Advanced Engineering Informatics 56 (2023) 101949

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Full length article

BTAD: A binary transformer deep neural network model for anomaly detection in multivariate time series data☆

Mingrui Ma [a,1], Lansheng Han [a,*,2], Chunjie Zhou [b,3]

[a] Hubei Key Laboratory of Distributed System Security, Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, 430074, China

[b] The Key Laboratory of Ministry of Education for Image Processing and Intelligent Control, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, 430074, Hubei, China

ARTICLE INFO

Keywords:
Multivariate time series data
Bi-Transformer model
Model-agnostic meta learning
Adaptive multi-head attention mechanism
Self-conditioning mechanism

ABSTRACT

In the context of big data, if the task of multivariate time series data anomaly detection cannot be performed efficiently and accurately, it will bring great security risks to industrial systems. However, fast model inference requirements, unlabeled datasets and excessively long time series make it a challenging problem to build an accurate and fast anomaly detection model. In this paper, we propose an unsupervised Bi-Transformer anomaly detection method (BTAD) for multivariate time series data, which uses Bi-Transformer structure to extract dataset association features, and uses an improved adaptive multi-head attention mechanism to infer trends in each meta-dimension of multivariate time series data in parallel. The modified Decoder structure prevents the reconstructed output of BTAD from being disturbed by the input information. Self-conditioning mechanism could enhance the robustness to noisy data, and improve model's generalization ability. Experiments show that BTAD could outperform other models in detection performance and training efficiency. Taking NAB dataset as an example, the AUC and $F1$ of BTAD are increased by more than 4.78% and 1.40% separately. Finally, we look forward to the future development trend of BTAD, and put forward the corresponding improvement ideas.

1. Backgrounds and motivations

Modern industrial systems generate large amounts of high-dimensional sensor data at all times, which may contain many anomalous data. If the anomalies are not explored, detected and fixed in time, it is likely to jeopardize the normal operation of the system and even lead to the consequences of system downtime. Anomaly detection and diagnosis can find system defects and deficiencies in time, so as to ensure the normal operation of system functions, maintain the stability and enhance the robustness of the system. Time series datasets are the result of different types of engineering components (sensors, server clusters, robots, etc.) interacting with natural environments (rivers, mountains, atmospheric pressure, etc.), humans or other systems, which contain both time trends and a large amount of random interference terms. Therefore, effectively capturing time series trends and extracting series features from data containing noise is the key to the multivariate time series anomaly detection task. However, it is always a challenging problem to establish a system that can quickly and accurately locate the anomaly, because the datasets in this field have the characteristics of large data volume, no label, large data fluctuation range, difficulty in capturing anomalies, unbalanced data distribution [1]. Furthermore, anomalies are often caused by the joint action of multiple variables, rather than relying on a single variable. For example, many data-driven industries, including but not limited to Internet of Things (IoT), Autonomous Driving Systems (ADS), robotics and source management generate massive amount of volatile, multimodal, distributed time series datasets [2]. In the context of Industry 4.0, the geographic distance of distributed system databases

☆ This work is supported by National Natural Science Foundation of China: 6217071437, 62072200, 62127808, and National Key Research and Development Program of China: 2022YFB3103403.

* Corresponding author.
E-mail addresses: m202271767@hust.edu.cn (M. Ma), hanlansheng@hust.edu.cn (L. Han), cjiezhou@hust.edu.cn (C. Zhou).

[1] His research interests include information security, network security, neural network, deep learning and artificial intelligence.

[2] His research interests includes malicious code, big data security and network security, and has published more than 50 papers in various well-known journals and conferences.

[3] His research interests include safety and security control of industrial control systems, theory and networked control systems, and artificial intelligence applications of Industrial Internet system.

https://doi.org/10.1016/j.aei.2023.101949
Received 2 November 2022; Received in revised form 20 February 2023; Accepted 15 March 2023
Available online 27 March 2023

and the gradual rise of federated learning paradigms have resulted in only a strictly limited amount of available data in the aforementioned systems. In addition, next-generation system applications require rapid inference speed to recover from system anomalies, optimize Quality of Service (QoS), and enhance service reliability [3].

In this context, anomaly detection models based on supervised methods are difficult to adapt to new anomaly detection tasks, because they must rely on clear labels in the training dataset to set decision boundaries or divide confidence intervals. The above mentioned challenges have given rise to a large number of unsupervised anomaly detection studies. In fact, the data imbalance becomes a prerequisite for unsupervised anomaly detection models, because only in this case can the model effectively learn the normal data's feature distribution from a large number of normal data and a small number of anomaly data. According to the timeline, we divide the studies of anomaly detection in multivariate time series data into three categories:

- anomaly detection based on statistical methods;
- anomaly detection based on machine learning algorithms;
- anomaly detection based on deep learning and neural networks.

Most of the statistical methods are incapable of capturing volatility long trend time-series as they usually rely on wavelet theory [4] or Hilbert transform [5], etc. to model data, and are rarely applied to anomaly detection in high-order multivariate time series datasets [6]. Principal Component Analysis (PCA) [7] and Markov chain [8] are also available statistical methods for modeling time series distribution.

Another problem with statistical methods is that they generally suffer from "performance bottlenecks". Taking PCA as an example, it can effectively reduce the dimension of multivariates, and find out several variables that have the greatest impact on the stable operation of the system. However, it ignores the fact that many anomalies are caused by numerical fluctuations of variables (influencing factors) with little correlation in the stable operation of the system. PCA cannot address the root cause of this problem, because PCA will always prioritize the "principal components" and ignore "non-principal components".

In view of the shortcomings of statistical methods, machine learning algorithms such as Regression model [9], Support Vector Machine (SVM) [10] and K-means clustering [10] have been applied to the task of modeling the distribution of time series data. Compared with statistical methods, These methods have improved model decision performance, but are still limited by information memory capacity for modeling time trends.

With the development of Deep Neural Network (DNN) [11], it is undeniable that most of the advanced contemporary models adopt some form of DNN. Researchers find that Recurrent Neural Networks (RNN) in NLP tasks have excellent contextual memory and feature extraction capabilities for anomaly detection tasks with serialized data.

Long-Short-Term-Memory (LSTM) [12] could effectively model long-term and short-term memory dependencies, and has been widely used in various unsupervised anomaly detection tasks. However, due to the linear structure and many logic gates, LSTM suffers from high computational cost and slow operation speed. GRU [13] simplifies the operation flow of logic gates on the basis of LSTM, but has limited efficiency improvement.

Transformer [14], as one of the State-Of-The-Art (SOTA) neural network structures in NLP tasks, has a unique attention mechanism that enables it to have the ability to remember contextual information farther than RNNs. Position Encoding enables Transformer to parallelize single-shot inference on the complete input sequence without significantly increasing the computational overhead like RNN structures. Tuli et al. proposed TranAD [15], a Transformer-based anomaly detection model for multivariate time series data. The idea of this work is highly pioneering and has excellent anomaly detection performance with rapid inference time.

In this paper, we proposes BTAD, a compound structure of Bi-Transformer for anomaly detection task of multivariate time series data, introducing adaptive multi-head attention mechanism to improve detection performance and enhance generalization ability. BTAD's main contributions are as follows:

i. We propose the Bi-Transformer architecture, which can carry out feature extraction and operation from two dimensions in parallel, which improves the performance and efficiency of multivariate time series data anomaly detection tasks;
ii. We construct an adaptive multi-head attention mechanism that enables BTAD to effectively capture the features of each dimension in multivariate time series data;
iii. We improve numerous auxiliary methods, such as alternating update strategy in the generative adversarial training approach, dataset division method in Model-agnostic meta learning (MAML) and modified Decoder structure, which enable BTAD to become a universal anomaly detection model for multivariate time series data. We further propose an evaluation metric that can consider both model performance and time efficiency.

The rest of the paper is organized as follows: Section 2 summarizes the related work. Section 3 describes the algorithm principle and operation process of BTAD in detail. Section 4 evaluates the performance and efficiency of BTAD through a variety of comparative experiments. Section 5 performs ablation analysis and sensitivity analysis of BTAD and discusses the limitations of proposed method. Section 6 summarizes the article.

2. Related work

Anomaly detection of serialized data is a long-term problem in scientific research. According to different data types, serialized data can be divided into two types: unary and multivariate. For the former, relevant methods mainly focus on series analysis and tracking of unary data to detect the existing anomalies [16]. For the latter, correlation methods mainly use multiple series to track each variable [17,18].

2.1. Methods based on statistics

Animesh [19] et al. discussed and summarized the methods of modeling time series data using PCA, process regression and hidden Markov chains, and pointed out the limitations and shortcomings of each method. As the research progressed, some methods evolved from statistics emerged. Paul [20] proposed GraphAn, a graph representation of low-dimensional embedding for detecting anomalous subsequences. This method converts the input of time series into graphs, and uses graph distance measurement to detect outliers. Some studies [21,22] adopt isolation forest, which uses a collection of multiple isolation trees to recursively divide the feature space for anomaly detection. Asrul [23] et al. proposed ARIMA, an Auto-Regressive Integrated Moving Average method to model and detect anomalous behaviors, which is also considered as one of the most representative models in statistical methods.

2.2. Methods based on machine learning

Osman [24] et al. combined SVM with linear regression to perform anomaly detection for wireless sensor networks in medical domain. Yiyang [25] et al. used a pre-trained single-class SVM and an adaptive extended Kalman filter to detect anomalies and improve the security of CAV system. SAND [26], CPOD [27] and Elle [28] all used clustering and database read/write history to detect outliers. Wenli [29] et al. integrated mean clustering with SVM to effectively improve model training efficiency and anomaly detection accuracy. Dhiman [30] et al. used adaptive threshold and twin support vector machine (TWSVM) for anomaly detection of two univariate time series data (gearbox oil and bearing temperatures) in wind turbines.

2.3. Methods based on neural network and deep learning

Currently, models based on deep learning and neural networks (DAD) mainly use reconstruction, prediction, generative, confidence score analysis etc. to detect anomalies. DAD learn hierarchical discriminative features from data, which eliminates the need of developing manual features by domain experts.

Kyle [31] et al. proposed LSTM-NDT, which takes the input sequence as training data and predicts the data of the next timestep on the current timestep. Guangxuan [32] et al. integrated LSTM and Generative Adversarial Networks (GAN), and proposed a model named LSTM-GAN to detect time series anomalies. Hong-soon [33] et al. proposed a scheme to detect anomaly data using LSTM autoencoder. However, all LSTM methods suffer from long time series data memory loss and inefficient modeling, especially when the dataset is noisy.

Bo [34] et al. proposed DAGMM, which reduces the dimension in the feature space by using a deep autoencoder Gaussian mixture model, and applies RNN to model time sequences. The parameters of each Gaussian are determined by the parameters of the DNN model. The autoencoder compresses the original input sequence into a latent space, and uses a recurrent estimation network to predict the next data point in the latent space. Ya [35] et al. proposed OmniAboration, a stochastic RNN for multivariate time series data anomaly detection. Compared with pure LSTMs, such methods have better anomaly detection performance, but still suffer from long training time problem.

Chuxu [36] et al. proposed MSCRED, which innovatively converts input sequences into 2-dimensional images using a CNN, and subsequently feeds them into a Conv-LSTM neural network to capture time patterns. Finally, the input feature matrix is reconstructed using a convolutional decoder by encoding inter-sensor correlations and feature mapping of temporal information, and the remaining feature matrix is further used to detect and diagnose anomalies. Li [37] et al. innovatively used an LSTM-GAN model, and modeled the time series distribution with generators. Also, this work uses discriminator losses to compute anomaly score. This work is highly inspiring and many GAN-based anomaly detection models for time series data have emerged on this basis.

Recent works have shown a trend of method mixing to compensate for the shortcomings of individual models. Hang [18] et al. proposed MTAD_GAT, a method for modeling feature and time correlation using graph attention network. Yuxin [38] et al. proposed CAEM, a method using convolutional autoencoder. It transmits time series through CNN, and further processes the output of CNN through Bi-LSTM neural network to capture the long-term time trend. Julien [39] et al. proposed USAD, an unsupervised anomaly detection method that enables fast and stable detection of multivariate data. The structure of USAD adopts an autoencoder with dual decoders and an adversarial training framework, which can effectively reduce the performance overhead associated with training. OpenGauss [40] is improved on the basis of LSTM structure by using a tree-based LSTM, which reduces the performance consumption of the model in terms of memory and resource usage, and is able to capture temporal trends in the presence of noisy data. However, the small input window makes OpenGauss ineffective in capturing long-term dependencies. In the experiment part, BTAD will also compare with the above various SOTA methods based on different neural network architectures, such as MTAD_GAT, USAD, etc., in terms of performance and efficiency.

Recently, there are also many tasks using Transformer model for anomaly detection. TranAD [15] restacks and reconstructs Transformer's architecture for better anomaly detection performance. HitAnomaly [41] separates the log into log template sequence and parameter value sequence, and uses two encoders to encode log template sequence and parameter value sequence respectively, then treats Transformer as a classification model to complete the corresponding anomaly detection task. Although HitAnomaly adopts Transformer model, it is only applicable to natural language log data, and cannot use general continuous time series data as input, so its application range is limited. Also, HitAnomaly does not improve the structure of Vanilla Transformer to further improve the detection performance. This problem also exists in the work of Haixuan [42], Markus [43] and Sergio [44] et al. Hayato [45] et al. used the attention mechanism in Transformer model for unsupervised anomaly noise detection task. Chuankai [46] et al. proposed an unsupervised log anomaly detection method LSADNET based on local information extraction and global sparse Transformer model to learn global dependencies between logs. Some scholars also began to apply Transformer to anomaly detection tasks in different domains, which confirms the generality of Transformer model structure from the side. TransAnomaly [47] applies Transformer to visual anomaly detection in videos. GTA [48] uses Transformer to learn graph structure and performs multivariate time series anomaly detection in IoT scenario. DCT-GAN [49] can further improve the accuracy and generalization ability of the model by using the method based on Dilated Convolutional Transformer. Yixin [50] et al. proposed TADDY, a Transformer-based dynamic graph anomaly detection method. TADDY is able to accomplish the task of dynamic graph anomaly detection by constructing a comprehensive node encoding strategy in the absence of information encoding.

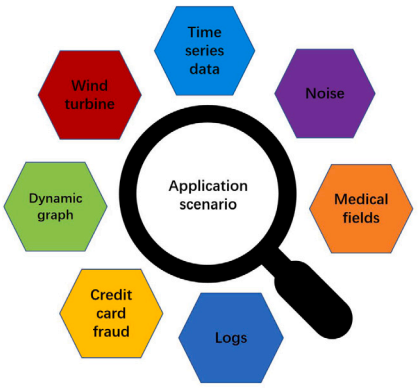

Fig. 1. Real-time application scenarios.

2.4. Hybrid method

A few methods also employ multiple strategies simultaneously to obtain better performance. Subama [51] et al. used a combination of SVM and Naive Bayes methods to build an anomaly detection system, and experimentally verified the superior performance of this hybrid method. Stratis [52] et al. simultaneously used three methods including wavelet analysis, Hilbert transform and neural network to detect anomalies in time series data.

2.5. Real-time application scenarios of anomaly detection

Serialized anomaly detection has the characteristic of multiple scenarios, such as wind turbine anomaly detection [30], log anomaly detection [41], noise anomaly detection [45], dynamic graph anomaly detection [50], credit card fraud anomaly detection [53,54], etc. In fact, time series data is a branch of serialized data. Credit card transaction records, timesteps of system logs, and other such structured types of data that have logical sequential relationships can be processed using deep learning methods with contextual feature extraction capabilities. Fig. 1 shows our summary of the real-time application scenarios for serialized data.

However, the presentation of sequential relationships varies in different application scenarios. For example, in credit card fraud, corresponding information on temporal and spatial behaviors needs to be integrated [53], whereas in wind turbines, only univariate time series

need to be considered [30]. Researchers need to modify, adapt the model structure to specific application scenarios, and perform transfer learning to achieve better performance.

2.6. Summary of related work

Let us make a brief summary: statistical methods and machine learning algorithms are not very effective in modeling time series data. LSTM and GRU neural networks, although capable of capturing contextual information of time series data, suffer from slow inference speed and inefficient operation. Research on Transformer's application in anomaly detection is still in its infancy, and most of the related research is focused on textualized anomaly detection tasks such as logs. Hybrid models are the future trend of related research in this area.

3. BTAD algorithm design

3.1. Preprocess

To promote stable training and enhance the robustness of BTAD, we convert datasets from different sources into a unified format, as shown in the following steps:

A. Identify and distinguish whether the dataset is a unary or a multivariate time series dataset
Identifying and distinguishing dataset types is one of the strategies to optimize the efficiency of BTAD. BTAD can automatically shield one of the Transformer structures to complete the anomaly detection task for unary time series data. In the rest of the cases, BTAD will activate Bi-Transformer simultaneously to improve detection performance. Another benefit of shielding is the ability to further reduce the performance overhead required for BTAD training and testing.

B. Removing irrelevant information from datasets & unifying formats and specifications
In this step, we remove some information irrelevant to anomaly detection, such as the source of the dataset, the description of the dataset, etc, and only reserve core information, such as dataset size, anomaly labels, time step, etc.

C. Matrix transpose and reshape operation (for multivariate time series datasets)
A multivariate time series dataset can be abstractly described as the matrix shown in Fig. 2.
Among them, we assume that the input multivariate time series dataset is a $m \times n$ size matrix. $U_i(1 \le i \le n)$ of dimension 0 represents a unit, and $T_j(1 \le j \le m)$ of dimension 1 represents a time step in the multivariate time series.
Both dimension 0 and dimension 1 require Transformer to extract anomaly features in the form of sliding windows for subsequent operations. Therefore, for dimension 1, we perform a reshape operation to convert it to the same matrix shape as dimension 0 for Transformer processing. The specific flow is shown in Fig. 3.

D. Normalization operation and sliding window sequence conversion
For multivariate time series datasets, the normalization operation needs to be carried out once from dimension 0 and dimension 1 respectively, and the running processes are identical in both directions. For unary time series datasets, the normalization operation only needs to carried out from dimension 0. Therefore, we only take dimension 0 as an example to illustrate here:
From the perspective of dimension 0, the multivariate time series dataset matrix can be regarded as

$$S = \{T_1, T_2, \dots, T_m\} \tag{1}$$

where $T_j(1 \le j \le m)$ represents a row in the matrix. Each element in the multivariate time series matrix can be indexed through the coordinate form such as $U_pT_q(1 \le p \le n, 1 \le q \le m)$.
We define the normalization formula as follows:

$$Normalize_s = \frac{T_j - \min(S)}{\max(S) - \min(S) + \delta} \tag{2}$$

where, δ is a small constant vector, which is designed to prevent the denominator from being 0 in some extreme cases of certain datasets. $\min(S)$ and $\max(S)$ respectively represent the mode wise minimum and mode wise maximum vectors under the current dataset.
Through the normalization operation, we compress the data range of the dataset into the space $[0, 1)$, then the multivariate time series dataset matrix is converted into:

$$S_{afternor} = \{T_{afternor1}, T_{afternor2}, \dots, T_{afternorm}\} \tag{3}$$

To further strengthen the contextual feature extraction ability of BTAD, we define an sliding input window $W_{T_{afternorj}}$ at time step $T_j(1 \le j \le m)$ of length $2k$:

$$W_{T_{afternorj}} = \{T_{afternorj-k}, T_{afternorj-k+1}, \dots, T_{afternorj+k}\} \tag{4}$$

In the case of $afternorj < k$ or $afternorj + k > m$, a padding operation is used to maintain a fixed window size. In time series tasks, anomalies at different time steps are not independent of each other, but are influenced by contextual associations (i.e., previous or subsequent time steps). Therefore, the sliding window allows BTAD to no longer be limited to analyzing anomalies for individual time steps, but to analyze serialized anomalies using the window size as the perceptual field range of the model, thus helps to attenuate fluctuations in the anomaly scores, avoid the problem of training instability triggered by excessive deviation from anomalies, similar to low-pass filter technology, which is also a common practice in previous works [39].
Now, the complete multivariate time series matrix can be re-expressed as:

$$W = \{W_{T_{afternor1}}, W_{T_{afternor2}}, \dots, W_{T_{afternorm}}\} \tag{5}$$

We define the current processing time step of Transformer as t. Then $S^t_{afternor}$ and W^t denote the time slice until the time step t of the normalized complete sequence and the sliding input window of length $2k$ for time step t respectively.
In order to use the generative adversarial training idea, we take both $S^t_{afternor}$ and W^t as the input of BTAD, which will be explained later in Section 3.2.2.

E. Divide train set, test set, add labels to test set, and store them in *.npy* format
The last stage of preprocessing is to divide the dataset into train set and test set. Based on the experience of previous work [55], we divide 80% of the dataset into train set and 20% into test set, label test set with $[0, 1]$ to measure the performance of different models in test stage. For a unary dataset, the final preprocessing format can be visually represented as:

$$Dataset = [\text{'}train\text{'}, \text{'}test\text{'}, \text{'}labels\text{'}] \tag{6}$$

For a multivariate dataset, we need to store both the original and the reshaped data, so the final preprocessing format can be expressed as:

$$Dataset_1 = [\text{'}train\text{'}, \text{'}test\text{'}, \text{'}labels\text{'}] \tag{7}$$

$$Dataset_2 = [\text{'}reshaped\_train\text{'}, \text{'}reshaped\_test\text{'}, \text{'}reshaped\_labels\text{'}] \tag{8}$$

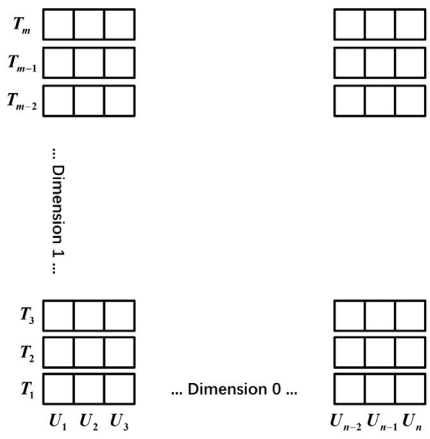

Fig. 2. The matrixed form of multivariate time series dataset.

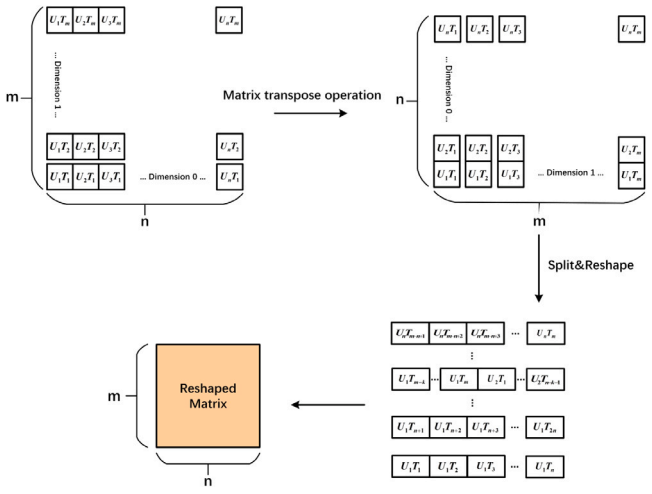

Fig. 3. Matrix transpose & reshape.

$$Dataset = Dataset_1 \cup Dataset_2 \tag{9}$$

Finally, all datasets are stored in *.npy* format for subsequent processing.

3.2. BTAD model

3.2.1. The overall structure of BTAD model

The overall structure of BTAD is shown in Fig. 4.

According to Fig. 4, for the multivariate time series data matrix, Transformer1 and Transformer2 in BTAD perform neural network operation and forward propagation from two dimensions. For Transformer2, the matrix dimension of Reshaped matrix is consistent with multivariate time series data matrix. The inputs of Transformer1, Transformer2 are both matrix $S^t_{afternor} \in R^{2k\times n}$ and matrix $W^t \in R^{t\times n}$. Both Transformer1 and Transformer2 generate two reconstructed output matrices ($RO_1, RO_2 \in R^{2k\times n}$) each, and obtain their respective anomaly score As_1 and As_2 by the relevant calculation methods. As_2 is further converted into the time step format. The Composite Anomaly Score (As) is obtained by weighted average calculation and judge whether there is an anomaly (see Section 3.3). For unary time series data, the multivariate time series data matrix degenerates into a vector. According to Section 3.1, BTAD will shield Transformer2, only call Transformer1 for operation, and use As_1 to determine whether there is an anomaly.

3.2.2. Detail design of transformer in BTAD model

Transformer has demonstrated superior performance in many NLP and CV tasks, but most of the existing studies have only applied Transformer to the log templates anomaly detection as described in Section 2.3. In order to meet the needs of multivariate time series anomaly detection tasks, we modify and restack the structure of Transformer, and incorporate various auxiliary methods.

Taking Transformer1 as an example, the internal structure of Transformer in BTAD is shown in Fig. 5 (corresponding to the blue part in Fig. 4).

Among them, the internal structures of Encoder1, Encoder2, Encoder3 and Encoder4 in Fig. 5 are basically the same. The slight difference is that Encoder3 and Encoder4 use a mask method in the adaptive multi-head attention mechanism to prevent local information from interfering during the training of Transformer, and can obtain further timesteps in the same input batches, as shown in Fig. 6.

In Fig. 6, the dotted arrows are residual connections of information. The role of introducing residual connections is to solve the problem of gradient disappearance and the degeneracy of weight matrix.

The process of Fig. 6 can also be described by the following formula:

$$EO_1 = LN(EI_1 + MHA(EI_1)) \tag{10}$$

$$EO_2 = LN(EO_1 + FFNN(EO_1)) \tag{11}$$

where, EI_1 is the input of a single encoder and EO_2 is the output of a single encoder. LN represents layer normalization operation, + represents matrix addition operation, $FFNN$ represents Feed-Forward Neural Network, and MHA is the adaptive multi-head attention mechanism mentioned later. The above operations use $S^t_{afternor}$ and W^t to generate attention weights for capturing temporal trends. For Transformer1, each current timestep does not depend on the output of the previous timestep, so BTAD is able to perform inference operations on multiple sliding windows in parallel, thus reducing training time.

The internal structures of Decoder1, Decoder2, Decoder3 and Decoder4 in Fig. 5 are identical, where Decoder1, Decoder2 and Decoder3, Decoder4 are connected in serial mode respectively, for the purpose of generative adversarial training using the corresponding reconstructed outputs. The specific structure of decoder is shown in Fig. 7.

Instead of using the Encoder–Decoder attention layer, decoder uses self-attention layer. This is because BTAD needs to judge the anomaly through the reconstruction error, while the Encoder–Decoder attention layer will focus on the input sequence, thus causing interference to the reconstruction output (see Section 5.1).

The process in Fig. 7 can also be described by the following formula:

$$DO_1 = LN(DI_1 + MHA(DI_1)) \tag{12}$$

$$DO_2 = LN(DO_1 + MHA(DO_1)) \tag{13}$$

$$DO_3 = LN(DO_2 + FFNN(DO_2)) \tag{14}$$

$$DO_4 = LeakyRelu(DO_3) \tag{15}$$

All symbols in the formula have the same meaning as the encoder symbols explained above. DI_1 is the input of a decoder and DO_4 is the final output of a decoder. BTAD uses $LeakyRelu$ activation function in a single decoder, and $Sigmoid$ activation function is used in the final output part of the stacked model. The reason is that $Sigmoid$ activation function can effectively compress the output into a $[0, 1]$ range, thus matching the data range of the normalized input and promoting the operation of BTAD. In order to facilitate the following description, some symbolic specifications for the input and output of the model in Fig. 5 are given here, which will not be explained later.

i. The input $S^t_{afternor}$ is called MI_1 (Model Input 1)
ii. The input W^t is called MI_2 (Model Input 2)

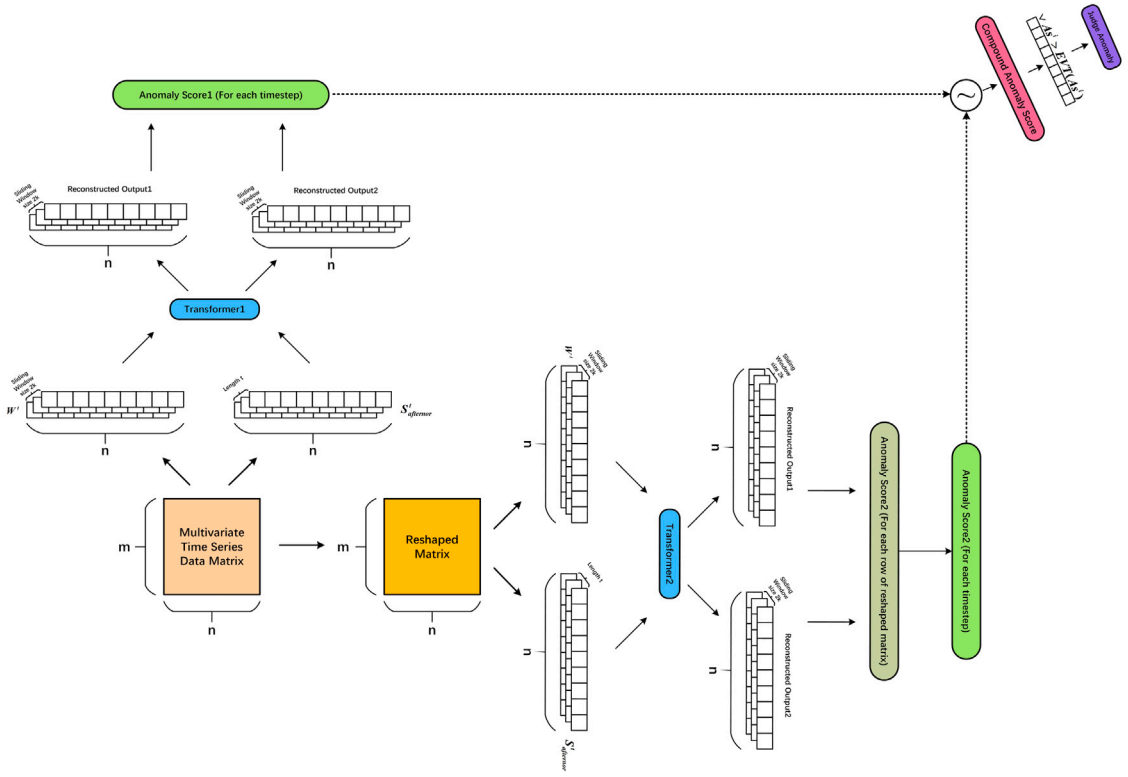

Fig. 4. The overall structure of BTAD model.

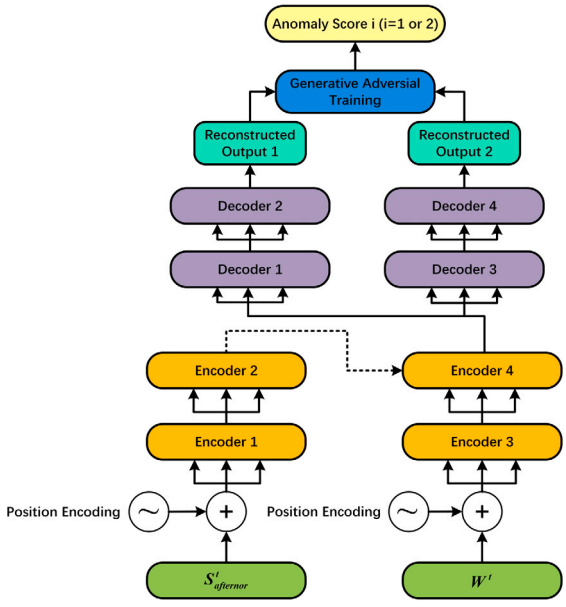

Fig. 5. The internal structure of BTAD model.

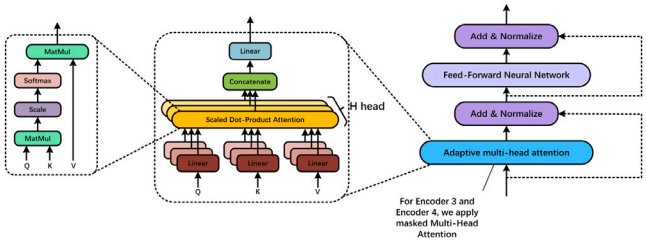

Fig. 6. The Encoder structure of BTAD.

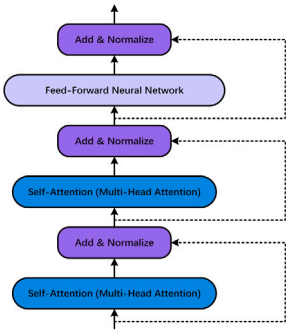

Fig. 7. The Decoder structure of BTAD.

iii. The Reconstructed Output1 of Decoder2 in the first stage is called MO_1S_1 (Model Output1 at Stage 1)
iv. The Reconstructed Output2 of Decoder4 in the first stage is called MO_2S_1 (Model Output2 at Stage 1)
v. The Reconstructed Output2 of Decoder4 in the second stage is called MO_2S_2 (Model Output2 at Stage 2)
vi. Encoder1, Encoder2, Encoder3, Encoder4 and Decoder1, Decoder2, Decoder3, Decoder4 are abbreviated as E_1, E_2, E_3, E_4 and D_1, D_2, D_3, D_4 respectively.

Note: Stage 1 and Stage 2 refers to Section 3.2.2.3 Generative Adversarial Training method.

3.2.2.1. *Position encoding.*

For the multivariate time series data anomaly detection task, we still need to determine the position information of each time step or each meta-dimension. BTAD model adopts an absolute Position Encoding method based on sin and cos function, which prevents the problem of

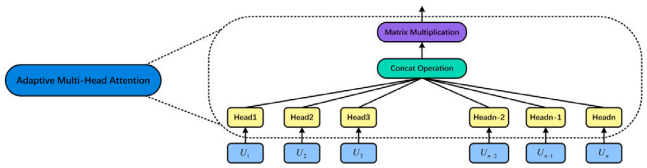

Fig. 8. The adaptive multi-head attention mechanism of BTAD.

encoding collisions in dimensions up to 20,000.

$$\overrightarrow{p_t} = f(t)^{(i)} := \begin{cases} \sin(\omega_k \cdot t) & \text{if(i=2k)} \\ \cos(\omega_k \cdot t) & \text{if(i=2k+1)} \end{cases} \tag{16}$$

where $\omega_k = \frac{1}{10000^{2k/d}}$.

After calculating the position information, it is necessary to add the Position Encoding vector to the model input vector, as shown in the following formula:

$$\Psi'(\omega_t) = \Psi(\omega_t) + \overrightarrow{p_t} \tag{17}$$

To ensure correct vector addition, the dimension of the Position Encoding vector $\overrightarrow{p_t}$ must be consistent with the input dimension.

3.2.2.2. Adaptive multi-head attention mechanism.

We define the Scale-dot product attention score formula as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{scale}})V \tag{18}$$

Q, K and V are three learnable parameters in Transformer, which are $Query$ matrix, Key matrix and $Value$ matrix respectively. They are essentially composed of $Query$ vectors, Key vectors and $Value$ vectors, and the specific weight values will be dynamically adjusted during model training. The essential reason for adopting matrix operation is to improve the operation efficiency of Transformer. The $softmax$ distribution is used to generate the convex combination weights for the matrix V and allows us to compress the calculation result into a smaller representative embedding space, which simplifies the subsequent neural network model inference operations.

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{19}$$

The operation of $\sqrt{scale}$ is to make the gradient of the model more stable, prevent the obvious fluctuations of weight, and promote stable training.

The above operation of attention mechanism can be regarded as a complete operation process of single attention head. In the BTAD model, we propose an adaptive multi-head attention mechanism whose number of attention heads is automatically adjusted according to the dimension of the current input to the model. Taking Transformer1 in Fig. 4 as an example, the number of attention heads is aligned with Dimension 0 of the multivariate time series data matrix. The impact of each element in the current time step on the system anomaly is measured by assigning a separate attention head to each element of the multivariate data and calculating an attention score using the attention mechanism mentioned above. The adaptive multi-head attention mechanism allows BTAD to achieve better performance than Vanilla Transformer because it can capture and pass numerical information of all elements without being limited to a fixed number of attention heads, which further extends the ability of the model on focusing at different positions, and enhance the attention layer by giving attention many sub representations. Fig. 8 visually illustrates BTAD's adaptive multi-head attention mechanism.

After completing the operation of each attention head with the adaptive multi-head attention mechanism, we apply $Concat$ operation to connect the operation results (Z matrix) of all attention heads. In order to avoid the problem of matrix dimension explosion, an additional weight matrix W^O is used to calculate with the matrix [56], and the final result matrix is re-compressed to the same dimension size as the matrix output by a single attention head. In the whole BTAD model, Q, K and V matrices mentioned above participate in joint training to adjust weights. The whole process can be described by the following formulas:

$$MHA(Q, K, V) = Concat(Head_1, Head_2, \ldots, Head_n) \tag{20}$$

where $Head_i = Attention(Q_i, K_i, V_i)(1 \le i \le n)$.

$$MatrixMult = MHA(Q, K, V) \bigotimes W^O \tag{21}$$

where $\bigotimes$ stands for matrix multiplication.

3.2.2.3. Generative adversarial training method.

Since GAN [37] has been proved to have excellent performance in the task of anomaly detection, and the 2-stage training methods proposed in TranAD [15] could further amplify the anomalous features and reduce false positive rates, BTAD adopts and modifies the 2-stage generative adversarial training method to train the model (the blue module in Fig. 5), and the pseudocode of the algorithm is as follows:

Algorithm 1 Two Stages Adversarial Training Algorithm

```
n=0
while n < N do
    for i=1 to T do
        MO_1S_1 = D_2(E_4(Dataset('train')))
        MO_2S_1 = D_4(E_4(Dataset('train')))
        MO_2S_2 = D_4(E_4(Dataset('train'), ||MO_1S_1
        −Dataset('train')||_2))
        Loss_1 = δ^{−n}||MO_1S_1 − Dataset('train')||_2 + (1−
        δ^{−n})||MO_2S_2 − Dataset('train')||_2
        Loss_2 = δ^{−n}||MO_2S_1 − Dataset('train')||_2 − (1−
        δ^{−n})||MO_2S_2 − Dataset('train')||_2
        Adjusting BTAD weights using an alternating
         update strategy
    n = n + 1
```

However, due to the difference in the model stacking structure, we involve the reconstructed outputs generated by D_2 and D_4 in the generative adversarial training stage.

Therefore, the final loss functions of BTAD in the 2-stage training are:

$$Loss_1 = \delta^{-n}\|MO_1S_1 - Dataset(\text{`}train\text{'})\|_2 + (1-\delta^{-n})\|MO_2S_2 - Dataset(\text{`}train\text{'})\|_2 \tag{22}$$

$$Loss_2 = \delta^{-n}\|MO_2S_1 - Dataset(\text{`}train\text{'})\|_2 - (1-\delta^{-n})\|MO_2S_2 - Dataset(\text{`}train\text{'})\|_2 \tag{23}$$

where δ is the evolutionary parameter, $Loss_1$ is the cumulative loss of D_2 and $Loss_2$ is the cumulative loss of D_4.

Drawing on the remarkable training success of GAN networks in the field of image generation and image repairing, we propose a strategy to alternately update the network weights during back propagation computation. Specifically, during each training epoch, $Loss_1$ is first propagated with the BP algorithm from D_2 to D_1, then to E_4 and the associated encoder modules. During this process, the internal weights of D_3 and D_4 are frozen. After the propagation process of $Loss_1$ is calculated, then $Loss_2$ is propagated from D_4 to D_3, then to E_4 and the

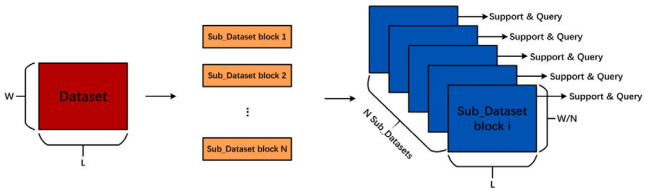

Fig. 9. Using MAML methods to divide the dataset.

associated encoder modules. Similarly, the internal weights of D_1 and D_2 are frozen. BTAD completes one epoch of weight updates during model training if and only if the propagation process of both $Loss_1$ and $Loss_2$ are computed. The alternating update strategy can further enhance the adversarial learning effect based on the amplified bias of the 2-stage training process (i.e., each training epoch extends the update step of D_1, D_2, D_3 and D_4), allowing BTAD to be fully trained and achieve remarkable performance even with a small number of training epochs (see Section 5.3.2).

3.2.2.4. Model-agnostic meta learning.

We introduce MAML technology [57] because MAML can not only enhance the universality of the model, but also enable the model to quickly learn the characteristics of new anomaly detection categories and carry out efficient anomaly detection with only a small amount of training data when facing new anomaly detection datasets. MAML is essentially an idea rather than a specific algorithm, whose purpose is to allow a neural network model to acquire as many features as possible with limited data. BTAD uses gradient update methods in MAML at each epoch of training to update the individual weight matrices in the neural network [57].

In order to attenuate the instability of generative adversarial training, guarantee the complete randomness or non-repeatability of the data for each training epoch, enhance the few-shot learning ability, accelerate the convergence speed of BTAD and improve the detection accuracy under multiple datasets, we propose our own division methods. Concretely, we further divide each complete training set of multivariate time series data into N $Sub\_Datasets$, each of which also contains its own $supportset$, $queryset$ and $labels$, which can be expressed as:

$$Sub\_Dataset = [\text{'}sub\_support\text{'}, \text{'}sub\_query\text{'}, \text{'}sub\_labels\text{'}] \tag{24}$$

N is not a fixed value, but is set to different values depending on the specific dataset. Since BTAD adopts a sliding window form of input, we cannot simply use random sampling to collect data and generate $Sub\_Datasets$, as this would result in the loss of time series dependencies. BTAD divides the complete training set into blocks, each of which is a continuous multivariate time series data, and blocks are not intersected by each other, as shown in Fig. 9.

Suppose $Dataset \in R^{L\times W}$, then $Sub\_Dataset \in R^{L\times W/N}$ for each $Sub\_Dataset$ block.

3.2.2.5. Self-conditioning mechanism.

The self-conditioning mechanism of BTAD has been clearly reflected in Section 3.2.2.3 where the alternating update strategy optimize the model weight, and in Section 3.2.2.4 the learning step of MAML updates the weight matrix in neural network after each training epoch. In addition, the learning rate in BTAD is dynamically adjusted by $StepLR$, and the $AdamW$ optimizer is adopted. These methods allow BTAD to automatically optimize the parameters in the neural network model according to the results of existing training data. Therefore, we collectively refer to these methods as BTAD's self-conditioning mechanism.

3.3. Anomaly score

In Section 3.2, we discuss the complete running process and implementation details of single Transformer model in Fig. 4. We now use the trained BTAD model for testing. The pseudo code of the whole process is shown in Algorithm 2:

Algorithm 2 BTAD Testing Algorithm

1: **for** i=1 to 2 **do**
2: **for** t=1 to T **do**
3: $MO_1S_1 = D_2(E_4(Dataset(\text{'}test\text{'})))$
4: $MO_2S_1 = D_4(E_4(Dataset(\text{'}test\text{'})))$
5: $MO_2S_2 = D_2(E_4(Dataset(\text{'}test\text{'}), ||MO_1S_1 - Dataset(\text{'}train\text{'})||_2)), D_4(E_4(Dataset(\text{'}test\text{'}), ||MO_1S_1 - Dataset(\text{'}train\text{'})||_2))$
6: $As_i = \rho \times ||MO_1S_1 - Dataset(\text{'}test\text{'})||_2 + (1-\rho) \times ||MO_2S_2 - Dataset(\text{'}test\text{'})||_2$
7: $As = WeightedMean(As_1, As_2)$
8: $JA = 1(if(As \geq EVT(As)))$

Among them, the variable $i(i = 1or2)$ in Algorithm 2 is used to reveal the anomaly score As_i calculated by the i^{th} Transformer. As denotes Anomaly Score. It weights the As_1 and As_2 calculated by the two Transformers respectively according to a scale factor ϕ to obtain As. ϕ can be fine-tuned according to different datasets used for training and testing. For unary time series datasets, $As = As_1$. Then we use a dynamic threshold adjustment algorithm (EVT) to judge whether As will trigger system anomalies, and give the final diagnosis results of the whole BTAD model.

From Algorithm 2, the core formula for calculating As_i is

$$As_i = \rho \times \|MO_1S_1 - Dataset(\text{'}test\text{'})\|_2 + (1-\rho) \times \|MO_2S_2 - Dataset(\text{'}test\text{'})\|_2 \tag{25}$$

ρ is used to regulate the contribution of D_2 and D_4 to the anomaly score. Meanwhile, the sum of coefficients is 1, which is used to ensure that the total weight is constant.

In the test stage, BTAD still follows the 2-stage process during training, and uses the results of Stage 1 to provide reference for the weight adjustment of Stage 2. As_1 and As_2 differ in a theoretical sense. For Transformer1, it calculates the anomaly scores for different dimensions of each time step; For Transformer2, it calculates the anomaly scores for different time steps of each dimension. We weight average the anomaly scores calculated by two Transformer, so as to ensure that the As generated by BTAD can take into account both the anomalies in the time series dimension and the anomalies in the multivariate series dimension. As the anomaly thresholds vary across different datasets, we apply Extreme Value Theory (EVT) method to automatically and dynamically adjust the thresholds according to different test situations. This method is essentially a statistical method for fitting the data distribution with the Generalized Pareto Distribution (GPD) using the EVT (corresponding to line 8 of Algorithm 2). Fig. 10 shows this process with the MBA dataset as an example.

As can be seen in Fig. 10, the peaks of the anomaly scores in different dimensions are highly correlated with the noise in the corresponding dimensional data. Anomaly scores are higher for time steps (noise) that change significantly within the time series. In addition, the fluctuations of anomaly scores vary for different dimensions, indicating that the self-conditioning mechanism of BTAD is able to assign different weights to different dimensions of different datasets, thus enhancing the adaptability and robustness of BTAD to noisy data.

4. Experiment and evaluation

4.1. Experimental setup

We summarize the experimental environment as shown in Table 1.

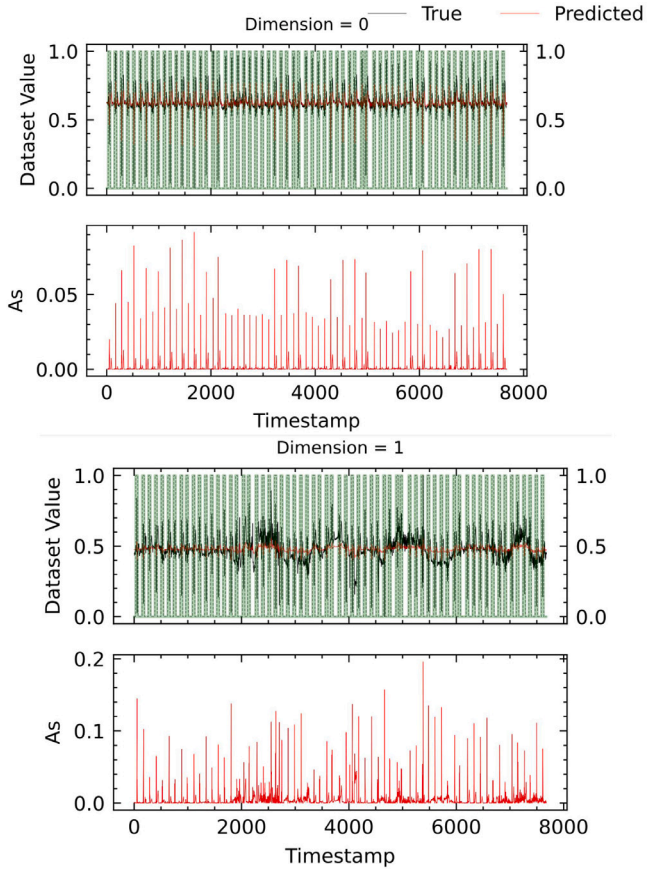

Fig. 10. Visualization flow of BTAD anomaly detection task.

Table 1
Experimental environment.

CPU	Intel Xeon E7-4820 V4 (40 CPUs)
RAM	32GB
GPU	NVIDIA Quadro RTX-5000 ×2
Language	Python 3.8.3
AI Framework	PyTorch 1.8.1

4.2. Dataset sources

In the experimental part, we use 6 different datasets (including 5 publicly available datasets) for our experiments. The relevant information and introduction of each dataset are as follows:

i. SMD (server machine dataset) [35]: SMD is a dataset that monitors the resource utilization of 28 high-performance computers in the server computing cluster for 5 weeks. It is a typical multivariate time series dataset with 38 dimensions.
ii. SWaT (Secure Water Treatment Dataset) [58]: SWaT dataset is a classic anomaly detection dataset derived from sensor data of a real water treatment plant. The water treatment plant obtains normal and anomaly data by running normally for 7 days and abnormally for 4 days. In addition to the sensor data that can collect relevant information such as water level and flow rate, some actuator operations (such as pumps, valves, etc.) are also recorded in the data. Its data dimension is 1 and belongs to unary time series dataset.
iii. SMAP (Soil Moisture Active Passive Dataset) [31]: SMAP is a dataset of soil samples and telemetry information collected by NASA's Mars rover. It is a typical multivariate time series dataset with 25 dimensions.

Table 2
Dataset Information.

Dataset	Dimensions	Data volume
SMD	38	1416830
SWaT	1	946276
SMAP	25	562800
MBA	2	200005
NAB	1	8068
MSDS	10	292860

iv. MBA (MIT-BIH Supraventricular Arrhythmia Dataset) [59]: MBA dataset is a medical dataset which contains the electrocardiogram recordings of 4 patients. It contains two different types of anomalies (supraventricular contractions and premature heartbeats). Its data dimension is 2, which belongs to binary time series dataset.
v. NAB (Numenta Anomaly Benchmark) [60]: NAB is a comprehensive dataset including many real-world aspects, such as CPU utilization, temperature sensor readings, etc. Its data dimension is 1 and belongs to unary time series dataset.
vi. MSDS (Multi-Source Distributed System Dataset) [61]: MSDS is also a comprehensive dataset with multiple data sources, such as application logs, distributed system metrics, etc. This dataset is specially built for AI research. Its data dimension is 10 and belongs to multivariate time series dataset.

Table 2 simplifies the information of relevant datasets.

4.3. Comparative experiment

In the comparative experiment part, we compare BTAD with several methods based on different types of neural network architectures. We do not include statistical methods and machine learning algorithms in the comparison range as the performance of them is much lower than that of neural network models. We respectively select TranAD [15] as the representative method of Transformer architecture; LSTM_NDT [31] as the representative method of LSTM neural network; MAD_GAN [37] as the representative method of GAN neural network; MTAD_GAT [18] as the representative method of Graph Neural Network (GNN); GDN [17] as the representative method of graph attention network; MSCRED [36] as the representative method of CNN; USAD [39] as the representative method based on the Encoder–Decoder architecture. The above scientific research achievements basically cover the mainstream neural network model methods applied to anomaly detection tasks.

We select $Precision$, $Recall$, $F1$ which are commonly used in the field of neural networks and deep learning, as well as the commonly used evaluation index (AUC) in the binary classification task to evaluate the detection performance of the above models and BTAD.

The calculation formulas of $Precision$, $Recall$ and $F1$ evaluation indexes are as follows:

$$Precision = \frac{TP}{TP + FP} \tag{26}$$

$$Recall = \frac{TP}{TP + FN} \tag{27}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{28}$$

At the same time, we examine the training efficiency of each model. The evaluation from the aspects of performance and efficiency allows us to more comprehensively examine the overall performance of a method. We also manually construct datasets with only 20%, 40%, 60% and 80% of Table 2 to simulate the performance of each model with insufficient training data. All experiment results are obtained by repeating 10 times under the same test environment and taking the average

Table 3
Performance comparison between BTAD and other models on 6 different datasets.

Dataset	SMD				SWaT			
Model	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1
BTAD	0.9940	0.9974	0.9984	0.9957	0.9977	0.6879	0.8438	0.8143
TranAD	0.8893	0.9974	0.9923	0.9403	0.9697	0.6957	0.8462	0.8101
MAD_GAN	0.9561	0.8440	0.9720	0.8966	0.9593	0.6956	0.8456	0.8064
LSTM_NDT	0.9424	0.8428	0.9358	0.8898	0.7777	0.0108	0.5052	0.0214
MTAD_GAT	0.8258	0.9199	0.9354	0.8703	0.9760	0.6956	0.8465	0.8123
GDN	0.7169	0.9973	0.9783	0.8342	0.9696	0.6956	0.8462	0.8101
MSCRED	0.7253	0.9974	0.9596	0.8399	0.9999	0.6770	0.8385	0.8074
USAD	0.8759	0.9973	0.9633	0.9327	0.9677	0.6878	0.8438	0.8040
Dataset	SMAP				MBA			
Model	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1
BTAD	0.8274	0.9999	0.9899	0.9056	0.9548	0.9999	0.9879	0.9769
TranAD	0.8078	1.0000	0.9885	0.8937	0.9612	0.9698	0.9787	0.9655
MAD_GAN	0.8157	0.9213	0.9789	0.8653	0.9396	0.9589	0.9706	0.9492
LSTM_NDT	0.8322	0.7326	0.8601	0.7792	0.9117	0.9423	0.9589	0.9267
MTAD_GAT	0.7517	0.9999	0.9840	0.8582	0.9012	0.9264	0.9711	0.9136
GDN	0.7892	0.9589	0.9800	0.8658	0.8441	0.9367	0.9527	0.8880
MSCRED	0.8102	0.9054	0.9569	0.8552	0.9269	0.9458	0.9702	0.9363
USAD	0.7739	0.9999	0.9827	0.8725	0.8953	0.9587	0.9700	0.9259
Dataset	NAB				MSDS			
Model	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1
BTAD	0.8889	0.9999	0.9996	0.9412	1.0000	0.8007	0.9003	0.8893
TranAD	0.8797	0.9823	0.9540	0.9282	1.0000	0.8026	0.8925	0.8905
MAD_GAN	0.8652	0.7002	0.8462	0.7740	0.9999	0.6107	0.8053	0.7583
LSTM_NDT	0.6399	0.6666	0.8322	0.6530	0.9999	0.8005	0.8007	0.8892
MTAD_GAT	0.8408	0.7265	0.8208	0.7795	0.9999	0.7964	0.8982	0.8866
GDN	0.8088	0.7858	0.8538	0.7971	0.9989	0.7925	0.8958	0.8838
MSCRED	0.8522	0.6700	0.8402	0.7502	1.0000	0.7887	0.8826	0.8819
USAD	0.8421	0.6666	0.8329	0.7441	0.9999	0.7958	0.8979	0.8863

value to obtain statistical significance. The specific experimental results are shown in Tables 3–6.

As seen in Table 3, the average $F1$ and AUC value of BTAD for the 6 datasets are 0.9205 and 0.9533. For $F1$, BTAD's performance is only slightly behind TranAD (0.8905) in MSDS dataset. For AUC, BTAD's performance is only slightly behind TranAD (0.8462), MAD_GAN (0.8456), GDN (0.8462) and MTAD_GAT (0.8465) model in SWaT dataset, and is equal to USAD model (0.8438), while it is ahead of other models in the remaining datasets. It is worth noting that BTAD model has obvious performance advantages on datasets with high dimensions and large data volumes such as SMD. This is because the Bi-Transformer structure adopted by BTAD can effectively parallelize anomaly detection from different dimensions for large-scale multivariate time series datasets. In addition, the modified decoder structure helps BTAD to produce reconstructed output more accurately in the complex latent space of high-dimensional datasets without receiving inference from the input. The performance of all models on SWaT dataset is relatively weak due to its large scale in terms of sequence length. Even Transformer may forget some information dependencies between extreme long distances. Among all the comparison models, TranAD also shows good performance on different datasets, which is closely related to its attention mechanism and generative adversarial training approach.

MSCRED can effectively retain time information due to the use of continuous observations as input values. It has good performance on partial datasets, but suffers from difficulties in identifying anomalies close to normal tendencies and slow model inference speed. BTAD's Bi-Transformer architecture can effectively capture both local information and global dependencies from different dimensions simultaneously. The improved 2-stage training method further amplifies anomalous features. At the same time, the Transformer in BTAD can effectively track all inputs and capture long-term dependencies due to the introduction of Position Encoding and residual-connection methods.

Table 4 shows the training time required for all methods to achieve the performance in Table 3.

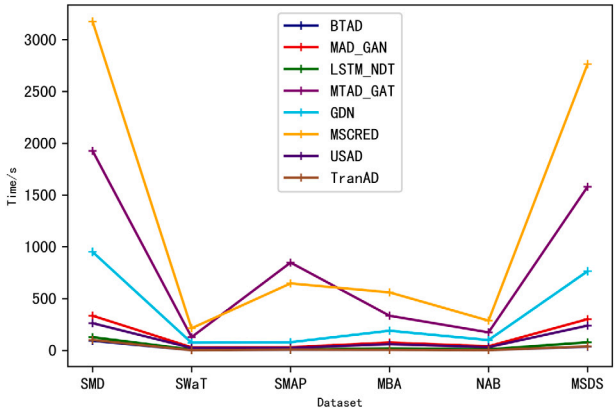

Fig. 11. Time efficiency comparison of 7 models.

As can be seen from Table 4, both BTAD and TranAD show significant efficiency advantages compared with other models because both use meta-learning strategies to accelerate model training. Among the 7 models, the continuous observation input of MSCRED model and the GRU structure of MTAD_GAT model make their operation speed quite inefficient. In large volume datasets such as SMD, their training time is more than 10 times slower than BTAD. Besides BTAD and TranAD, only USAD considers the problem of time performance optimization, but with limited effect. Therefore, although both USAD model and MAD_GAN model adopt the generative adversarial training method, the training time of USAD model is reduced compared with that of MAD_GAN model. After combining the performance indexes in Table 3 and the time efficiency in Table 4, BTAD model has the best comprehensive performance among all 7 models. Fig. 11 shows the results in Table 4 graphically, which better visualizes the efficiency advantage of BTAD and TranAD.

Table 4
Efficiency comparison between BTAD and other 7 models on different datasets.

Dataset / Model	SMD	SWaT	SMAP	MBA	NAB	MSDS
	Time (s)					
BTAD	92.8881	1.7756	6.6304	5.4831	2.4167	35.9454
TranAD	100.4392	1.8986	8.0124	5.6987	2.3832	38.8586
MAD_GAN	334.8236	29.5996	30.6314	75.6603	39.7251	301.4795
LSTM_NDT	127.5400	7.4594	7.7605	19.4458	10.0636	77.8719
MTAD_GAT	1926.8700	127.8575	848.0100	334.5484	173.8848	1579.9471
GDN	951.2648	73.4518	78.7033	189.7535	99.3113	764.3773
MSCRED	3175.9200	213.3390	646.7705	559.9000	287.5003	2764.5600
USAD	262.7442	23.5435	24.0086	60.0322	31.2616	239.0076

Table 5
The $PerformanceScore$ of 8 models.

Dataset / Model	SMD	SWaT	SMAP	MBA	NAB	MSDS
	PerformanceScore					
BTAD	197.8685	207.9957	262.3226	441.9611	737.9964	73.3844
TranAD	134.4916	184.2652	220.1675	381.9198	527.5715	69.0333
MAD_GAN	72.6339	33.9831	106.5911	132.7052	25.8363	11.8656
LSTM_NDT	67.2877	0.0669	52.0632	157.5698	18.3284	33.1848
MTAD_GAT	38.2781	24.7213	53.4498	80.0117	16.3660	34.6769
GDN	43.9761	27.4949	84.3570	68.0892	24.8723	37.2876
MSCRED	34.5948	20.6958	46.4859	83.7710	14.0524	28.5328
USAD	89.3379	35.5388	122.5906	121.4733	21.3247	46.4681

In order to better and more intuitively consider the performance of a model in terms of performance and training efficiency, we propose an evaluation index using numerical fitting method, called $PerformanceScore$, which can be calculated by the following formula:

$$PerformanceScore = \frac{e^{(AUC+F1-1)\times K}}{\log_{10} Time} \tag{29}$$

Among them, the numerator of $PerformanceScore$'s calculation formula adopts the exponential form in order to amplify the performance impact of the two evaluation indexes AUC and $F1$. The purpose of subtracting 1 is that the AUC and $F1$ values are actually meaningless if they are lower than 0.5, because even if pure probability prediction is used, the results of the corresponding AUC and $F1$ values are 0.5. The denominator takes a logarithm to the running time so as to weaken the impact of time on $PerformanceScore$. For example, although MTAD_GAT model runs tens of times longer on SWaT dataset than LSTM_NDT model, its performance is much better, so the MTAD_GAT model achieves a higher $PerformanceScore$. K in the numerator is an adjustable parameter, whose purpose is to balance the relationship between performance and efficiency, and can be set to different values according to different demand scenarios. For example, in a scenario with high performance requirements, the value of parameter K should be increased, while in a scenario with high efficiency requirements, the value of parameter K should be decreased. Here we set K to 6. $PerformanceScore$ can effectively consider the comprehensive capabilities of a model in terms of both performance and efficiency. For example, MTAD_GAT model and GDN model have very similar performance on SWaT dataset, but GDN model has a higher $PerformanceScore$ because the training time of GDN model is much shorter than MTAD_GAT model.

Combining data in Tables 3 and 4, the results of all models under $PerformanceScore$ evaluation index are shown in Table 5.

Table 5 also support the previous performance analysis conclusions of different models based on the data in Tables 3 and 4. BTAD is the performance leader among all models.

Table 6 shows the performance of each model with 20% training data volume. Here, we choose SWaT and SMAP dataset as examples to represent the performance of each model in large-scale datasets with sharply reduced data volumes.

As can be seen from Table 6, when the amount of data in the train set decreases sharply, 6 comparison models show different degrees of performance degradation. The performance of LSTM_NDT model on SMAP dataset decreases most significantly, AUC and $F1$ decreases by 18.54% and 30.48% respectively. BTAD model has almost no performance degradation on the SWaT dataset, and even has a slight improvement in AUC compared with the results of 100% training data volume, which can almost be considered as a data fluctuation. BTAD also performs well on SMAP dataset, with the performance degradation of less than 0.5% for both AUC and $F1$. The improved MAML dataset division strategy allows BTAD to quickly learn the anomalous features in sequence data with a small data volume, which shows its high performance even with limited data. With 20% of training data, the performance of USAD and TranAD model are the closest to BTAD. This is inseparable from the idea that both USAD and TranAD use the generative adversarial training approach to train the Encoder–Decoder structure, which further improve the learning ability, thus enabling them to perform well on a smaller amount of data. But in general, BTAD has the best comprehensive performance among all methods, both in the complete training dataset and in the training dataset with 20% data volume.

5. BTAD model analysis and discussion

5.1. Comparative analysis

We evaluate the BTAD model using self-attention at the decoder (refer to $Model1$) and the BTAD model using Encoder–Decoder attention at the decoder (refer to $Model2$) with SWaT and SMAP datasets at 20% and 100% data volumes respectively, the results are shown in Table 7.

In different test environments, $Model2$ shows different magnitudes of performance degradation compared to $Model1$, which also confirms the discussion in Section 3.2.2.

5.2. Ablation analysis

In the ablation analysis section, in order to investigate the importance of each component in BTAD model, we use the methods of removing the self-conditioning mechanism of BTAD, removing the generative adversarial training method of BTAD, and only using the basic Transformer model (i.e., removing the self-conditioning mechanism, removing the Bi-Transformer architecture, removing the generative

Table 6
The performance of each model at 20% training data.

Dataset	SWaT				SMAP			
Model	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1
BTAD	0.9676	0.6957	0.8461	0.8094	0.8202	0.9999	0.9894	0.9012
TranAD	0.9310	0.6946	0.8405	0.7956	0.7897	0.9726	0.9790	0.8717
MAD_GAN	0.9218	0.6944	0.8405	0.7921	0.7965	0.9011	0.9554	0.8456
LSTM_NDT	0.7214	0.0108	0.5014	0.0213	0.7917	0.4117	0.7006	0.5417
MTAD_GAT	0.9017	0.6756	0.8404	0.7724	0.7164	0.9999	0.9809	0.8348
GDN	0.9334	0.6656	0.8217	0.7771	0.7324	0.9577	0.9463	0.8300
MSCRED	0.9199	0.6456	0.8133	0.7587	0.7849	0.8988	0.9150	0.8380
USAD	0.9455	0.6823	0.8410	0.7926	0.7551	0.9999	0.9798	0.8604

Table 7
Comparative analysis results.

Dataset	SWaT (100% data volume)				SMAP (100% data volume)			
Model	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1
$Model1$	0.9977	0.6879	0.8438	0.8143	0.8274	0.9999	0.9899	0.9056
$Model2$	0.9798	0.6724	0.8327	0.7975	0.8034	0.9731	0.9855	0.8801
Dataset	SWaT (20% data volume)				SMAP (20% data volume)			
Model	Pre	Rec	AUC	F1	Pre	Rec	AUC	F1
$Model1$	0.9676	0.6957	0.8461	0.8094	0.8202	0.9999	0.9894	0.9012
$Model2$	0.9442	0.6701	0.8285	0.7839	0.7977	0.9691	0.9853	0.8751

Table 8
The comparison results of ablation analysis.

Dataset	SMD		NAB	
Model	AUC	F1	AUC	F1
BTAD	0.9984	0.9957	0.9996	0.9412
BTAD (without self-conditioning)	0.9910	0.9294	0.9780	0.9230
BTAD (without adversarial)	0.9932	0.9487	0.8330	0.7442
BTAD (original Transformer)	0.9761	0.8200	0.8329	0.7441

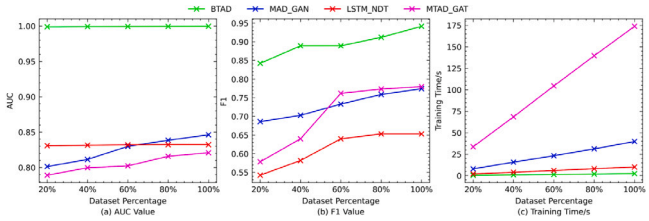

Fig. 12. AUC, $F1$ and training time of BTAD versus comparison models for different dataset sizes.

adversarial training method and using a fixed number of attention heads) for comparison experiments. We take SMD and NAB datasets as examples to illustrate with AUC and $F1$ evaluation indexes. Results are shown in Table 8.

In Table 8, compared with BTAD, the AUC and $F1$ of the basic Transformer structure on SMD dataset and NAB dataset decreases by 2.23%, 17.65%, 16.68% and 20.94% respectively. When we remove the self-conditioning mechanism and the generative adversarial training method, BTAD also shows different degrees of performance degradation. In the NAB dataset, the performance of BTAD model without generative adversarial training decreases significantly, and the performance is almost the same as the original Transformer model, indicating that both adversarial training and self-conditioning mechanism contribute to BTAD. More results are shown in Fig. 14.

5.3. Sensitivity analysis

5.3.1. Sensitivity to data volume

In Table 6, the experimental results of SWaT and SMAP show that BTAD model performance is almost insensitive to the dataset size and is capable of anomaly detection on small-scale datasets. In Fig. 12, we further show the results of $F1$, AUC evaluation indexes versus training time for BTAD and 3 comparison models at 20%, 40%, 60%, 80% and 100% data volumes, taking the NAB dataset as an example. It can be seen that BTAD has higher anomaly detection performance and better time efficiency at all dataset scales, which again supports the superiority of improved MAML methods in BTAD.

5.3.2. Sensitivity to training epochs

Taking SMD dataset as an example, we analyze the performance of BTAD model when the number of training epochs is 1–10. The specific results are shown in Table 9.

Table 9
Relationship between the number of training epochs and the performance of BTAD model under SMD dataset.

Epochs	Pre	Rec	AUC	F1
1	0.0000	0.0000	0.5	0.0000
2	0.9008	0.9973	0.9930	0.9466
3	0.9278	0.9974	0.9947	0.9613
4	0.9630	0.9974	0.9967	0.9799
5	0.9940	0.9974	0.9984	0.9957
6	0.9992	0.9974	0.9986	0.9983
7	0.9992	0.9974	0.9986	0.9983
8	0.9992	0.9974	0.9986	0.9983
9	0.9992	0.9974	0.9986	0.9983
10	0.9992	0.9974	0.9986	0.9983

As can be seen from Table 9, except for the extreme case where the BTAD model is unable to effectively adjust the model weights during the first training epoch, BTAD model can achieve high anomaly detection performance within a short number of training epochs, and the model performance has stabilized when the number of training epochs exceeds 6. This is due to the self-conditioning approach and modified 2-stage generative adversarial training method of BTAD, in which Stage 2 training process will amplify the training loss of Stage 1, so as to timely optimize the weight parameters of neural networks in the model. Alternating update strategy further accelerates the model weight adjustment. Table 9 also shows that the number of training epochs should be generally set at 5–6 epochs when applying BTAD model for anomaly detection tasks. Too few training epochs will not allow the model to be fully trained, and too many training epochs will only cause additional resource consumption and reduce the time

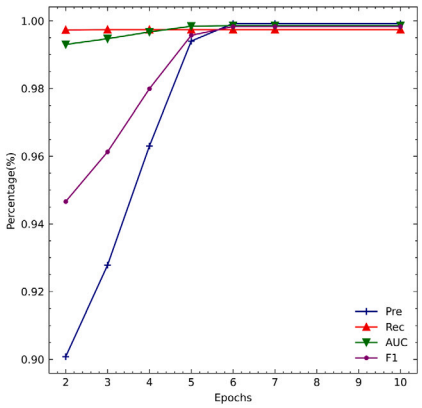

Fig. 13. The relationship between BTAD model performance and the number of training epochs.

Table 10
Relationship between window size and BTAD model performance under SMD dataset.

Window size	Pre	Rec	AUC	F1	Time (s)
5	0.9921	0.9974	0.9983	0.9948	48.5273
10	0.9940	0.9974	0.9984	0.9957	92.8881
13	0.9936	0.9974	0.9983	0.9955	131.9831
15	0.9907	0.9974	0.9982	0.9940	178.0146
17	0.9866	0.9974	0.9980	0.9920	195.0223

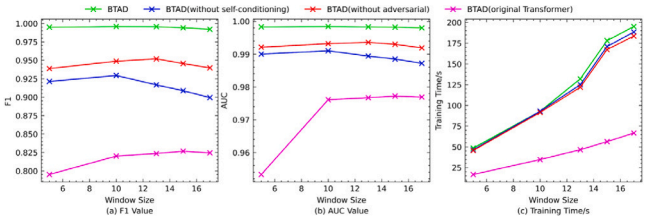

Fig. 14. $F1$, AUC and training time of BTAD and its variants at different window sizes.

efficiency of BTAD model. Therefore, we set the number of training epochs in the previous comparison experiments as 5. Fig. 13 illustrates this fact more visually in graphical form:

5.3.3. Sensitivity to sliding window size

Similarly to Table 9, we investigate the impact of sliding window size on BTAD model performance, and the results are shown in Table 10.

As seen in Table 10 that the sliding window size has an impact on both the performance and time efficiency of BTAD model. When the window is small, the model inference takes less time and the training efficiency is higher. However, too small sliding window will lose a large amount of contextual information, which will lead to performance degradation. If the window is too large, it will not only increase the computational effort of model inference, reduce the training efficiency, but also increase the usage of hardware resources. More critically, too large sliding window may cause some short sequence anomalies to be included in the complete window sequence, which cannot be effectively identified by the model and lead to performance degradation. Therefore, considering both time efficiency and performance, we finally choose a sliding window of size 10, which provides a reasonable balance between the two. We show the performance of BTAD and its ablation variant models with different window sizes in Fig. 14.

5.4. Limitations of proposed method

The above experiments and analysis illustrate that BTAD has efficient and highly accurate anomaly detection performance. However, BTAD also has application limitations. Transformer itself suffers from excessive performance overhead and system resource consumption, and the Bi-Transformer architecture of BTAD further amplifies this disadvantage. Therefore, the deployment of BTAD may be limited on edge computing platforms with scarce computing resources and weak computility. BTAD needs to complete training and save model files on high computing power platforms (i.e., servers, computing clusters) in advance before deploying to edge devices.

6. Conclusion

We propose a universal Bi-Transformer based anomaly detection model BTAD, which can detect anomalies in multivariate time series data. The Bi-Transformer structure of BTAD can parallelize the anomaly inference on the dataset from two different dimensions. The results of ablation analysis also show that the adaptive multi-head attention mechanism, the generative adversarial training approach and the self-conditioning mechanism effectively improve the anomaly detection performance of BTAD. Improved MAML strategies optimize the performance of BTAD on small scale datasets while improving the model training efficiency. Experiments show that BTAD model improves $F1$ by more than 5% and reduce the training time by more than 7% compared to other SOTA methods on large scale datasets such as SMD. Therefore, BTAD model can meet the requirements of rapid, accurate and unsupervised anomaly detection tasks in modern industrial systems, and has practical application value. We also put forward the $Performance Score$ evaluation index, which can comprehensively measure the overall capability of a model in terms of performance and efficiency.

In the future, we will look for ways to further improve the detection performance of BTAD and explore how to reduce the performance overhead of the Bi-Transformer structure, such as using Informer [62] model with low computational complexity.

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.

Data availability

Data will be made available on request.

Acknowledgments

The authors thank the anonymous reviewers for their insightful suggestions on this work.

References

[1] X. He, K. Zhao, X. Chu, AutoML: A survey of the state-of-the-art, Knowl.-Based Syst. 212 (2021) 106622, http://dx.doi.org/10.1016/j.knosys.2020.106622.
[2] S. Thudumu, P. Branch, J. Jin, J.J. Singh, A comprehensive survey of anomaly detection techniques for high dimensional big data, J. Big Data 7 (1) (2020) 1–30, http://dx.doi.org/10.1186/s40537-020-00320-x.
[3] J. Bellendorf, Z.Á. Mann, Classification of optimization problems in fog computing, Future Gener. Comput. Syst. 107 (2020) 158–176, http://dx.doi.org/10.1016/j.future.2020.01.036.
[4] J. Zhou, B. Zhang, L. Fan, Z. Lu, Aeromagnetic anomaly detection under low SNR conditions using multiscale wavelet energy accumulation, in: 2020 IEEE 20th International Conference on Communication Technology, ICCT, IEEE, 2020, pp. 1641–1644, http://dx.doi.org/10.1109/ICCT50939.2020.9295710.

[5] E.J. Son, W. Kim, Y.-M. Kim, J. McIver, J.J. Oh, S.H. Oh, Time series anomaly detection for gravitational-wave detectors based on the Hilbert–Huang transform, J. Korean Phys. Soc. 78 (10) (2021) 878–885, http://dx.doi.org/10.1007/s40042-021-00094-2.

[6] H. Abbasimehr, M. Shabani, M. Yousefi, An optimized model using LSTM network for demand forecasting, Comput. Ind. Eng. 143 (2020) 106435, http://dx.doi.org/10.1016/j.cie.2020.106435.

[7] Y. Jin, C. Qiu, L. Sun, X. Peng, J. Zhou, Anomaly detection in time series via robust PCA, in: 2017 2nd IEEE International Conference on Intelligent Transportation Engineering, ICITE, IEEE, 2017, pp. 352–355, http://dx.doi.org/10.1109/ICITE.2017.8056937.

[8] D. Zang, J. Liu, H. Wang, Markov chain-based feature extraction for anomaly detection in time series and its industrial application, in: 2018 Chinese Control and Decision Conference, CCDC, IEEE, 2018, pp. 1059–1063, http://dx.doi.org/10.1109/CCDC.2018.8407286.

[9] W. Hu, J. Gao, B. Li, O. Wu, J. Du, S. Maybank, Anomaly detection using local kernel density estimation and context-based regression, IEEE Trans. Knowl. Data Eng. 32 (2) (2018) 218–233, http://dx.doi.org/10.1109/TKDE.2018.2882404.

[10] E.H. Budiarto, A.E. Permanasari, S. Fauziati, Unsupervised anomaly detection using K-means, local outlier factor and one class SVM, in: 2019 5th International Conference on Science and Technology, Vol. 1, ICST, IEEE, 2019, pp. 1–5, http://dx.doi.org/10.1109/ICST47872.2019.9166366.

[11] M.T. Hagan, H.B. Demuth, M. Beale, Neural Network Design, PWS Publishing Co., 1997.

[12] 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, IEEE, 2019, pp. 513–517, http://dx.doi.org/10.1109/ATIT49449.2019.9030505.

[13] Z. Qu, L. Su, X. Wang, S. Zheng, X. Song, X. Song, A unsupervised learning method of anomaly detection using gru, in: 2018 IEEE International Conference on Big Data and Smart Computing, BigComp, IEEE, 2018, pp. 685–688, http://dx.doi.org/10.1109/BigComp.2018.00126.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017) http://dx.doi.org/10.48550/arXiv.1706.03762.

[15] S. Tuli, G. Casale, N.R. Jennings, TranAD: Deep transformer networks for anomaly detection in multivariate time series data, Proc. VLDB 15 (6) (2022) 1201–1214.

[16] G. Mbiydzenyuy, Univariate time series anomaly labelling algorithm, in: International Conference on Machine Learning, Optimization, and Data Science, Springer, 2020, pp. 586–599, http://dx.doi.org/10.1007/978-3-030-64580-9_48.

[17] A. Deng, B. Hooi, Graph neural network-based anomaly detection in multivariate time series, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 5, 2021, pp. 4027–4035, http://dx.doi.org/10.1609/aaai.v35i5.16523.

[18] 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, http://dx.doi.org/10.1109/ICDM50108.2020.00093.

[19] A. Patcha, J.-M. Park, An overview of anomaly detection techniques: Existing solutions and latest technological trends, Comput. Netw. 51 (12) (2007) 3448–3470, http://dx.doi.org/10.1016/j.comnet.2007.02.001.

[20] P. Boniol, T. Palpanas, M. Meftah, E. Remy, Graphan: Graph-based subsequence anomaly detection, Proc. VLDB Endow. 13 (12) (2020) 2941–2944, http://dx.doi.org/10.14778/3415478.3415514.

[21] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 Eighth Ieee International Conference on Data Mining, IEEE, 2008, pp. 413–422, http://dx.doi.org/10.1109/ICDM.2008.17.

[22] T.R. Bandaragoda, K.M. Ting, D. Albrecht, F.T. Liu, J.R. Wells, Efficient anomaly detection by isolation using nearest neighbour ensemble, in: 2014 IEEE International Conference on Data Mining Workshop, IEEE, 2014, pp. 698–705, http://dx.doi.org/10.1109/ICDMW.2014.70.

[23] A.H. Yaacob, I.K. Tan, S.F. Chien, H.K. Tan, Arima based network anomaly detection, in: 2010 Second International Conference on Communication Software and Networks, IEEE, 2010, pp. 205–209, http://dx.doi.org/10.1109/ICCSN.2010.55.

[24] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus, B. Furht, Anomaly detection in medical wireless sensor networks using SVM and linear regression models, Int. J. E-Health Med. Commun. (IJEHMC) 5 (1) (2014) 20–45, http://dx.doi.org/10.4018/ijehmc.2014010102.

[25] Y. Wang, N. Masoud, A. Khojandi, Real-time sensor anomaly detection and recovery in connected automated vehicle sensors, IEEE Trans. Intell. Transp. Syst. 22 (3) (2020) 1411–1421, http://dx.doi.org/10.1109/TITS.2020.2970295.

[26] P. Boniol, J. Paparrizos, T. Palpanas, M.J. Franklin, SAND: streaming subsequence anomaly detection, Proc. VLDB Endow. 14 (10) (2021) 1717–1729, http://dx.doi.org/10.14778/3467861.3467863.

[27] L. Tran, M.Y. Mun, C. Shahabi, Real-time distance-based outlier detection in data streams, Proc. VLDB Endow. 14 (2) (2020) 141–153, http://dx.doi.org/10.14778/3425879.3425885.

[28] K. Kingsbury, P. Alvaro, Elle: Inferring isolation anomalies from experimental observations, 2020, http://dx.doi.org/10.48550/arXiv.2003.10554, arXiv preprint arXiv:2003.10554.

[29] W. Shang, J. Cui, C. Song, J. Zhao, P. Zeng, Research on industrial control anomaly detection based on FCM and SVM, in: 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE, IEEE, 2018, pp. 218–222, http://dx.doi.org/10.1109/TrustCom/BigDataSE.2018.00042.

[30] H.S. Dhiman, D. Deb, S. Muyeen, I. Kamwa, Wind turbine gearbox anomaly detection based on adaptive threshold and twin support vector machines, IEEE Trans. Energy Convers. 36 (4) (2021) 3462–3469, http://dx.doi.org/10.1109/TEC.2021.3075897.

[31] 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, http://dx.doi.org/10.1145/3219819.3219845.

[32] G. Zhu, H. Zhao, H. Liu, H. Sun, A novel LSTM-GAN algorithm for time series anomaly detection, in: 2019 Prognostics and System Health Management Conference, PHM-Qingdao, IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/PHM-Qingdao46334.2019.8942842.

[33] H.-S. Nam, Y.-K. Jeong, J.W. Park, An anomaly detection scheme based on lstm autoencoder for energy management, in: 2020 International Conference on Information and Communication Technology Convergence, ICTC, IEEE, 2020, pp. 1445–1447, http://dx.doi.org/10.1109/ICTC49870.2020.9289226.

[34] B. Zong, Q. Song, M.R. Min, W. Cheng, C. Lumezanu, D. Cho, H. Chen, Deep autoencoding gaussian mixture model for unsupervised anomaly detection, in: International Conference on Learning Representations, 2018, URL https://openreview.net/forum?id=BJJLHbb0-.

[35] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2828–2837, http://dx.doi.org/10.1145/3292500.3330672.

[36] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, N.V. Chawla, A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, 2019, pp. 1409–1416, http://dx.doi.org/10.1609/aaai.v33i01.33011409.

[37] 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: International Conference on Artificial Neural Networks, Springer, 2019, pp. 703–716, http://dx.doi.org/10.1007/978-3-030-30490-4_56.

[38] Y. Zhang, Y. Chen, J. Wang, Z. Pan, Unsupervised deep anomaly detection for multi-sensor time-series signals, IEEE Trans. Knowl. Data Eng. (2021) http://dx.doi.org/10.1109/TKDE.2021.3102110.

[39] J. Audibert, P. Michiardi, F. Guyard, S. Marti, M.A. Zuluaga, Usad: Unsupervised anomaly detection on multivariate time series, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3395–3404, http://dx.doi.org/10.1145/3394486.3403392.

[40] G. Li, X. Zhou, J. Sun, X. Yu, Y. Han, L. Jin, W. Li, T. Wang, S. Li, Opengauss: An autonomous database system, Proc. VLDB Endow. 14 (12) (2021) 3028–3042, http://dx.doi.org/10.14778/3476311.3476380.

[41] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, Z. Luan, Hitanomaly: Hierarchical transformers for anomaly detection in system log, IEEE Trans. Netw. Serv. Manag. 17 (4) (2020) 2064–2076, http://dx.doi.org/10.1109/TNSM.2020.3034647.

[42] H. Guo, S. Yuan, X. Wu, Logbert: Log anomaly detection via bert, in: 2021 International Joint Conference on Neural Networks, IJCNN, IEEE, 2021, pp. 1–8, http://dx.doi.org/10.1109/IJCNN52387.2021.9534113.

[43] M. Fält, S. Forsström, T. Zhang, Machine learning based anomaly detection of log files using ensemble learning and self-attention, in: 2021 5th International Conference on System Reliability and Safety, ICSRS, IEEE, 2021, pp. 209–215, http://dx.doi.org/10.1109/ICSRS53853.2021.9660694.

[44] S.R. Wibisono, A.I. Kistijantoro, Log anomaly detection using adaptive universal transformer, in: 2019 International Conference of Advanced Informatics: Concepts, Theory and Applications, ICAICTA, IEEE, 2019, pp. 1–6, http://dx.doi.org/10.1109/ICAICTA.2019.8904299.

[45] H. Mori, S. Tamura, S. Hayamizu, Anomalous sound detection based on attention mechanism, in: 2021 29th European Signal Processing Conference, EUSIPCO, IEEE, 2021, pp. 581–585, http://dx.doi.org/10.23919/EUSIPCO54536.2021.9616201.

[46] C. Zhang, X. Wang, H. Zhang, H. Zhang, P. Han, Log sequence anomaly detection based on local information extraction and globally sparse transformer model, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4119–4133, http://dx.doi.org/10.1109/TNSM.2021.3125967.

[47] H. Yuan, Z. Cai, H. Zhou, Y. Wang, X. Chen, TransAnomaly: Video anomaly detection using video vision transformer, IEEE Access 9 (2021) 123977–123986, http://dx.doi.org/10.1109/ACCESS.2021.3109102.

[48] 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. (2021) http://dx.doi.org/10.1109/JIOT.2021.3100509.

[49] Y. Li, X. Peng, J. Zhang, Z. Li, M. Wen, DCT-GAN: Dilated convolutional transformer-based GAN for time series anomaly detection, IEEE Trans. Knowl. Data Eng. (2021) http://dx.doi.org/10.1109/TKDE.2021.3130234.

[50] Y. Liu, S. Pan, Y.G. Wang, F. Xiong, L. Wang, Q. Chen, V.C. Lee, Anomaly detection in dynamic graphs via transformer, IEEE Trans. Knowl. Data Eng. (2021) http://dx.doi.org/10.1109/TKDE.2021.3124061.

[51] S. Shakya, S. Sigdel, An approach to develop a hybrid algorithm based on support vector machine and Naive Bayes for anomaly detection, in: 2017 International Conference on Computing, Communication and Automation, ICCCA, IEEE, 2017, pp. 323–327, http://dx.doi.org/10.1109/CCAA.2017.8229836.

[52] S. Kanarachos, J. Mathew, A. Chroneos, M. Fitzpatrick, Anomaly detection in time series data using a combination of wavelets, neural networks and Hilbert transform, in: 2015 6th International Conference on Information, Intelligence, Systems and Applications, IISA, IEEE, 2015, pp. 1–6, http://dx.doi.org/10.1109/IISA.2015.7388055.

[53] D. Cheng, S. Xiang, C. Shang, Y. Zhang, F. Yang, L. Zhang, Spatio-temporal attention-based neural network for credit card fraud detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 01, 2020, pp. 362–369, http://dx.doi.org/10.1609/aaai.v34i01.5371.

[54] F. Carcillo, Y.-A. Le Borgne, O. Caelen, Y. Kessaci, F. Oblé, G. Bontempi, Combining unsupervised and supervised learning in credit card fraud detection, Inform. Sci. 557 (2021) 317–331, http://dx.doi.org/10.1016/j.ins.2019.05.042.

[55] I. Guyon, et al., A scaling law for the validation-set training-set size ratio, AT&T Bell Lab. 1 (11) (1997).

[56] J. Alammar, The illustrated transformer, 2018, https://jalammar.github.io/illustrated-transformer/. (Accessed 29 December 2022).

[57] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 1126–1135, URL https://proceedings.mlr.press/v70/finn17a.html.

[58] 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, http://dx.doi.org/10.1109/CySWater.2016.7469060.

[59] G.B. Moody, R.G. Mark, The impact of the MIT-BIH arrhythmia database, IEEE Eng. Med. Biol. Mag. 20 (3) (2001) 45–50, http://dx.doi.org/10.1109/51.932724.

[60] S. Ahmad, A. Lavin, S. Purdy, Z. Agha, Unsupervised real-time anomaly detection for streaming data, Neurocomputing 262 (2017) 134–147, http://dx.doi.org/10.1016/j.neucom.2017.04.070.

[61] S. Nedelkoski, J. Bogatinovski, A.K. Mandapati, S. Becker, J. Cardoso, O. Kao, Multi-source distributed system data for ai-powered analytics, in: European Conference on Service-Oriented and Cloud Computing, Springer, 2020, pp. 161–176, http://dx.doi.org/10.1007/978-3-030-44769-4_13.

[62] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 12, 2021, pp. 11106–11115, http://dx.doi.org/10.1609/aaai.v35i12.17325.