Journal of Computers Vol. 35 No. 3, June 2024, pp. 193-207
doi: 10.53106/199115992024063503014

Spatial-temporal Attention Model Based on Transformer Architecture for Anomaly Detection in Multivariate Time Series Data

Lai Zeng*, Xiaomei Yang

The College of Electrical Engineering, Sichuan University,
Chengdu 610065, Chengdu, China
{csenglai, yangxiaomei}@stu.scu.edu.cn

Received 24 September 2023; Revised 22 January 2024; Accepted 24 March 2024

Abstract. Anomaly detection with multivariate time series data collected by multi-sensors is a challenging problem due to the complexity and high dimension of data and the difficulty of manually labelling data. This paper proposes a novel unsupervised anomaly detection model using spatial-temporal self-attention based on transformer architecture, denoted the biself-attention anomaly detection (BSAAD) model. The BSAAD model not only utilizes a time-step encoder with a self-attention mechanism to capture temporal correlation but also constructs a sensor encoder with a self-attention mechanism to capture spatial correlation among multivariate time series data. To amplify the reconstruction errors of anomalous points during network training, a two-phase training style with an adversarial training strategy is used to improve the anomaly detection performance of the BSAAD model. Experiments on six multivariate time series datasets show that the BSAAD model outperforms state-of-the-art anomaly detection methods.

Keywords: biself-attention, transformer, anomaly detection, multivariate time series data, BSAAD model

1 Introduction

Modern IT systems use multi-sensors to collect numerous multivariate time series data continuously, which represent the system status [1]. To investigate potential risks in IT systems, we need to determine whether the systems are anomalous based on these data. Conventionally, anomaly detection can be implemented by using supervised learning or unsupervised learning methods. Commonly, labelled data are required in supervised learning methods; however, a large amount of un-labelled data always exists in real-world measured data. Even if these data can be manually labelled, it is not only a time-consuming process, but it also sometimes generates incorrect labels in the case of anomalies close to normal data points, which poses challenges to supervised learning methods. Therefore, we focus on detecting anomalies in multivariate time series data in an unsupervised way.

Unsupervised anomaly detection tasks should formulate proper criteria to distinguish between anomalous and normal points. Many machine learning methods have been proposed in recent years, such as principal component analysis (PCA) [2], density-based methods represented by local outlier factor (LOF) [3], linear models with one-class SVM [4], and tree- based methods proposed in isolated forest [5]. However, the increasing complexity of data means these methods are no longer the best choices. With the development of deep learning, anomaly detection based on deep learning has become a research hotspot. LSTM-NDT [6] uses stacked long-short- term-memory (LSTM) networks to predict input sequences and models the prediction errors to infer the likelihoods of anomalies, though LSTM cannot be parallel trained, having high computational complexity. OmniAnomaly [7] captures normal patterns of time series based on a stochastic recurrent neural network and detects anomalies by reconstruction probabilities. DAGMM [8] learns a low-dimensional representation from data by an autoencoder and feeds the reconstruction error into a Gaussian mixture model for anomaly detection. MAD-GAN [9] embeds LSTM into a generative adversarial network as the generator and discriminator (GAN) to capture the temporal correlation of time series. However, due to a lack of stability, the GAN-based model has suboptimal performance on complex datasets. USAD [10] reconstructs the input data with adversely trained autoencoders and detects anomalies based on anomaly errors. The simple architecture enables USAD to provide fast training. Recently, Graph Neural Networks (GNN) [11], has achieved significant success in modeling various types of data. MTAD-GAT [12] employs two parallel graph attention mechanisms to separately handle time and feature data, resulting

* Corresponding Author

in improved accuracy. Graph Deviation Network (GDN) [13] enhances interpretability by utilizing Graph Neural Networks to learn relationships between variables.

Additionally, the Transformer [14] has been proven to have good performance in processing long time series and capturing latent associations within sequences. TranAD [15] reconstructs input data based on transformer architecture with adversarial training, which improves performance compared with USAD. Anomaly Transformer [16] uses a transformer architecture to learn the time-point associations in time series and utilizes association-based criteria to detect anomalies. This work achieves promising improvement for anomaly detection in multivariate time series.

However, these works use a self-attention mechanism only on time steps and do not learn the dependencies between multi-sensor data. This gap leaves room for our proposed improvements in capturing the intricate correlations from high- dimensional sensor data. Actually, the spatiotemporal correlation among multivariate time series from different sensors is critical for anomaly detection. For example, the Secure Water Treatment (SWaT) system [17] contains various measurements, such as water level, flow rate, valve status, etc., that are collected continuously by multi-sensors. Specifically, correlations among multi-sensor data cannot be ignored: for instance, opening valves will cause changes in the flow rate and water level, which further triggers a chain reaction of the sensors given the internal mechanism of the system. However, it is difficult to obtain the relationships among multi-sensor data in real-world scenarios, leading to the problem of how to capture the hidden dependencies among sensors in multivariate time series.

To address this issue, we propose a novel method using spatial-temporal attention in multivariate time series for anomaly detection based on transformer architecture, denoted as the biself-attention anomaly detection (BSAAD) model. The BSAAD model contains a time-step encoder and a sensor encoder, where the time-step encoder uses self-attention to learn long-term dependencies along the time-step dimension. while the sensor encoder uses self-attention to capture the correlations along the sensor-spatial dimension. Once the features from the two encoders are fused, we feed them into decoders to generate reconstruction of input sequence. Then, the anomaly label can be predicted by computing the reconstruction error between the input sequence and its reconstruction. Further- more, to enhance the prediction performance, we use a two- phase training style with an adversarial training strategy to amplify the reconstruction errors of anomalous points, which aids in better distinguishing between normal and anomalous points. Thus, the architecture of the BSAAD model allows us to predict anomalies in multivariate time series accurately. The contributions of our work are as follows:

- We propose a biself-attention framework to capture long-range correlations from both time steps and spatial sensors based on transformer architecture. The temporal-spatial correlations can be extracted in parallel by thetime step encoder and sensor encoder.
- We use a two-phase training style with an adversarial training strategy, which is conducive to amplifying the reconstruction errors of anomalous points and improving performance for anomaly detection.
- We perform experiments on six multi-sensor datasets. The results show that the BSAAD model has better performance than baseline models in anomaly detectionin multivariate time series.

The remainder of this work is organized as follows. We introduce the preliminaries in Section 2. The details of the proposed BSAAD model are introduced in Section 3. Section 4 describes the experiments and the performance of the BSAAD model. Finally, we summarize this work in Section 5.

2 Preliminaries

In this section, we will introduce the fundamental task of multivariate time series anomaly detection and the basic concepts of attention mechanism.

2.1 Problem Formulation

To continuously monitor the system, measures are captured from different sensors. Commonly, the observations are recorded at equal-space timestamps. These observed time series $\mathcal{T} \in \mathbb{R}^{m \times T}$ are denoted as

$$\mathcal{T} = \{x_1, x_2, \cdots, x_T\}, \tag{1}$$

where $x_t \in \mathbb{R}^m$ denotes the observation at a specific timestamp t, m > 1 denotes that $\mathcal{T}$ is a multivariate time series, i.e., containing more than one observation at t, and T denotes the length of the time series.

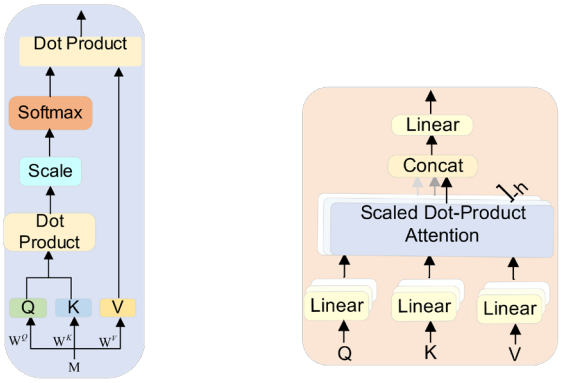
To implement anomaly detection, it is necessary to establish the mapping relationship $f(*)$ between $\mathcal{T}$ and its output Y,

$$Y = f(\mathcal{T}), \tag{2}$$

where $Y = \{y_1, y_2, \ldots, y_T\}$, and y_t is a binary label vector, given by

$$y_t = \begin{cases} 0, & x_t \in \text{normal} \\ 1, & x_t \in \text{abnormal}. \end{cases} \tag{3}$$

Using the constructed $f(*)$ on testing data $\hat{\mathcal{T}} = \{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_T\}$, the predicted output determines whether $\hat{x}_t$ is anomalous or not. Due to the scale and structural complexity of multi-variables, anomaly detection of multivariate time series is more challenging than that of univariate time series.



(a) Scaled dot-product attention block (b) Multi-head attention block

Fig. 1. Scaled dot-product attention block and multi-head attention block

2.2 Self-attention Module

Self-attention, as an attention mechanism, is used to determine the relationships between objects at different locations within a sequence [14]. For example, in the sentence "The girl sang as she walked", the two words "girl" and "she" refer to the same object, having a high correlation. In anomaly detection applications, this attention mechanism can be utilized to capture relationships in multivariate time-series data, which helps the model identify abnormalities more effectively. The self-attention module is used to construct the network in the proposed BSAAD model, where the self-attention module maps the input sequence matrix M into three matrices, the query matrix (Q), key matrix (K) and value matrix (V), as follows

$$Q = MW^Q,\ K = MW^Q,\ V = MW^Q, \tag{4}$$

where (W^Q, W^Q, W^Q) are the parameters to be learned. Although there are various ways to implement self-attention, "scaled dot-product attention (SDPA") [14], as shown in Fig. 1(a), is adopted to compute the dot products of the queries with all keys divided each by dk. Finally, a softmax function is applied to obtain the weight attention on the values:

$$\text{Attention}\left(Q,K,V\right)=\text{softmax}\left(\frac{QK^{\text{T}}}{\sqrt{d_k}}\right)V, \tag{5}$$

where $(*)^{\text{T}}$ is a transpose operation and d_k denotes the dimension of the key vector.

3 Methodology

In this section, we introduce our proposed BSAAD model, including data preprocessing, model architecture, two-phase training, and anomaly detection.

3.1 Data Preprocessing

To handle the different scales in the values of different dimensions, we normalize the multivariable data on the range [0,1) for both training data and testing data as follows

$$x_t'=\frac{x_t-\min\left(\mathcal{T}\right)}{\max\left(\mathcal{T}\right)-\min\left(\mathcal{T}\right)+\epsilon}, \tag{6}$$

where $\max(\mathcal{T})$ and $\min(\mathcal{T})$ denote the maximum and minimum of $\mathcal{T}$, respectively, and ϵ is a small constant to prevent zero-division.

Compared with information from a single point x_t, historical data points before x_t usually provide more information and contribute to a better understanding of x_t; thus, instead of just x_t at the current timestamp t, a windowed sequence $W_t\in\mathbb{R}^{\text{m}\times K}$ with length K, defined as

$$W_t=\left\{x_{t-K+1}',x_{t-K+2}',\cdots,x_t'\right\}, \tag{7}$$

is used to model time series $W = \{W_1, W_2, \ldots, W_{T-K+1}\}$. Thus, we use W and $\hat{W}$ rather than $\mathcal{T}$ and $\hat{\mathcal{T}}$ in Section 4 as the training input and testing input, respectively.

3.2 Model Architecture

BSAAD model follows the encoder–decoder structure, comprised of three main substructures: encoder submodule consisting of a window encoder, sensor encoder and time-step encoder, feature fusion submodule, and decoder submodule, as shown in Fig. 2. In our model, the window encoder utilizes the windowed sequence W_t as input. To leverage a larger context, the complete input sequence $C\in\mathbb{R}^{\text{m}\times L}$ for an input batch is taken as the input of sensor encoder and time step encoder, where L is the sequence length of C. In other word, the complete sequence C contains all of the samples in an input batch, such sequence contains long temporal trends in time series, which allows model utilize the wider context information to reconstruct W_t. The sensor encoder and the time step encoder operate on C with a focus score to capture spatial correlations among the different sensors and temporal correlations among different time stamps, respectively. Furthermore, temporal and spatial information are fused in the feature fusion submodule, which is then fed into window encoder to obtain a latent representation of the input window W_t, then the decoders use this to generate reconstruction of the input window W_t. More details of the abovementioned substructures will be discussed in the following. To match the upcoming layers, we will first use the learned embeddings to map the inputs to vectors with dimension d_m.

(1) Positional Embedding: To enhance distinctions between datapoint features, position embedding is used to inject the relative position information into the multivariate sequence. First, the input sequences are transformed into a matrix form with modality dm. Then, we adopt sine and cosine functions of different frequencies as the positional encoding [14], given by

$$\mathrm{PE}(t,2k)=\sin\left(\frac{t}{10000^{\frac{2k}{d_m}}}\right), \tag{8}$$

$$\mathrm{PE}(t,2k+1)=\cos\left(\frac{t}{10000^{\frac{2k}{d_m}}}\right), \tag{9}$$

Where $\mathrm{PE}(t, 2k)$ and $\mathrm{PE}(t, 2k + 1)$ denote the even and odd dimensions of the positional encoding layer, respectively, and k is the dimension index in the range of $[1, d_m/2]$. Positional embedding has the ability to associate information across multivariate sequences with spatial positional awareness.

(2) Sensor Encoder: The sensor encoder uses the multi-head self-attention mechanism to capture the correlations among different sensors. After the complete sequence C and focus score F, which represents the deviation between the real input and reconstruction input (more details can be found in Section 3.3), are concatenated to matrix $C_F \in \mathbb{R}^{\mathrm{m}\times(L+K)}$, denoted as

$$C_F = Concat(C,F), \tag{10}$$

then, the input embedding and positional encoding are applied on C_F to obtain $X_s \in \mathbb{R}^{\mathrm{m}\times d_m}$.

To jointly capture the information of different representation subspaces from different sensors, multi-head attention, concatenated from h self-attention, as shown in Fig. 1(b), is applied on X_s as:

$$\mathrm{MultiHead}_s\left(X_s,X_s,X_s\right)=\mathrm{Concat}(\mathrm{Head}_{s,1},...,\mathrm{Head}_{s,h})W_s, \tag{11}$$

where $W_s \in \mathbb{R}^{hd_m\times d_m}$, and $\mathrm{Head}_{s,i}$ denotes the i-th self-attention, given by

$$\mathrm{Head}_{s,i}=\mathrm{Attention}\left(X_sW_{s,i}^{Q},X_sW_{s,i}^{K},X_sW_{s,i}^{V}\right), \tag{12}$$

where ($W_{s,i}^{Q}$, $W_{s,i}^{K}$, $W_{s,i}^{V}$) are the parameters to be learned related to the Q matrix, K matrix and V matrix, respectively, in the i-th self-attention.

Two skip connections are used to alleviate the difficulty of training deep neural networks. The output feature $X_s^2 \in \mathbb{R}^{\mathrm{m}\times d_m}$ of the sensor encoder layer is obtained as:

$$X_s^1=\mathrm{Norm}\left(X_s+\mathrm{MultiHead}_s\left(X_s,X_s,X_s\right)\right), \tag{13}$$

$$X_s^2=\mathrm{Norm}\left(X_s^1+\mathrm{FeedForward}\left(X_s^1\right)\right), \tag{14}$$

where $\mathrm{Feedforward}(.)$ denotes the forward neural layer, $\mathrm{Norm}(.)$ denotes layer normalization, which can improve the training speed and make the model more robust, and the symbol "+" denotes matrix addition according to the data and their residual.

(3) Time-step Encoder: The time-step encoder obtains dependencies from different time steps, similar to the sensor encoder, by using a multi-head self-attention mechanism. In contrast to using C_F in the sensor encoder,

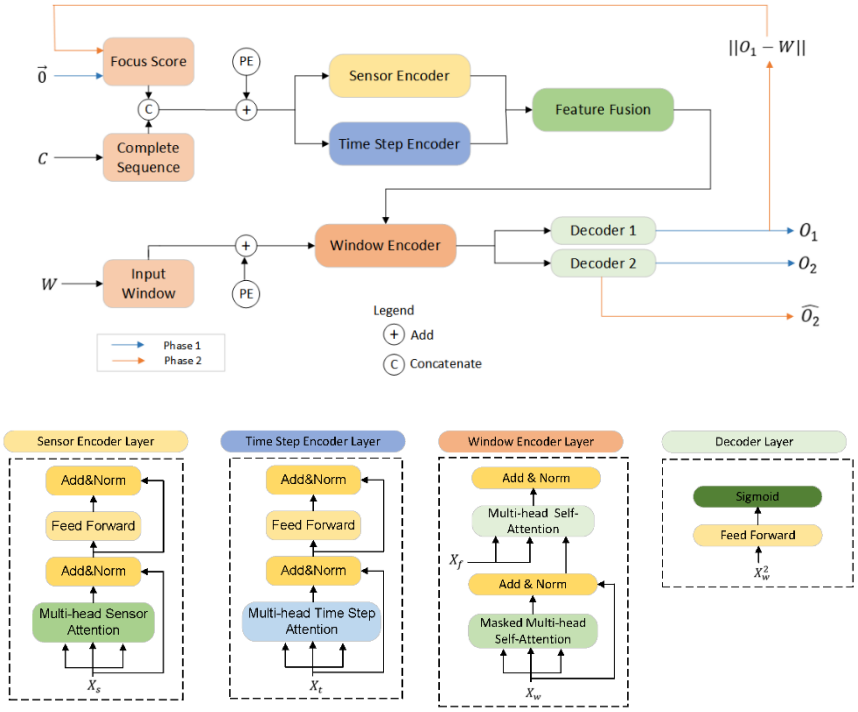


Fig. 2. Architecture of the BSAAD model

position embedding is applied to the transpose of matrix data C_F to obtain X_t as the input of the time-step encoder, $X_t \in \mathbb{R}^{(L+K)\times d_m}$. Accordingly, the output feature $X_t^2 \in \mathbb{R}^{(L+K)\times d_m}$ of the time-step encoder is given by

$$X_t^1 = \text{Norm}\left(X_t + \text{MultiHead}_t\left(X_s, X_s, X_s\right)\right), \tag{15}$$

$$X_t^2 = \text{Norm}\left(X_s^1 + \text{FeedForward}\left(X_s^1\right)\right), \tag{16}$$

where

$$\text{MultiHead}_t\left(X_t, X_t, X_t\right) = \text{Concat}(\text{Head}_{t,1}, ..., \text{Head}_{t,h})W_t, \tag{17}$$

where $W_t \in \mathbb{R}^{hd_m \times d_m}$ are the parameters to be learned for the time-step encoder.

(4) Feature Fusion: To fuse the temporal and spatial features from the time-step encoder and the sensor encoder, respectively, the feature fusion module is utilized to form a new feature X_f, given by

$$X_f = \text{Concat}_s\left(X_s^2, X_t^2\right), \tag{18}$$

where $X_f \in \mathbb{R}^{(m+L+K)\times d_m}$, and Concat_s denotes the concatenate operation along the sensor dimension, allowing the model to capture feature information from both the sensor dimension and time-step dimension.

(5) Window Encoder: Similarly, we apply input embedding and positional encoding to the input window data W_t to obtain $X_w \in \mathbb{R}^{m\times d_m}$. The window encoder includes two multi-head self-attentions. The first multi-head attention performs a mask operation on X_w, which is applied to conceal the window sequences for future timestamps within the same input batch. The second multi-head self-attention receives keys and values from the fused feature X_f of the sensor and time-step encoders, which allows the window encoder for the attention operation to utilize different time steps and 'sensor information from the complete se-

quence. The queries of the second multi-head self-attention are from the output of the previous masked multi-head self-attentions. Finally, the window encoder performs the following operation:

$$X_w^1 = \text{Norm}\left(X_w + \text{Mask}\left(\text{MultiHead}_t\left(X_s, X_s, X_s\right)\right)\right), \tag{19}$$

$$X_w^2 = \text{Norm}\left(X_w^1 + \text{MultiHead}\left(X_w^1, X_f, X_f\right)\right), \tag{20}$$

where $X_w^2 \in \mathbb{R}^{m \times d_m}$ is the output feature of the window encoder.

(6) Decoder 1 and Decoder 2: To implement adversarial training with stability in the subsequent procedure, two separate decoders, Dec1 and Dec2, consisting of a feedforward network and sigmoid function, are used to analyze the weight features of different sensors and time steps extracted from complete sequences and window sequences. Two decoders perform the operation on

$$O_i = \text{Sigmoid}\left(\text{FeedForward}\left(X_w^2\right)\right) i = 1, 2, \tag{21}$$

where $O_i \in \mathbb{R}^{m \times K}$ denotes the outputs of Dec1 or Dec2. To match the normalized input window W_t, we use the sigmoid function to map the outputs on the range [0, 1]. After the above operations, we take C and W_t as inputs and obtain the reconstructed input windows O_1 and O_2

3.3 Two-phase Training

Similar to other encoder-decoder models, the BSAAD model predicts the anomaly label using the reconstruction error, i.e., the deviation between reconstructed and original inputs. However, traditional encoder-decoder models tend to ignore anomalies when the reconstruction error is not large enough. To solve this problem, we train our model in a two-phase adversarial training style to amplify the reconstruction error. The focus score F, obtained from the reconstruction loss of Dec1 in the first phase, is introduced into the second phase as a prior of the short-range temporal trends and allows the BSAAD model to focus on the anomaly points by assigning higher neural network activation to anomaly points in the time series. As Fig. 3 shows, if the trend of the time series changes suddenly (This implies there may have been anomalies), F would be higher. Additionally, according to F, the sensor encoder and time step encoder will modify the attention weights for the anomaly points, which further prompts the model generate a poor reconstruction for anomalies.

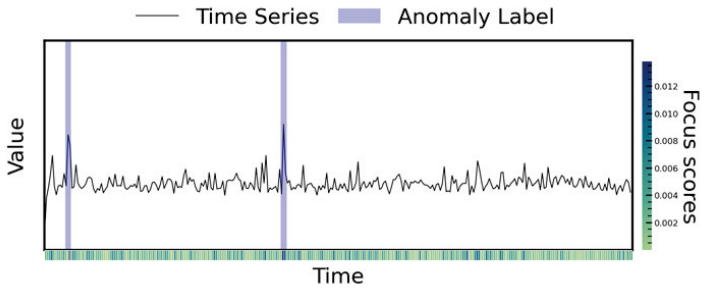


Fig. 3. Visualization of focus scores

1) Phase 1: *Reconstruct Input Window*: In this phase, the focus score F is initialized as a zero matrix, i.e., $F = [0]_{L \times m}$. Two decoders have the same objectives and attempt to reconstruct the input window W_t as well as possible. The reconstruction losses of Dec1 and Dec2 are defined as:

$$L_1^1 = \left\| O_1 - W_t \right\|_2, \tag{22}$$

$$L_2^1 = \left\| O_2 - W_t \right\|_2, \tag{23}$$

respectively, where $\|.\|_2$ denotes the L2 norm.

2) Phase 2: *Reconstruct Input Window with Focus Score*: In this phase, the reconstruction loss of Dec1 is used as the focus score, i.e., $F = L_1^1$, which allows the encoders to focus on data points with high deviations. In addition, to improve the robustness of our model, Dec1 and Dec2 are trained in an adversarial way. At this phase, Dec1 is trained to fool Dec2 by reconstructing the input window W_t as perfectly as possible, while Dec2 is trained to distinguish between input window W_t and the reconstruction of Dec1. Thus, Dec1 would generate a perfect reconstruction (i.e., $O_1 = W_t$) and F would be a zero matrix in both phase 1 and phase 2, which prompts Dec2 to generate the same outputs in two phases (i.e., $O_2 = \hat{O}_2$). Using the adversarial training strategy, we can simplify the objective of Dec1 as minimizing the deviation between output $\hat{O}_2$ of Dec2 and input window W_t, while Dec2 maximizes the deviation between $\hat{O}_2$ and W_t; thus, the adversarial loss is defined as:

$$\min_{\text{Dec1}} \max_{\text{Dec2}} \left\| \hat{O}_2 - W_t \right\|_2. \tag{24}$$

Furthermore, the adversarial losses for Dec1 and Dec2 can be rewritten from (20) as

$$L_1^2 = + \left\| \hat{O}_2 - W_t \right\|, \tag{25}$$

$$and\ L_2^2 = - \left\| \hat{O}_2 - W_t \right\|_2, \tag{26}$$

respectively. (19) and (21) show that the two decoders aim to minimize the reconstruction losses in phase 1, while they have opposite objectives in phase 2.

For the sake of training stability, the losses of two phases need to be combined in an evolutionary way. We define the final cumulative losses as

$$L_1 = \frac{1}{n} \left\| O_1 - W_t \right\|_2 + \left(1 - \frac{1}{n} \right) \left\| \hat{O}_2 - W_t \right\|_2, \tag{27}$$

$$L_2 = \frac{1}{n} \left\| O_2 - W_t \right\|_2 - \left(1 - \frac{1}{n} \right) \left\| \hat{O}_2 - W_t \right\|_2, \tag{28}$$

where n is a training epoch and $1/n$ and $(1-1/n)$ are the weights for the reconstruction loss and the adversarial loss, respectively. At the beginning of training, two decoders generate poor reconstructions in phase 1. Thus, the focus score obtained by the reconstruction loss is unreliable; therefore, we give higher weights to the reconstruction loss. With increases in n, the reconstructions of decoders move closer to W_t in phase 1, the focus score becomes valuable, and the adversarial loss is given a higher weight.

3.4 Anomaly Detection

After the parameters of the proposed BSAAD model are trained by the online training procedure, the BSAAD model can be applied to the test window input $\hat{W}_t$ and the complete sequence $\hat{C}$. Instead of obtaining the anomaly label directly, we first calculate the anomaly score for this window, given as

$$s = \frac{1}{2}\left\|O_1 - \hat{W}_t\right\|_2 + \frac{1}{2}\left\|\hat{O}_2 - \hat{W}_t\right\|_2, \tag{29}$$

where O_1 is the reconstruction output of Dec1 in phase 1 and $\hat{O}_2$ is the reconstruction output of Dec2 in phase 2. We combine the outputs for two phases to obtain the anomaly score. The anomaly score for the i-th ($i \le m$) variant in $\hat{W}_t \in \mathbb{R}^{m \times K}$ is denoted s_i. Given a threshold, the data points of the anomaly scores that are larger than the given thresholds are labelled anomalies. As other works [7, 18] have determined, the anomaly thresholds are automatically determined based on the peaks over threshold (POT) method [18]. The POT method takes the samples exceeding a given sufficient threshold as observations and fits the distribution with a GPD (generalized Pareto distribution) to infer the distribution of the extreme values. Then, we can obtain the threshold for dimension i by the POT method, denoted $\mathrm{POT}(s_i)$. Furthermore, the anomaly label y_t can be defined as

$$y_t^{(i)} = \begin{cases} 1, \ if \ s_i > \mathrm{POT}(s_i) \\ 0, \ else \end{cases}, \tag{30}$$

$$y_t = \bigvee_i y_t^{(i)}, \tag{31}$$

where $y_t^{(i)}$ is the anomaly label of the i-th variant $\hat{x}_t$ at timestamp t.

4 Experimental Setup and Results

We trained all models using the PyTorch-1.11.0 library, while adopting the hyperparameters of the baseline models as specified in their respective papers. The hyperparameters are set as follows.

- Window length=10.
- Dropout in all encoders=0.1.
- Hidden units in input embedding layer=128.
- Heads in sensor encoder layer=8.
- Heads in time step encoder layer=8.

For training, we use AdamW optimizer with learning rate of 0.01 and batchsize is 128. All experiments are performed on a Windows 11 workstation, which is equipped with 32-GB RAM, an Intel i5-11400 CPU and a RTX3060 GPU. In later section, we first introduce the datasets used in our experiments and the evaluation metrics for anomaly detection and then conduct comparison and ablation experiments and sensitivity analysis.

Table 1. The characteristics of six datasets

Dataset name	Training set size	Test set size	No. of dimensions	No. of entities	Anomaly ratio (%)
SMD	708405	708420	38	28	4.16
SWaT	496800	449919	51	1	11.98
MSL	58317	73729	27	55	10.72
WADI	1048571	172801	123	1	5.99
SMAP	135183	427617	55	25	13.13
MSDS	146430	146430	10	1	5.37

4.1 Datasets

We use six commonly used public datasets to demonstrate the effectiveness of our model. The basic characteristics of the datasets are summarized in Table 1. These datasets can be briefly described as follows.

- Server Machine Dataset (SMD): The SMD dataset, introduced by [7], contains time-sequence data collected by 28 different machines, each with 38 metrics. Due to many trivial sequences in the dataset, as noted by [15], we use only the nontrivial sequences (i.e., machine-1-1, 2-1, 3-2 and 3-7) in our experiments. The models are separately trained for each model, and the average performance is evaluated.
- Secure Water Treatment (SWaT): The SWaT dataset [17] is collected by a water treatment system, which ran continuously for 11 days with normal operations for the first 7 days and attacked for 4 days.
- Soil Moisture Active Passive and Mars Science Laboratory (SMAP&MSL): SMAP and MSL datasets [18] contain the data of surface and subsurface soil moisture, soil moisture profile, and soil moisture anomalies collected by NASA. Similar to the SMD dataset, SMAP and MSL have many trivial sequences, as noted by [20]; thus, we use only nontrivial telemetry channels (i.e., A4, C2 and T1).
- Water Distribution (WADI): Similar to the SWaT dataset, the WADI dataset is collected by the WADI testbed [21], which ran continuously for 16 days with normal operations for 14 days and attacked for 2 days.
- Multi-Source Distributed System (MSDS): As a multisource dataset, MSDS contains distributed metrics, logs, and tracing data collected by a complex distributed system [22].

4.2 Evaluation Metrics

To evaluate anomaly detection results, we use the precision (P), recall (R), and F1 score (F1), calculated as follows:

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}},\ \mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}},\ \mathrm{F1} = 2\cdot\frac{\mathrm{P}\cdot\mathrm{R}}{\mathrm{P}+\mathrm{R}}, \tag{32}$$

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

4.3 Comparison Experiments

To demonstrate the effectiveness of the BSAAD model for anomaly detection in multivariate time series, we compare it with seven popular network models, including IF [5], DAGMM [8], LSTM-NDT [6], MAD-GAN [9], OmniAnomaly [7], TranAD [15], USAD [10], GDN [13] and CAE-M [23]. The Isolation Forest (IF) used in this work is from the scikit-learn implementation. The codes of other methods except for DAGMM are open-sourced, and DAGMM is reproduced based on the settings specified in [8]. All these models are trained and tested on six datasets. Then, the performances for anomaly detection are shown in Table 2.

DAGMM uses a deep autoencoder Gaussian mixture model for feature space reduction, capturing correlations between different features in the time series. However, its modeling capability for temporal dependencies in the time dimension is relatively weak. The LSTM-NDT model performs well on the SMD dataset but exhibits poor performance on other datasets. This is attributed to the LSTM-NDT's use of LSTM structures to model the temporal dependencies in the data, without capturing the correlations between different features in space. Additionally, its reliance on the NDT thresholding method makes it sensitive to dataset variations, as it cannot dynamically set thresholds based on sequence changes [15]. The POT method used in BSAAD fits the distribution of extreme values in the sequence, allowing it to dynamically set thresholds adaptively. Fig. 4 illustrates the visualization of anomaly detection in the first dimension of the SMD test set, indicating that the POT method can accurately set anomaly thresholds. MAD-GAN and USAD share similar drawbacks, limiting their anomaly detection performance. In contrast, BSAAD employs a multi-head attention mechanism to encode time series at both temporal and spatial levels, capturing both temporal dependencies and spatial correlations in the data, effectively enhancing the anomaly detection performance of BSAAD.

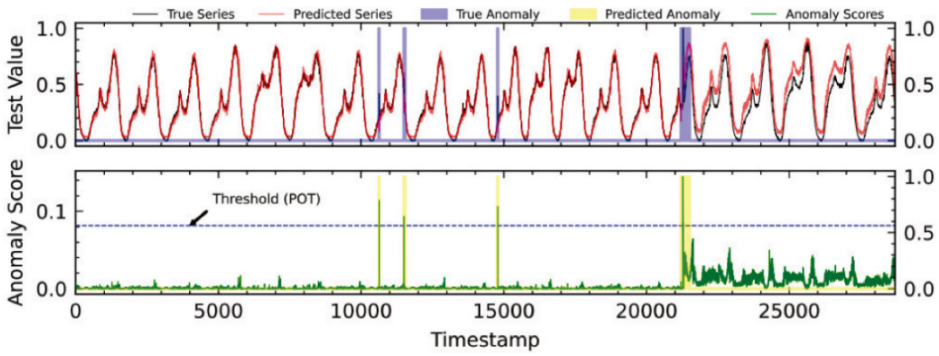


Fig. 4. Visualization of anomaly prediction for the SMD test set (machines 3-7, dimension 0) using the BSAAD model

Table 2. Evaluation metrics for anomaly detection on different datasets (P: Precision, R: Recall)

Method	SMD			SWaT			MSL		
	P	R	F1	P	R	F1	P	R	F1
IF	0.2129	0.9988	0.3374	0.4139	0.9359	0.5740	0.6250	1.0	0.7692
DAGMM	0.7416	0.8957	0.7820	0.9034	0.7897	0.8427	0.7363	0.9999	0.8481
LSTM-NDT	0.8016	0.8947	0.8329	0.9171	0.7213	0.8075	0.5678	1.0	0.7243
MAD-GAN	0.8830	0.7160	0.7598	0.9552	0.7058	0.8118	0.8516	0.9999	0.9199
OmniAnomaly	0.9119	0.8753	0.8914	0.9636	0.7708	0.8565	0.7848	0.9999	0.8794
TranAD	0.9410	0.9136	0.9246	0.9510	0.7704	0.8512	0.9038	0.9999	0.9495
USAD	0.8800	0.9036	0.8890	0.9663	0.7670	0.8552	0.7949	0.9999	0.8857
GDN	0.8142	0.8949	0.8544	0.9697	0.6957	0.8101	0.9293	0.9844	0.9371
CAE-M	0.8913	0.8931	0.9127	0.9644	0.6839	0.8006	0.7862	1.0	0.8769
BSAAD	0.9661	0.9229	0.9419	0.9435	0.8083	0.8707	0.9038	0.9999	0.9495

Method	WADI			SMAP			MSDS		
	P	R	F1	P	R	F1	P	R	F1
IF	0.0554	0.9999	0.1049	0.4269	0.9999	0.5984	0.8491	0.9974	0.9172
DAGMM	0.4797	0.8296	0.6079	0.8069	0.9999	0.8931	0.9999	0.8026	0.8905
LSTM-NDT	0.2227	0.8296	0.3512	0.8523	0.7326	0.7879	0.9997	0.1852	0.3125
MAD-GAN	0.4545	0.6140	0.5224	0.7974	0.9999	0.8873	0.9999	0.6107	0.7583
OmniAnomaly	0.7745	0.6541	0.7092	0.8009	0.9999	0.8894	0.9999	0.7964	0.8867
TranAD	0.3993	0.8296	0.5391	0.8043	0.9999	0.8915	0.9999	0.8026	0.8905
USAD	0.2096	0.8296	0.3347	0.8139	0.9999	0.8974	0.9999	0.7959	0.8863
GDN	0.2912	0.7931	0.4260	0.7479	0.9833	0.8501	0.9989	0.8026	0.8900
CAE-M	0.3142	0.8013	0.4307	0.8082	0.9433	0.8746	0.9901	0.8392	0.9110
BSAAD	0.7557	0.8296	0.7909	0.8842	0.9999	0.9385	0.9999	0.8790	0.9356

As Table 2 shows, our BSAAD model achieves the best average F1 score for all six datasets among the eight models. For example, BSAAD achieves up to 0.0116 and 0.2518 improvements in the F1 score, respectively, over the second best TranAD model [15] on the WADI dataset. TranAD can capture long-term correlations between data points because it applies self-attention to sequences along the time-step dimension; thus, TranAD has the second best performance but has poor results on the WADI dataset due to the high sensor dimension. Among all the models, the IF model has relatively poor performance on most datasets since it does not utilize temporal information. Actually, for anomaly detection, temporal information is essential to exploit temporal trends between data points. USAD takes a sequence of observations containing short-term temporal information as input and thus has good performance on SMAP but performs relatively poorly on complex datasets. OmniAnomaly performs well on the SWaT and WADI datasets because it models complex time series data and captures temporal dependencies in time series by GRU. Compared with other models, our BSAAD model not only applies

self-attention to time steps but also applies self-attention to sensors, enabling it to capture long-range dependencies from both aspects, which leads to good performance on large-scale datasets (SMD, WADI) in terms of the sequence length and number of sensors.

4.4 Ablation Experiments

To verify the effectiveness of the sensor encoder, focus score, and adversarial training in our BSAAD model, we perform ablation experiments using the WADI, SWaT, and MSDS datasets. This is due to the fact that these datasets contain sensor dimensions and sequence lengths of different scales (see Table 1). The experimental setup is as follows:

- Without (w/o) Sensor Encoder: only self-attention along the time-step dimension is applied to the input sequences.
- Without (w/o) Focus Score: the focus score used to amplify the reconstruction error is discarded, i.e., fixed to be a zero matrix.
- Without (w/o) Adversarial Training: one decoder rather than the original two decoders is used to reconstruct the input window.

The experimental results of F1 score are shown in Fig. 5, which show that removing the sensor encoder greatly decreases the F1 score on the WADI dataset and results in slight decreases in the SWaT and MSDS datasets. Thus, the sensor encoder helps to improve performance, especially for high-dimensional datasets. After we remove the focus score, the F1 score slightly decreases for the three datasets. These results shows that the focus score aids in improving the prediction performance. Not using the adversarial training strategy degrades the prediction performance most, as indicated by the F1 score on the WADI and SWaT datasets, because adversarial training can enhance the robustness of our proposed model.

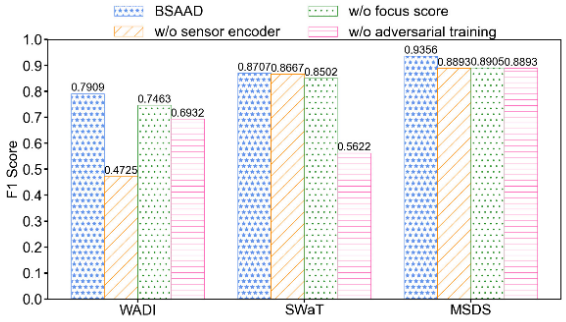


Fig. 5. The results of ablation experiments

4.5 Sensitivity Analysis

To study the sensitivity of the BSAAD model, we change the ratio of the training set and window length to evaluate the performance of the BSAAD model. As the SWaT dataset has relatively moderate dimensions and sequence lengths compared to other datasets (see Table 1), the experiments for sensitivity analysis are performed on the SWaT dataset.

Different ratios of the training set (20%-100%) are used to observe the performance variation for the BSAAD model in terms of the F1 score. For comparison, we also use other models to perform the experiments on the same ratio of the training set. As Fig. 6 shows, the BSAAD model is relatively insensitive to changes in the size of the training dataset. Because of the usage of the time window and transformer architecture, the BSAAD model can fully exploit temporal trends, which leads to relatively stable performance.

As the window length is changed, the F1 score of BSAAD and its ablated versions are shown in Fig. 7 BSAAD and its ablation version (w/o sensor encoder) have close performance on SWaT dataset for diffrent window lengths, whereas the performance of BSAAD on WADI dataset surpasses that of its ablated version by a sig-

nificant margin (see Fig. 5(a)). This is because the sensor encoder facilitates modeling high-dimensional datasets. Moreover, a large window does not improve the performance of anomaly detection since a short anomaly, especially a point anomaly, may be undetected due to being hidden within many data points, while a window that is too small would no longer represent the short-term temporal trends in the time series and fail to detect anomalies. For comprehensive consideration, a window length of 10 is a reasonable selection for the BSAAD model.

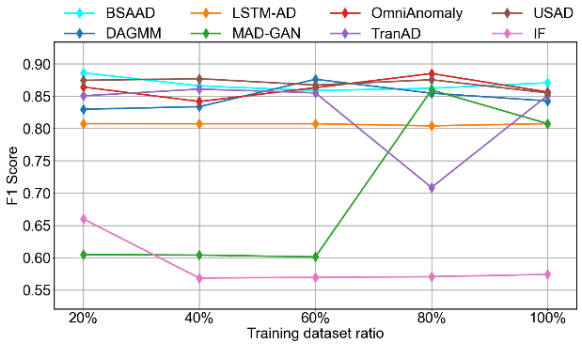


Fig. 6. F1 score with different training dataset ratios on SWaT dataset

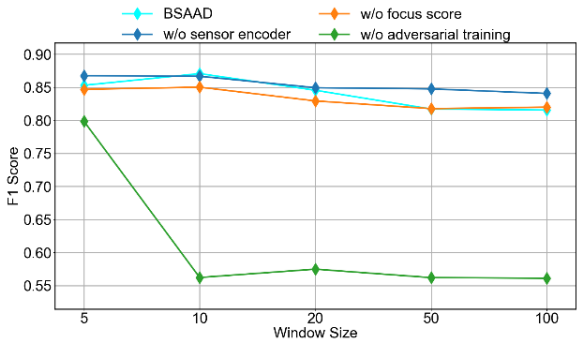


Fig. 7. F1 score with different window lengths on SWaT dataset

4.6 Validation of the Effectiveness of the POT Method

In practical engineering applications, the selection of anomaly thresholds is crucial for improving anomaly detection performance. To validate the effectiveness of the POT method proposed in this paper, we first enumerate all possible anomaly thresholds, denote the best F1 score as $F1_{best}$, and compare it with the F1 score obtained by setting anomaly thresholds using the POT method, denoted as $F1_{POT}$. The comparison results are presented in Table 3. From Table 3, it can be observed that the F1 score obtained through the POT method is only slightly lower than $F1_{best}$ (0.0029 to 0.0131). This indicates the effectiveness of setting anomaly thresholds through the POT method.

Table 3. Comparison between the F1 score obtained through the POT method and $F1_{best}$

Evaluation metrics	SMD	SWaT	MSL	WADI	SMAP	MSDS
$F1_{POT}$	0.9419	0.8707	0.9495	0.7909	0.9385	0.9356
$F1_{best}$	0.9550	0.8736	0.9537	0.8012	0.9482	0.9401

4.7 Training Time

In this section, we evaluate the computational performance of the BSAAD model by assessing its training time and comparing it with benchmark models. Table 4 presents the average training time for a single iteration of BSAAD and benchmark models on different datasets. In comparison to the benchmark models, BSAAD utilizes a Transformer-based spatiotemporal self-attention mechanism to extract spatiotemporal dependencies in parallel. This allows for simultaneous consideration of information from other neurons when processing each neuron, enabling parallel computation. Not only does BSAAD enhance performance in unsupervised anomaly detection, but it also reduces training time compared to the baseline models, which has a comparatively inferior overall performance.

Table 4. Training time in seconds per epoch on each dataset

Models	SMD	SWaT	MSL	WADI	SMAP	MSDS
DAGMM	204.05	62.51	18.46	169.34	23.19	114.12
LSTM-NDT	284.17	70.43	32.77	286.79	28.36	389.66
MAD-GAN	155.56	106.44	28.52	287.60	39.08	340.10
OminAnomaly	134.48	129.67	29.45	273.28	138.80	302.83
TranAD	38.42	54.77	8.97	112.14	4.67	17.49
USAD	85.97	106.32	28.39	239.47	30.31	34.82
GDN	309.94	2128.71	90.93	3947.02	65.32	476.37
CAE-M	983.09	1844.91	602.84	4534.53	194.11	409.14
BSAAD	33.42	43.57	11.58	96.41	9.84	14.81

5 Conclusion

In this paper, we propose the BSAAD model by using biself-attention in multivariate time series to implement unsupervised anomaly detection. The BSAAD model contains two key components, i.e., a sensor encoder and a time-step encoder, which capture the correlations along the sensor dimension and time-step dimension, respectively. The two-phase training allows the BSAAD model to amplify the reconstruction error, and adversarial training improves training stability. The incorporation of the POT adaptive threshold method further enhances the performance of STSAAD. In comparison to traditional fixed-threshold methods, the POT approach dynamically sets the threshold based on the data distribution, contributing to the improved robustness of anomaly detection. We use six public datasets to test and verify the performance of the model. The results show that the BSAAD model outperforms some state-of-the-art methods in terms of the F1 scores. Specially, BSAAD achieves an improvement of 0.0173 for F1 score on WADI datasets. With lower training times compared to the baseline methods, BSAAD proves to be an ideal choice for modern industrial systems where accurate and rapid anomaly predictions are essential.

For the future, we will consider applying model-agnostic meta learning (MAML) to our model to achieve better performance on small-scale datasets. Additionally, we aim to explore the application of cost-benefit analysis for each model component based on the deployment setting, aiming to avoid expensive computations.

References

[1] J. Goh, S. Adepu, M. Tan, Z.S. Lee, Anomaly detection in cyber physical systems using recurrent neural networks, in: Proc. 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), 2017.

[2] S. Li, J. Wen, A model-based fault detection and diagnostic methodology based on pca method and wavelet transform, Energy and Buildings 68(2014) 63–71.

[3] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, Lof: identifying density-based local outliers, in: Proc. 2000 ACM SIGMOD international conference on Management of data, 2000.

[4] B. Scholkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional

distribution, Neural computation 13(7)(2001) 1443–1471.
[5] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: Proc. 2008 eighth IEEE international conference on data mining, 2008.
[6] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, in: Proc. 2018 ACM SIGKDD international conference on knowledge discovery & data mining, 2018.
[7] 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: Proc. 2019 ACM SIGKDD international conference on knowledge discovery & data mining, 2019.
[8] 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: Proc. 2018 International conference on learning representations, 2018.
[9] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, S.-K. Ng, Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks, in: Proc. 2019 International conference on artificial neural networks, 2019.
[10] J. Audibert, P. Michiardi, F. Guyard, S. Marti, M.A. Zuluaga, Usad: Unsupervised anomaly detection on multivariate time series, in: Proc. 2020 ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020.
[11] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proc. of the 30th International Conference on Neural Information Processing Systems, 2016.
[12] 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: Proc. 2020 IEEE International Conference on Data Mining (ICDM), 2020.
[13] A. Deng, B. Hooi, Graph Neural Network-Based Anomaly Detection in Multivariate Time Series, in: Proc. 2021 AAAI conference on artificial intelligence, 2021.
[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. Advances in neural information processing systems 30, 2017.
[15] S. Tuli, G. Casale, N.R. Jennings, Tranad: Deep transformer networks for anomaly detection in multivariate time series data. <https://arxiv.org/abs/2201.07284> , 2022.
[16] J. Xu, H. Wu, J. Wang, M. Long, Anomaly transformer: Time series anomaly detection with association discrepancy. <https://arxiv.org/abs/2110.02642> , 2021.
[17] A.P. Mathur, N.O. Tippenhauer, Swat: A water treatment testbed for research and training on ics security, in: Proc. 2016 international workshop on cyber-physical systems for smart water networks (CySWater), 2016.
[18] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, in: Proc. 2018 ACM SIGKDD international conference on knowledge discovery & data mining, 2018.
[19] A. Siffer, P.-A. Fouque, A. Termier, C. Largouet, Anomaly detection in streams with extreme value theory, in: Proc. 2017 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.
[20] T. Nakamura, M. Imamura, R. Mercer, E. Keogh, Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives, in: Proc. 2020 IEEE international conference on data mining (ICDM), 2020.
[21] C.M. Ahmed, V.R. Palleti, A.P. Mathur, Wadi: a water distribution testbed for research in the design of secure cyber physical systems, in: Proc. 2017 international workshop on cyber-physical systems for smart water networks, 2017.
[22] S. Nedelkoski, J. Bogatinovski, A.K. Mandapati, S. Becker, J. Cardoso, O. Kao, Multi-source distributed system data for ai-powered analytics, in: Proc. 2020 European Conference on Service-Oriented and Cloud Computing, 2020.
[23] Y. Zhang, Y. Chen, J. Wang, Z. Pan, Unsupervised deep anomaly detection for multi-sensor time-series signals, IEEE Transactions on Knowledge and Data Engineering 35(2)(2023) 2118-2132.