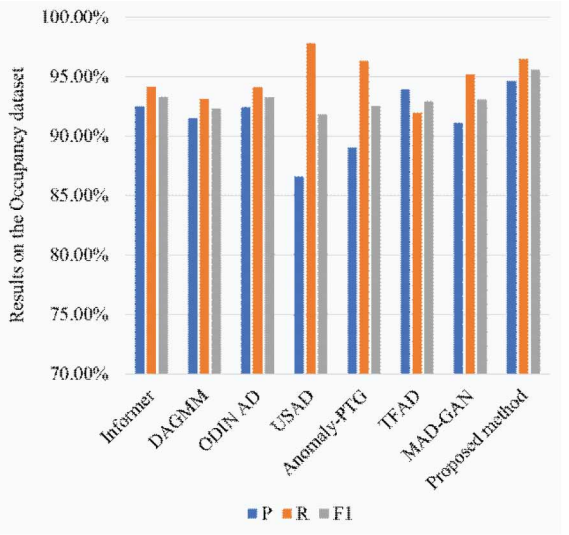


Table 6. Results on The Occupancy Dataset (Table 6 shows the results of all the models on the Occupancy dataset)

Model	Indicator		
	P	R	F1
Informer	92.49%	94.14%	93.30%
DAGMM	91.50%	93.13%	92.30%
ODIN AD	92.46%	94.11%	93.27%
USAD	86.59%	97.79%	91.85%
Anomaly-PTG	89.06%	96.29%	92.54%
TFAD	93.91%	91.97%	92.93%
MAD-GAN	91.13%	95.16%	93.10%
MSTSAD(ours)	**94.67%**	**96.49%**	**95.57%**

Secondly, considering the bidirectional spatiotemporal and temporal feature modules BiGRU, the improvement of the model is relatively significant. And it demonstrated that all the modules of GAT, variational autoencoder architecture, and Transformer encoding play a great role in improving the accuracy of model prediction.

Hyperparameter Experiment

Learning Rate Analysis

The experiment will be conducted with changing learning rates on the Occupancy dataset. The results are as follows when the learning rates are set to different values.

The result of changing learning rate of the MSTSAD model on Occupancy dataset is shown in Fig. 12. In Fig. 12, we set it as 0.1, 0.01, 0.001, 0.0001 and 0.00001, respectively. As the learning rate continues to decrease, the model prediction results become more accurate. When the value of the learning rate is 0.001, the MSTSAD model shows the best performance. And then as the learning

Figure 9. Results on The SWaT Dataset (shows the results of all the models on the Occupancy dataset)

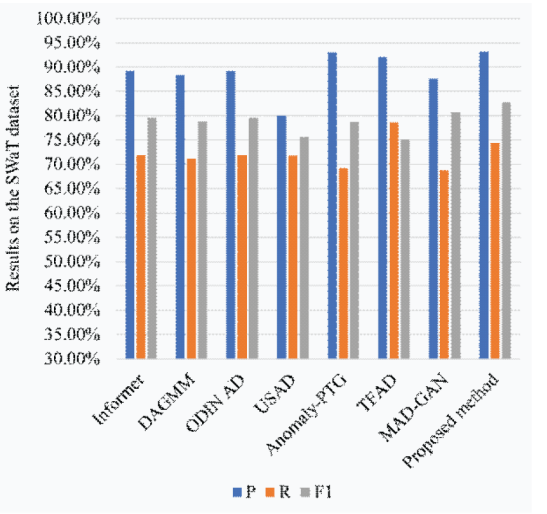


Table 7. Results on The SWaT Dataset (Table 7 shows the results of all the models on the SWaT dataset)

Model	Indicator		
	P	R	F1
Informer	89.24%	71.91%	79.64%
DAGMM	88.28%	71.14%	78.79%
ODIN AD	89.21%	71.89%	79.62%
USAD	80.01%	71.73%	75.67%
Anomaly-PTG	93.01%	69.17%	78.67%
TFAD	92.08%	78.57%	75.18%
MAD-GAN	87.58%	68.79%	80.69%
MSTSAD(ours)	**93.19%**	**74.38%**	**82.73%**

rate decreases, the model performance drops. Therefore, we set the learning rate of the MSTSAD model as 0.001.

Sensitivity Analysis of Sliding Windows

The experiment will be conducted for measuring the performance of the MSTSAD model with different sliding window lengths. The results of the MSTSAD model are as follows.

Fig. 13 shows the result of the different lengths of the sliding window on the MSTSAD model. In Fig. 13, the performance varies when setting the length of the sliding window as different values. When it is set to 25, the model exhibits the best performance.

The Impact of Different Layers in the Transformer-Encoder

Taking the Occupancy dataset as an example, experimental analysis will be conducted with some Transformer encoder layers. The experimental results of the MSTSAD model are as follows.

Figure 10. Results of Ablation Experiment on GPW Dataset (shows the results of the ablation experiment of the proposed MSTSAD models on the GPW dataset)

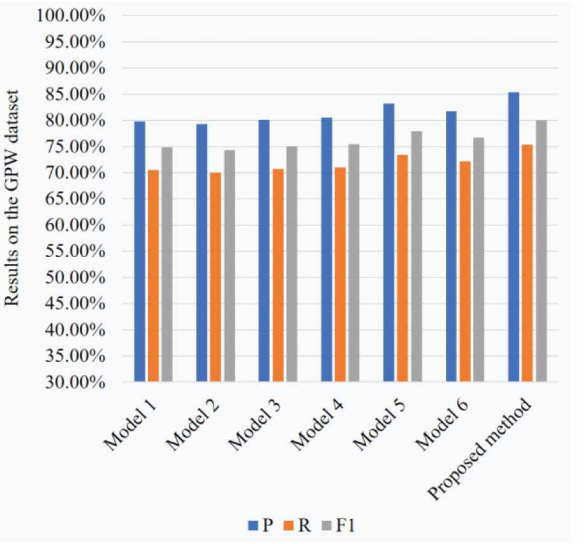


Table 8. Results of Ablation Experiment on GPW Dataset (Table 8 shows the results of the ablation experiment of the proposed MSTSAD models on the GPW dataset)

Model	Indicator		
	P	R	F1
Model 1	79.84%	70.48%	74.87%
Model 2	79.28%	69.98%	74.34%
Model 3	80.08%	70.68%	75.08%
Model 4	80.49%	71.05%	75.48%
Model 5	83.17%	73.42%	77.99%
Model 6	81.81%	72.21%	76.71%
MSTSAD(ours)	**85.35%**	**75.34%**	**80.03%**

Fig. 14 shows the result of the different transformer encoder layers of the MSTSAD model. In Fig. 14, the performance of the model varies when layers in the Transformer are set to different values. When it is set to 4, the model exhibits the best performance.

Analysis of Residual Network Layers

Taking the Occupancy dataset as an example, experimental analysis will be conducted with some residual network layers. The results of the MSTSAD model are shown in Fig. 15 when the residual network layers are set to different values.

Fig. 15 shows the result of the different residual network layers of the MSTSAD model. In Fig. 15, we set the residual network layers as 1, 2, 3, 4 and 5, respectively. As the layers continue to increase, the model prediction results become more accurate. When the residual network layers are set to 3, the

Figure 11. Results of Ablation Experiment on Occupancy Dataset (shows the results of the ablation experiment of the proposed MSTSAD models on the Occupancy dataset)

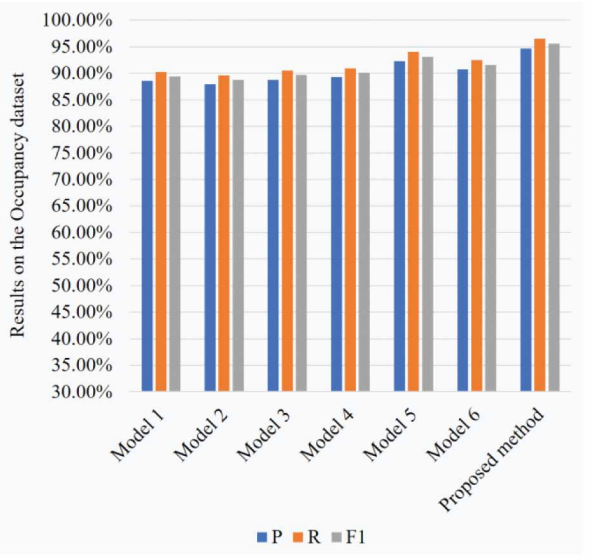


Table 9. Results of Ablation Experiment on Occupancy Dataset (Table 9 shows the results of the ablation experiment of the proposed MSTSAD models on the Occupancy dataset)

Model	Indicator		
	P	R	F1
Model 1	88.56%	90.27%	89.41%
Model 2	87.94%	89.63%	88.77%
Model 3	88.82%	90.53%	89.66%
Model 4	89.28%	91.00%	90.13%
Model 5	92.26%	94.03%	93.13%
Model 6	90.74%	92.49%	91.60%
MSTSAD(ours)	**94.67%**	**96.49%**	**95.57%**

MSTSAD model shows the best performance. And then as the residual network layers increases, the model performance drops. Therefore, the residual network layers of the MSTSAD model are set to 3.

Computing Performance Analysis

We study the computational performance of the MSTSAD model and compare it with other baseline models in this section. We measure the running memory and the time spent on the training process and inference process on ECG5000 and SWaT datasets. Table 10 shows the computing performance result of all the models. Mem means the running memory of the model while training. Tr-time and Inf-time mean the training time and inference time of the MSTSAD model, respectively. Compared to ODIN AD model on ECG5000 dataset, the MSTSAD model increases running memory by about 0.29 MB. The training time and inference time of the model also increases by 15 seconds and 1.22

Figure 12. The Influence of Learning Rate on the Proposed Model (shows the results of the different learning rate of the MSTSAD model on the Occupancy dataset)

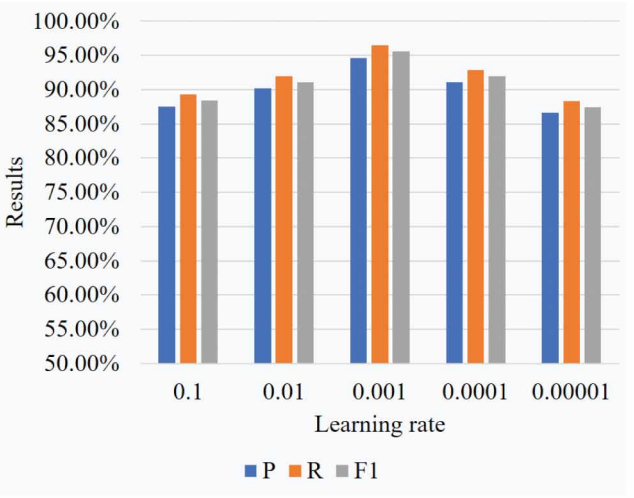
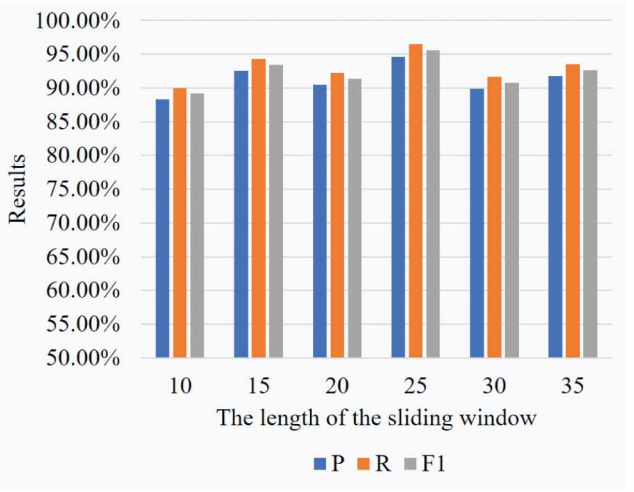


Figure 13. The Influence of the Sliding Window on the Proposed Model (shows the results of the different sliding window of the MSTSAD model on the Occupancy dataset)



seconds, respectively. Compared to the best-performing baseline Informer on SWaT dataset, the MSTSAD model increases running memory by about 0.65 MB. The training time and inference time of the model also increases by 140 seconds and 1.14 seconds, respectively.

Since the MSTSAD model applies a graph attention module and the improved transformer network architecture to extract the spatiotemporal features of the data, it leads to the network architecture of the MSTSAD model being more complex than the baselines. In summary, the proposed MSTSAD model achieves the best prediction performance among the baseline models, but the model also increases the time of running memory and training and inference processes to some extent.

Figure 14. The Impact of Transformer Encoder Layers of the MSTSAD Model (shows the results of the different encoder layers of the MSTSAD model on the Occupancy dataset)

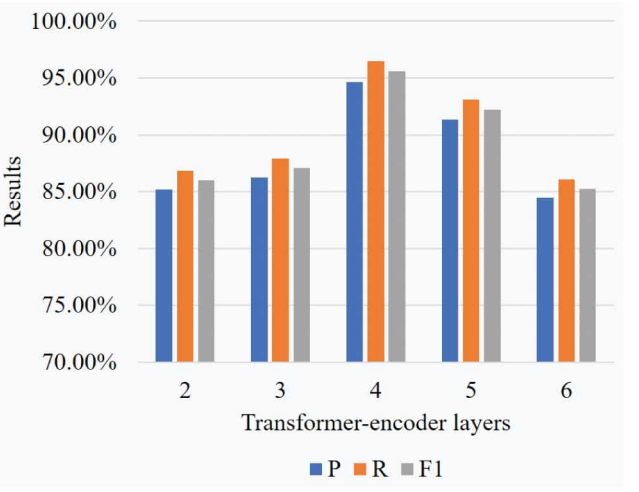
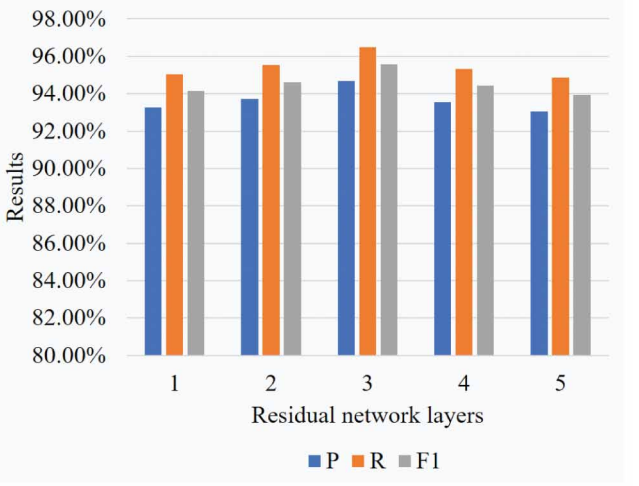


Figure 15. The Impact of Residual Network Layers on the MSTSAD Model (shows the results of the different residual networks layers of the MSTSAD model on the Occupancy dataset)



CONCLUSION

Aiming at the problems of information loss, difficulty in balancing spatial and temporal features, and influence of training data noise in current IIoT temporal data analysis methods, a fusion Transformer and adversarial training IIoT temporal data anomaly detection method is proposed. The performance of the MSTSAD model was validated through experiments. The results indicate that feature extraction by combining GAT and BiGRU can balance spatial and temporal features. Combining convolution and LSTM can capture multi-scale contextual information. The residual networks for deep feature extraction can effectively prevent phenomena such as vanishing gradients, exploding gradients, overfitting, and network degradation. The use of an improved Transformer can achieve effective fusion

Table 10. The Computing Performance Result of All the Models (Table 10 shows the results of the computing performance of all the model on the ECG5000 and SWaT dataset)

Dataset	ECG5000			SWaT		
Model	Mem (MB)	Tr-time (s)	Inf-time(s)	Mem (MB)	Tr-time (s)	Inf-time (s)
Informer	0.68	42	2.54	1.43	580	5.62
DAGMM	0.74	49	2.41	1.48	660	6.65
ODIN AD	0.57	40	2.23	1.55	790	6.87
USAD	0.69	41	2.87	1.68	830	7.64
Anomaly-PTG	0.85	53	2.96	1.73	880	8.56
TFAD	0.95	58	3.14	2.12	950	9.73
MAD-GAN	1.07	62	3.18	2.26	1020	11.34
MSTSAD(ours)	0.86	55	2.45	2.08	720	6.76

of multidimensional features. The variational autoencoder combined with adversarial training can effectively solve the problem of model susceptibility to training data noise, thereby greatly improving the detection performance of the model. However, due to the introduction of the GAT network and the improved transformer network architecture, the model complexity has increased slightly so that the MSTSAD model will increase running memory and the time during the training process and inference process to some extent.

Future work will focus on researching methods that combine hyperparameter optimization algorithms with the proposed model, in order to fully leverage the predictive and anomaly detection performance of the model. And we will optimize the model to reduce some parameters so that it can reduce training time and the running memory to some extent. In addition, further exploration of better DL models and optimization algorithms to improve model performance is also the direction of future efforts, and we will also focus on how to apply the proposed method in security and safety in the IIoT field.

CONFLICTS OF INTEREST

The authors of this publication declare there are no competing interests.

FUNDING STATEMENT

This work was supported by the Natural Science Project of Shaanxi Provincial Department of Education (No.22JK0616).

PROCESS DATES

Received: January 17, 2024, Revision: March 13, 2024, Accepted: March 28, 2024

CORRESPONDING AUTHOR

Correspondence should be addressed to Wendong Wang; yadxty@163.com

REFERENCES

Abdelrahman, O., & Keikhosrokiani, P. (2020). Assembly line anomaly detection and root cause analysis using machine learning. *IEEE Access : Practical Innovations, Open Solutions*, *8*(3), 189661–189672. doi:10.1109/ACCESS.2020.3029826

Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (2020). USAD: Unsupervised anomaly detection on multivariate temporal data. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,* 3395-3404. doi:10.1145/3394486.3403392

Balaji, S., & Sankaranarayanan, S. (2022). Hybrid deep-generative adversarial network based intrusion detection model for Internet of Things using binary particle swarm optimization. *IJEER*, *10*(4), 948–953. doi:10.37391/ijeer.100432

Chatterjee, A., & Ahmed, B. S. (2022). IoT anomaly detection methods and applications: A survey. *Internet of Things : Engineering Cyber Physical Human Systems*, *19*, 100568. doi:10.1016/j.iot.2022.100568

Chen, X., Deng, L., Huang, F., Zhang, C., Zhang, Z., Zhao, Y., & Zheng, K. (2021a). DAEMON: Unsupervised anomaly detection and interpretation for multivariate temporal data. *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2225-2230.

Chen, Y. P., Hao, Y., & Rakthanmanon, T. (2015). ECG5000: A general framework for never-ending learning from temporal data streams. *Data Mining and Knowledge Discovery*, *29*(6), 1622–1664. doi:10.1007/s10618-014-0388-4

Chen, Z., Chen, D., Yuan, Z., Cheng, X., & Zhang, X. (2021b). Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. *IEEE Internet of Things Journal*, *9*(12), 9179–9189. doi:10.1109/JIOT.2021.3100509

Deng, A., & Hooi, B. (2021). Graph neural network-based anomaly detection in multivariate temporal data. *IEEE Internet of Things Journal*, *12*(5), 35–44.

Ennaji, S., El Akkad, N., & Haddouch, K. (2023). i-2NIDS novel intelligent intrusion detection approach for a strong network security. [IJISP]. *International Journal of Information Security and Privacy*, *17*(1), 1–17. doi:10.4018/IJISP.317113

Fang, W., Chen, Y., & Xue, Q. (2021). Survey on research of RNN-based spatio-temporal sequence prediction algorithms. *Journal of Big Data*, *3*(3), 97–108. doi:10.32604/jbd.2021.016993

Garg, A., Zhang, W., Samaran, J., Savitha, R., & Foo, C. (2022). An evaluation of anomaly detection and diagnosis in multivariate temporal data. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(6), 2508–2517. doi:10.1109/TNNLS.2021.3105827 PMID:34464278

Guendouz, M., & Amine, A. (2023). A new feature selection method based on dragonfly algorithm for android malware detection using machine learning techniques. [IJISP]. *International Journal of Information Security and Privacy*, *17*(1), 1–18. doi:10.4018/IJISP.319018

Lai, K., Zha, D., Xu, J., & Zhao, Y. (2021). Revisiting temporal data outlier detection: Definitions and benchmarks. *Proc of the 35th Conf on Neural Information Processing Systems Datasets and Benchmarks Track*. Cambridge, MA*:* MIT Press, 38-43.

Li, D., Chen, D., Shi, L., Jin, B., Goh, J., & Ng, S. (2019). MAD-GAN: Multivariate anomaly detection for temporal data with generative adversarial networks. *International Conference on Artificial Neural Networks,* Cham*,* Springer, 703-716.

Li, G., & Jung, J. J. (2022). Deep learning for anomaly detection in multivariate temporal data: Approaches, applications, and challenges. *Information Fusion*, *91*, 93–102. doi:10.1016/j.inffus.2022.10.008

Li, G., Yang, Z., Wan, H., & Li, M. (2022). Anomaly-PTG: A temporal data data-anomaly-detection transformer framework in multiple scenarios. *Electronics (Basel)*, *11*(23), 3955. doi:10.3390/electronics11233955

Lu, K., Zeng, G., Luo, X., Weng, J., Luo, W., & Wu, Y. (2021). Evolutionary deep belief network for cyber-attack detection in industrial automation and control system. *IEEE Transactions on Industrial Informatics*, *17*(11), 7618–7627. doi:10.1109/TII.2021.3053304

Luis, M., Candanedo, & Veronique, F. (2016). Occupancy: Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Building*, *112*(1), 28–39.

Mathur, A. P., & Tippenhauer, N. O. (2016). SWaT: A water treatment testbed for research and training on ICS security. 2016 international workshop on cyber-physical systems for smart water networks (CySWater). IEEE, 31-36.

Meng, H., Zhang, Y., Li, Y., & Zhao, H. (2020). *Spacecraft anomaly detection via transformer reconstruction error. Lecture Notes in Electrical Engineering, 622, 351-362*. Springer.

Park, D., Hoshi, Y., & Kemp, C. C. (2017). A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robotics and Automation Letters*, *3*(3), 1544–1551. doi:10.1109/LRA.2018.2801475

Shen, L., Li, Z., & Kwok, J. (2020). Timeseries anomaly detection using temporal hierarchical one-class network. *Advances in Neural Information Processing Systems*, *33*(5), 13016–13026.

Song, J., Zhang, Z., Zhao, K., Xue, Q., & Gupta, B. B. (2023). A novel CNN-LSTM fusion-based intrusion detection method for industrial Internet. [IJISP]. *International Journal of Information Security and Privacy*, *17*(1), 1–18. doi:10.4018/IJISP.325232

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., & Pei, D. (2019). Robust anomaly detection for multivariate temporal data through stochastic recurrent neural network. *The 25th ACM SIGKDD International Conference. ACM*, 15-20.

Tran, D. H., Nguyen, V. L., Nguyen, H., & Jang, Y. M. (2022). Self-Supervised learning for time-series anomaly detection in industrial Internet of Things. *Electronics (Basel)*, *11*(14), 2146. doi:10.3390/electronics11142146

Wang, Y., Shen, Y., Mao, S., Chen, X., & Zou, H. (2019). LASSO and LSTM integrated temporal model for short-term solar intensity forecasting. *IEEE Internet of Things Journal*, *6*(2), 2933–2944. doi:10.1109/JIOT.2018.2877510

Wu, N., Green, B., Ben, X., & O'Banion, S. (2020). Deep transformer models for temporal data forecasting: The influenza prevalence case. *CS -. Machine Learning*, *7*(14), 515–527.

Wu, Z. (2024). An abnormal external link detection algorithm based on multi-modal fusion. [IJISP]. *International Journal of Information Security and Privacy*, *18*(1), 1–15. doi:10.4018/IJISP.337894

Xia, X., Pan, X., Li, N., He, X., Ma, L., Zhang, X., & Ding, N. (2022). GAN-Based anomaly detection: A review. *Neurocomputing*, *49*(3), 497–535. doi:10.1016/j.neucom.2021.12.093

Zangrando, N., Fraternali, P., Torres, R. N., & Petri, M. Pinciroli, Vago, N. O., & Herrera, S. (2023). ODIN AD: A framework supporting the life-cycle of temporal data anomaly detection applications. In: Guyet, T., Ifrim, G., Malinowski, S., Bagnall, A., Shafer, P., Lemaire, V. (eds) Advanced Analytics and Learning on Temporal Data. AALTD 2022. Lecture Notes in Computer Science, 13812, 181-196.

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., & Chawla, N. (2019). A deep neural network for unsupervised anomaly detection and diagnosis in multivariate temporal data. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 1409–1416. doi:10.1609/aaai.v33i01.33011409

Zhang, C. L., Zhou, T., Wen, Q. S., & Sun, L. (2022). TFAD: A decomposition temporal data anomaly detection architecture with time-frequency analysis. *Meeting31st ACM International Conference on Information and Knowledge Management (CIKM), Atlanta*, GA, 17-22.

Zhao, H., Wang, Y., Duan, J., Huang, C., Cao, D., Tong, Y., Xu, B., Bai, J., Tong, J., & Zhang, Q. (2020). Multivariate time-series anomaly detection via graph attention network. *2020 IEEE International Conference on Data Mining (ICDM)*, 841-850. doi:10.1109/ICDM50108.2020.00093

Zhong, Z., Zhao, Y., Yang, A., Zhang, H., Qiao, D., & Zhang, Z. (2022). Industrial robot vibration anomaly detection based on sliding window one-dimensional convolution autoencoder. *Shock and Vibration*, *2022*(3), 224–235. doi:10.1155/2022/1179192

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. *AAAI Conference on Artificial Intelligence,* 11106-11115. doi:10.1609/aaai.v35i12.17325

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. *In International conference on learning representations,* 487-498.

Yuan Tian, Master of Computer Application Technology, Lecturer. Graduated from Xidian University in 2013. Worked in Yan'an University. Her research interests include Internet of Things and cloud computing.

Wendong Wang, Master of Computer Science, Professor. Graduated from Xi'an Shiyou University in 2006. Worked in Yan'an University. His research interests include big data and cloud computing.

Jingyuan He, Master of Computer Software and Theory, Lecturer. Graduated from Xi'an Technological University in 2013. Worked in Yan'an University. Her research interests include big data and data mining.