DeAnomaly: Anomaly Detection for Multivariate Time Series Using Robust Decomposition and Memory-Augmented Diffusion Models

Hui Dou, Pengcheng Shi, Yiwen Zhang, Pengfei Chen, and Zibin Zheng, *Fellow, IEEE*

***Abstract*—Multivariate time series anomaly detection (MTS-AD) is of great significance in various modern industrial applications and IT systems. Recently, some unsupervised deep models have been developed for MTS-AD. However, these methods often struggle to handle the complex temporal patterns and inevitable noise in MTS data, resulting in limited performance. To overcome these challenges, we propose DeAnomaly, a novel anomaly detection framework based on time series decomposition. Specifically, DeAnomaly employs a two-phase training paradigm, consisting of structural pattern elimination and anomaly detection on remainders. The structural pattern elimination phase learns normal trend and seasonal components through spatial relationship modeling and time-frequency analysis, which are subsequently removed from the original time series to overcome the limitation of complex temporal patterns. The anomaly detection phase utilizes the robust characteristics of noise with denoising diffusion models to identify and distinguish between noise and actual anomalies. Since anomalies and small random fluctuations are mainly retained in the remainders, anomalies will be more clearly exposed. In this way, DeAnomaly can detect anomalies more accurately and robustly. We conduct extensive experiments on four real-world datasets and 13 baselines, experimental results demonstrate that DeAnomaly outperforms these state-of-the-arts.**

***Index Terms*—Denoising diffusion model, multivariate time series (MTS), time series decomposition, unsupervised anomaly detection.**

I. Introduction

IN TODAY'S data-driven world, time series data is critical in many fields, including industrial manufacturing, IT systems, healthcare, and the Internet of Things (IoT). Time series anomaly detection aims to identify observations that deviate significantly from normal patterns [1]. Effectively analyzing and detecting anomalies in these time series data is essential for maintaining the normal operation of systems and preventing potential risks. With advancements in data collection and storage technologies, an increasing number of systems are capable of generating and accumulating multivariate time series (MTS) data, which contains dynamic changes in multiple dimensions of the systems [2]. Taking the IT systems for instance, as the scale of data centers expands and system architectures become more complex, improving the observability from different perspectives is necessary for anomaly detection and thus generating diverse time series datasets. Although traditional machine learning methods such as K-means [3], one-class SVM [4], and KNN [5] have been recently employed for time series anomaly detection with some success over manual analysis, these methods become less effective for MTS anomaly detection (MTS-AD) considering the dimensionality and complexity of datasets.

With the continuous advancement of deep learning technology, methods based on deep neural networks have shown tremendous potential in MTS-AD [6] due to their powerful temporal and spatial feature extraction capabilities. In practical application scenarios, anomalous events are often rare to appear, and labeling these anomalies entails significant time and labor costs. Consequently, many efforts have been spent on leveraging unsupervised deep learning methods [7] in recent years. Specifically, prediction-based methods [8], [9], [10] utilizes historical data to forecast future observations and detects anomalies by comparing these predictions with actual values. This approach has achieved certain results; however, it is highly sensitive to noise and struggles with handling nonstationary data. As a result, reconstruction-based methods [11], [12], [13] have become increasingly popular due to their robustness and strong generalization ability. These methods construct a model that learns the normal patterns from time series data, uses the model to reconstruct the test data, and then detects anomalies based on the reconstruction error.

Although these reconstruction-based models perform well, they employ a single network to directly model high-dimensional time series without considering underlying complex temporal patterns. A single network may struggle to adapt to multiple patterns, potentially masking subtle anomalies during detection and thereby compromising model performance. Therefore, it is crucial to unravel these complex temporal patterns and then perform targeted representation learning. Although decomposition can help better extract time series features and improve the interpretability of the model,

Received 17 January 2025; revised 31 March 2025; accepted 24 April 2025. Date of publication 15 May 2025; date of current version 30 May 2025. This work was supported by the National Natural Science Foundation of China under Grant 61902440 and Grant 62272001. The Associate Editor coordinating the review process was Dr. Zhenghua Chen. *(Corresponding author: Yiwen Zhang.)*

Hui Dou, Pengcheng Shi, and Yiwen Zhang are with the School of Computer Science and Technology, Anhui University, Hefei 230601, China (e-mail: douhui@ahu.edu.cn; shipengcheng@stu.ahu.edu.cn; zhangyiwen@ahu.edu.cn).

Pengfei Chen is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China (e-mail: chenpf7@mail.sysu.edu.cn).

Zibin Zheng is with the School of Computer Science and Engineering, Sun Yat-sen University, Zhuhai 519082, China (e-mail: zhzibin@mail.sysu.edu.cn).

Digital Object Identifier 10.1109/TIM.2025.3570337

however, existing decomposition-based reconstruction methods [14], [15] ignore the fact that the raw time series inevitably suffer from anomalies, resulting in inaccurate decomposition of patterns. Consequently, how to extract meaningful features from the MTS containing complex temporal patterns in a reconstruction framework is the first challenge. In addition, effectively dealing with noise and enhancing the model's robustness and anti-interference capability in noisy environments is another significant challenge. In practice, time series data are inevitably affected by noise, including regular numerical noise, partial data loss, or changes in data dimensions. These noises may come from sensor failures, errors during data collection, or external environmental factors. In this article, changes in data dimensions are out of our concern. The presence of noise can not only obscure true anomaly signals but also conceal real patterns within the data, reducing the accuracy and stability of anomaly detection.

To address the aforementioned challenges, we propose a novel decomposition-based unsupervised anomaly detector called DeAnomaly. This detector employs a two-phase detection framework. By decomposing the time series, it not only overcomes the limitations of existing reconstruction-based methods in handling complex temporal patterns but also further refines the noise-handling process. In the first phase, we remove the structural patterns (specifically, the trend and seasonality) decomposed from the raw time series. To solve the problem of inaccurate decomposition, we construct a model to revise time series decomposition called ReTSDec and perform pattern-specific feature extraction inside the model. This model first combines GNN with a novel Gumbel-Softmax graph structure learning technique for spatial relationship modeling. Since the frequency domain is more sensitive to seasonality, we then use time–frequency cross-fusion modeling to better represent the time series. Eventually, the normal structural patterns reconstructed by model ReTSDec are removed from the original time series to obtain remainders containing only anomalies and random noise. Through this process, complex temporal patterns in the time series can be unraveled while extracting remainders that facilitate anomaly detection. In the second phase, based on the remainders obtained in the first phase, in order to further distinguish between actual anomalies and noise, we construct an anomaly detection model based on diffusion models named ADDiff. Through the training process of noise addition and denoising, ADDiff is naturally robust to noise. In addition, we design a memory block to be used as the denoising neural network, which can effectively avoid misjudging noise as anomalies by storing normal noise prototypes in the memory unit.

The contributions of this study are summarized as follows.

1) We propose DeAnomaly, a unique two-phase MTS-AD method designed to unravel complex temporal patterns in time series through decomposition, thereby making anomalies more conspicuous. At the same time, we effectively address the challenges posed by the accuracy of the decomposition.
2) The proposed DeAnomaly employs graph-based spatial relationship modeling and time–frequency cross-fusion modeling to extract accurate structural patterns after time series decomposition. It then utilizes a diffusion model to process the remainders, effectively distinguishing between true anomalies and noise.
3) Experiments conducted on four real-world datasets and 13 baselines demonstrate that DeAnomaly achieves state-of-the-art results. Comprehensive ablation studies confirm the effectiveness of the key components in our model.

The rest of the article is structured as follows. We review related work on time series anomaly detection in Section II. Section III provides relevant preliminaries in which the problem statement is given. In Section IV, we will introduce the proposed DeAnomaly in detail. Experiments will be conducted in Section V to prove the effectiveness of DeAnomaly, and conclusions will be drawn in Section VI.

II. Related Work

In this section, we outline various anomaly detection methods pertinent to our proposed model. We start by discussing standard techniques used in detecting anomalies in the MTS. Following this, we delve into more specific approaches that leverage time series decomposition and diffusion models.

A. MTS Anomaly Detection

In recent years, the field of MTS-AD has seen significant development. Researchers have mainly focused on using deep learning techniques to capture and model long-term dependencies and complex intervariable relationships in the MTS for more accurate and efficient anomaly detection. Notably, TranAD [12] enhances the accuracy and stability of the Transformer in anomaly detection through self-conditioning and adversarial training. ATF-UAD [16] accurately detects and locates anomalies in data through a dual-view adversarial mechanism involving a time reconstructor and a frequency reconstructor. In variable dependency modeling, GDN [9] is a graph neural network method based on the attention mechanism. It learns the dependency graph among sensors, detecting and interpreting deviations in these relationships for anomaly detection. These studies primarily model raw time series data using various deep learning techniques, employing reconstruction-based or prediction-based methods to identify anomalies. However, they overlook the complex temporal patterns contained in the time series, which adversely affects the modeling effectiveness and leads to a certain degree of false positives.

B. Decomposition for Time Series Anomaly Detection

Time series decomposition has recently been used for time series anomaly detection. DecompTran [14] combines signal decomposition structure and a Transformer to explicitly learn complex temporal patterns for anomaly detection. TFAD [15] utilizes both time-domain and frequency-domain information on each component after time series decomposition to improve detection performance. TADNet [17] employs seasonal-trend decomposition to link different types of anomalies with specific decomposition components, while achieving end-to-end decomposition and detection through pretraining and fine-tuning. Although all these methods can unravel complex temporal patterns in time series, improving model interpretability and detection performance, they do not account for the inaccuracies in decomposition that occur when the original time series are contaminated. In detail, the study RobustSTL [18] has shown that when high noise, abrupt changes in trends, and seasonal drifts and fluctuations occur in a series, classical decomposition methods cannot accurately extract the trend and seasonality components. Furthermore, the performance comparison and visualization experiments in Section V-D also demonstrate this phenomenon.

C. Diffusion Models for Time Series Anomaly Detection

Diffusion models [19], [20] have achieved remarkable results in various tasks. In recent years, researchers have increasingly explored their applications in time series anomaly detection. DiffusionAE [21] uses a diffusion model on the reconstruction of the autoencoder, effectively smoothing out anomalies and enhancing robustness to small noise levels. DDMT [22] integrates diffusion models with the Transformer while designing a dynamic masking mechanism to avoid excessive reconstruction of anomalies. ImDiffusion [23] combines time series imputation with diffusion models, utilizing neighboring values and step-by-step denoising outputs to detect anomalies in MTS data. These methods effectively improve the accuracy and robustness of anomaly detection by taking advantage of the superior unsupervised modeling capability of diffusion models and their ability to deal with noise. Unfortunately, the noise in the original series is often intertwined with other complex temporal patterns, which undermines the diffusion model's ability to handle noise and increases the likelihood of misclassifying noise as anomalies or missing actual anomalies. From the perspective of signal-to-noise ratio (SNR), in anomaly detection tasks, SNR is defined as the ratio of anomaly power to noise power. Noise power includes all components that interfere with anomaly detection. Since noise is intertwined with other patterns, the SNR is very low in this context, and some anomalies and noise have highly similar statistical distributions, increasing the probability of misclassification. Furthermore, diffusion models need to simultaneously suppress noise and fit structural patterns during denoising. However, these patterns may overlap in both the frequency and time domains, making it challenging to design a universal noise scheduling method within diffusion models. This weakens the noise-handling capability of diffusion models and can lead to either excessive smoothing or insufficient smoothing of signals during denoising, thereby distorting anomaly signals or amplifying noise signals, and ultimately reducing the accuracy of detection. Consequently, these methods still face challenges in adapting to noisy environments.

III. Preliminaries

A. Problem Formulation

For an MTS-AD task, consider a time series $\mathbf{X} = \{x_0, x_1, \ldots x_l, \ldots x_{L-1}\} \in \mathbb{R}^{L\times M}$ of length L, $x_l \in \mathbb{R}^M$ is an observation at timestamp l which includes M feature dimensions. Then, given a training data $\mathbf{X} \in \mathbb{R}^{L\times M}$ and a testing data $\mathbf{X}' \in \mathbb{R}^{L'\times M}$, our primary objective is to identify whether an unseen observation $x_{l'}$ from $\mathbf{X}'$ is an anomaly. We need to predict an output $\mathbf{Y} = \{y_0, y_1, \ldots y_{l'}, \ldots y_{L'-1}\}$, where $y_{l'} \in \{0, 1\}$. $y_{l'} = 1$ indicates that $x_{l'}$ is anomalous.

B. Time Series Decomposition

As we know, time series often exhibit a variety of different patterns. Therefore, it is helpful to decompose a time series ($\mathbf{X}$) into the following components: trend ($\mathbf{T}_X$), seasonality ($\mathbf{S}_X$), and remainder ($\mathbf{R}_X$). The decomposition can be described as: $\mathbf{X} = \mathbf{T}_X + \mathbf{S}_X + \mathbf{R}_X$. There are various methods to decompose time series into the above three components currently, such as moving averages, SEAT [24], and STL [25]. Each component represents a potential pattern. The trend component reflects long-term progress in the data. The seasonal component captures the periodic fluctuations, which are often related to the season, month, week, and time of day. The remainder component only contains unpredictable random fluctuations, that is, noise [26]. The core basis for considering the remainder as noise is whether its statistical characteristics satisfy the white noise assumption. [27] verifies this assumption through ACF/PACF, the box-pierce test, and the ljung-box test. Therefore, we can treat the remainder component as noise.

IV. Methodology

A. Overview

The overall architecture of DeAnomaly is shown in Fig. 1. The decomposition block first separates complex temporal patterns of the input series. For each variable in the MTS, we perform an independent decomposition. The multivariate nature will be taken into account in subsequent model processing. Next, DeAnomaly will be trained in two phases. In the first phase, in order to obtain the remainders that actually represent anomalies, we add the trend and seasonal components and feed them into the model ReTSDec to obtain the normal structural patterns. Then, we subtract them from the original series and get the revised remainders. In the second phase, we input these remainders into the model ADDiff for anomaly detection. In Sections IV-B–IV-E, we will introduce each part of our proposed DeAnomaly in detail.

B. Time Series Decomposition

To unravel the complex temporal patterns in the time series, we employ the idea of decomposition, which can separate each sequence $\mathbf{X}$ in the MTS into three parts: trend $\mathbf{T}$, seasonality $\mathbf{S}$, and remainder $\mathbf{R}$. Considering both performance and efficiency, we adopt a decomposition method based on a moving average. It is worth noting that with the ability to revise the structural patterns via the well-designed model ReTSDec, DeAnomaly is not limited to using a specific decomposition method. Detailed comparison among different decomposition methods will be provided in Section V-D. In order to extract the trend component from $\mathbf{X}$, we utilize multiple distinct kernels and integrate these extracted different patterns through

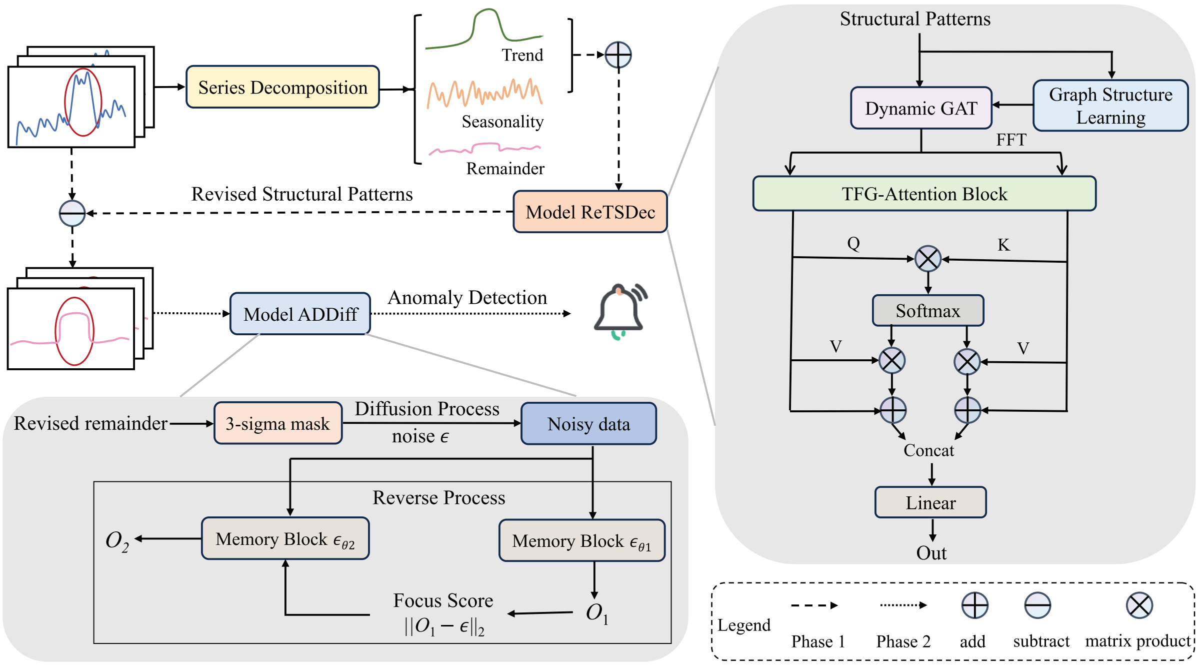

Fig. 1. Overview architecture of DeAnomaly. The workflow begins with a series decomposition block that splits the input series into three components: trend, seasonality, and remainder. The training process of DeAnomaly occurs in two stages: initially, the model ReTSDec is trained to capture the normal patterns of the trend and seasonal components. Subsequently, the revised remainder components are used to train the model ADDiff for anomaly detection. The detailed architecture of the model ReTSDec is depicted on the right panel, while the model ADDiff's architecture is illustrated on the lower panel.

a simple averaging operation. For the input series $\mathbf{X} \in \mathbb{R}^{L\times M}$, the process is

$$\mathbf{T} = \frac{1}{n}\sum_{i=1}^{n} \text{AvgPool}\left(\text{Padding}\left(\mathbf{X}\right)\right)_{\text{kernel}_i}. \tag{1}$$

Now, we can obtain the detrended series $\mathbf{X}^{\text{detrend}} = \mathbf{XT}$. Next, following the practice in [28], we extract the seasonal component from $\mathbf{X}^{\text{detrend}}$. First, seasonal segments are obtained by averaging $\mathbf{X}^{\text{detrend}}$ over a predefined period p. Then, the obtained segments are tiled to construct the overall seasonal component $\mathbf{S}$ of length L. Each element of the segments is calculated as follows:

$$\mathbf{S}_i = \frac{1}{n}\sum_{j=0}^{n-1} \mathbf{X}_{i+jp}^{\text{detrend}} \tag{2}$$

where n is the smallest integer satisfying $L < i + np$ for $0 \leq i < p$, $\mathbf{S}_i$ and $\mathbf{X}_{i+jp}^{\text{detrend}}$ denote the ith element of the seasonal segments and the $(i + jp)$th element of $\mathbf{X}^{\text{detrend}}$. After removing trend and seasonality from the series, we get the remainder component $\mathbf{R}$. It is worth noting that this remainder is not directly used for final anomaly detection considering its doubtful accuracy in the complex MTS scenarios.

C. Model ReTSDec

After obtaining the decomposed components, we use a model, ReTSDec, to capture the accurate structural patterns in the MTS. This model primarily consists of four parts, including graph structure learning, graph attention mechanism, time–frequency interaction layer, and time–frequency feature fusion layer.

1) Graph Structure Learning:

MTS is often continuously collected by numerous sensors, and there are usually intricate correlations between these collected variables. Therefore, establishing spatial relationships between multiple variables can more effectively model the MTS. In this article, we use graph structure to model spatial relationships. We consider each variable as a node and the relationships between variables as edges in the graph. Typically, we have no prior knowledge about adjacent correlations between multiple variables. In this case, we use graph structure learning to identify complex relationships between nodes. Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the node set, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the edge set. Note $e_{i,j}$ represents the uni-directed edge from node i to node j. The neighborhood of node i is defined as $\mathcal{N}(i) = \{j \in \mathcal{V} | e_{i,j} \in \mathcal{E}\}$.

Many existing methods [9], [29] focus on measuring the distance or similarity between nodes and selecting the top-K closest nodes for their neighbor dependencies. This will result in high computational complexity of $\mathcal{O}(N^2)$, while the learned node dependencies are symmetric, which is usually inconsistent with real-world situations [30]. In order to avoid the limitations caused by selecting top-K features, we adopt the Gumbel-Softmax sampling method, which has been shown to be effective for graph structure learning in [31]. It is defined as

$$A_{ij} = \text{Softmax}\left(\left(\beta_{ij} + g_{ij}\right)/\tau\right) \tag{3}$$

where A_{ij} is the connection strategy for any pair of nodes i and j, β_{ij} denotes the pairwise dependence parameter of nodes i and j, g is an independent and identically distributed sample drawn from the standard Gumbel distribution, and τ is the temperature parameter. In addition, the initial graph structure is a fully connected graph to facilitate better learning. Ultimately, the graph structure learning module can automatically learn the graph topology that represents the connections between all nodes.

2) Graph Attention Mechanism: In order to further capture the potential interrelationships between variables, we integrate the information of nodes with their neighbors based on GAT and the learned graph structure. For the input series $\mathbf{X}_{\text{in}} \in \mathbb{R}^{L\times M}$, the updated representation x_i' of node i is as follows:

$$x_i' = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}x_j\right) \tag{4}$$

where σ denotes a nonlinear activation function, $\mathcal{N}_i$ represents the neighbor set of node i, α_{ij} denotes the attention coefficient which indicates the importance of node j to node i, and $\mathbf{W}$ is a trainable weight matrix. The attention coefficient α_{ij} can be calculated by

$$\pi_{ij} = \text{LeakyReLU}\left(\mathbf{a}^{\mathbf{T}} \cdot \left[\mathbf{W}x_i \| \mathbf{W}x_j\right]\right) \tag{5}$$

$$\alpha_{ij} = \frac{\exp(\pi_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(\pi_{ik})} \tag{6}$$

where π_{ij} denotes the attention coefficient before softmax normalization, $\|$ represents concatenation operation, and $\mathbf{a}$ denotes the parameter vector of the attention mechanism.

GAT can break the limitation of GCN in updating node information. However, the standard GAT is static attention, which may make the obtained attention coefficient relatively unchanged and reduce the model fitting ability [32]. Therefore, in this article, we implement dynamic attention by altering the operation sequence in the standard GAT. Concretely, dynamic attention is calculated as follows:

$$\pi_{ij} = \mathbf{a}^{\mathbf{T}}\text{LeakyReLU}\left(\mathbf{W} \cdot \left[x_i \| x_j\right]\right). \tag{7}$$

By applying $\mathbf{W}$ after concatenation and swapping the positions of the activation function and the attention layer in the standard GAT, we can effectively compute the score for each query–key pair, addressing the limitations of standard GAT.

3) Time–Frequency Interaction Layer: After modeling the spatial relationships of the series, we model the temporal relationships simultaneously in the time domain and frequency domain. Time domain analysis can capture temporal dynamics well, and frequency domain analysis can adequately capture periodic changes. Combining the above two domains can effectively avoid erroneous results obtained by single-domain analysis. Let $\mathbf{X}_T$ generated by spatial modeling be the input of the time-domain branch. We then convert the time series from the time domain to the frequency domain via the Fourier transform. The data in the frequency domain consists of real-part (denoted as re) and imaginary-part (denoted as im). To facilitate the generalization ability of the model, we intersects re and im and get the input $\mathbf{X}_F = \{\text{re}_1, \text{im}_1, \text{re}_2, \text{im}_2, \ldots \text{re}_L, \text{im}_L\}$ of the frequency domain branch where L is series length. However, separate feature extraction in the time domain and frequency domain will lead to information isolation and limit the model's performance. The interaction of time–frequency domain information can promote the respective feature extraction capabilities. Therefore, we design a time–frequency guidance attention (TFG-Attention) block to extract time feature $\mathbf{X}_T'$ and frequency feature $\mathbf{X}_F'$ from the input.

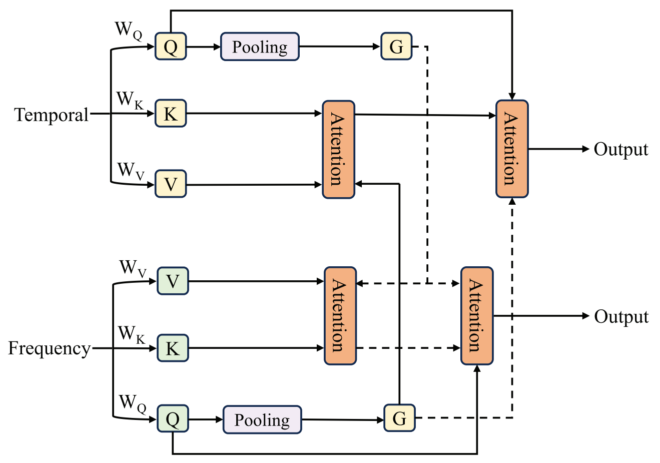

Fig. 2. TFG-Attention block.

As shown in Fig. 2, for an input $\mathbf{X} \in \mathbb{R}^{L\times d}$ from the time or frequency domain, where L is the input length and d is the input dimension, we first projects $\mathbf{X}$ to query $\mathbf{Q} \in \mathbb{R}^{L\times d_q}$, key $\mathbf{K} \in \mathbb{R}^{L\times d_k}$, and value $\mathbf{V} \in \mathbb{R}^{L\times d_v}$ through linear projections. Then, we introduce a guidance matrix from the exotic domain. The guidance matrix $\mathbf{G} = \text{Pooling}(\mathbf{Q}) \in \mathbb{R}^{l\times d_q}$, where l is much smaller than L. Finally, we can compute the guidance attention GuiAtt as follows:

$$\begin{aligned}\text{Att}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathbf{T}}}{\sqrt{d_k}}\right)\mathbf{V}\\ \text{GuiAtt}^T &= \text{MultiAtt}\left(\mathbf{Q}_T, \mathbf{G}_F, \text{MultiAtt}\left(\mathbf{G}_F, \mathbf{K}_T, \mathbf{V}_T\right)\right)\\ \text{GuiAtt}^F &= \text{MultiAtt}\left(\mathbf{Q}_F, \mathbf{G}_T, \text{MultiAtt}\left(\mathbf{G}_T, \mathbf{K}_F, \mathbf{V}_F\right)\right)\end{aligned} \tag{8}$$

where $\mathbf{G}_F$ denotes the frequency-to-time guidance matrix, $\mathbf{G}_T$ represents the time-to-frequency guidance matrix, and multihead attention MultiAtt$(\mathbf{Q},\mathbf{K},\mathbf{V})$ employs several distinct learned projection sets instead of a single one. Compressed time-domain features promote greater sensitivity to seasonal variations in the frequency domain, and compressed frequency-domain features guide the model to better focus on the dynamics of the time domain. At the same time, this quadruple attention calculation achieves linear complexity, and the global context modeling capability is maintained [33].

4) Time–Frequency Feature Fusion Layer: In order to effectively integrate features from different domains, we propose an attention-based feature fusion layer to learn a weighted combination of time and frequency domain features. The time feature $\mathbf{X}_T' \in \mathbb{R}^{L\times d}$ is first transformed to the matrix $\mathbf{Q}_{\text{temp}} \in \mathbb{R}^{L\times d}$ and $\mathbf{V}_{\text{temp}} \in \mathbb{R}^{L\times d}$ through linear projections. Similarly, the frequency feature $\mathbf{X}_F' \in \mathbb{R}^{L\times d}$ is also transformed to the matrix $\mathbf{K}_{\text{freq}} \in \mathbb{R}^{L\times d}$ and $\mathbf{V}_{\text{freq}} \in \mathbb{R}^{L\times d}$ through linear projections. Then, the time–frequency weight distribution value $\mathbf{O}_{tf}$ can be obtained by

$$\mathbf{O}_{tf} = \text{Softmax}\left(\frac{\mathbf{Q}_{\text{temp}}\mathbf{K}_{\text{freq}}^{\mathbf{T}}}{\sqrt{d}}\right). \tag{9}$$

We fuse time and frequency features through the following formula:

$$\overline{\mathbf{X}}'_T = \mathbf{O}_{tf} \times \mathbf{V}_{\text{temp}} + \mathbf{X}'_T \tag{10}$$

$$\overline{\mathbf{X}}'_F = \mathbf{O}_{tf} \times \mathbf{V}_{\text{freq}} + \mathbf{X}'_F \tag{11}$$

$$\mathbf{X}'_{TF} = \text{Concat}\left(\overline{\mathbf{X}}'_T, \overline{\mathbf{X}}'_F\right) \tag{12}$$

where $\overline{\mathbf{X}}'_T$ and $\overline{\mathbf{X}}'_F$ are the transformed time and frequency features, respectively, the Concat(·) is the concatenation function, and $\mathbf{X}'_{TF}$ is the time–frequency fusion feature. Skip connections are used to avoid information coverage between domains. The final reconstruction $\mathbf{X}_{\text{out}} \in \mathbb{R}^{L \times M}$ of model ReTSDec is obtained through a single linear layer. The training objective L_{dec} is defined as follows:

$$L_{\text{dec}} = \|\mathbf{X}_{\text{in}} - \mathbf{X}_{\text{out}}\|_2. \tag{13}$$

D. Model ADDiff

To accurately detect anomalies on the remainders, we design a detection model, which includes two main parts: a masking strategy and a memory-based diffusion model.

1) Masking Strategy: To improve the model's ability to understand temporal context information and better distinguish between random fluctuations and subtle anomalies, we mask significant anomalies of the revised remainder series $\mathbf{X}_{\text{re}} \in \mathbb{R}^{L \times M}$ input to the model ADDiff. In detail, we use the 3-sigma rule [34] to detect obvious anomalies and create a binary mask matrix $\mathbf{M} \in \mathbb{R}^{L \times M}$ based on this, the masked series $\mathbf{X}_0 \in \mathbb{R}^{L \times M}$ is obtained by Hadamard product of $\mathbf{M}$ with $\mathbf{X}_{\text{re}}$. This masking behavior has the effect of reducing the percentage of anomalies, which is beneficial to downstream modules.

2) Memory-Based Diffusion Model: This module is based on diffusion models, a class of generative models inspired by nonequilibrium thermodynamics [19]. Diffusion models consist of a diffusion process and a reverse process. During the diffusion process, we incrementally add Gaussian noise to the initial input sample $\mathbf{X}_0$ over T steps. Mathematically, this can be described as follows:

$$q(\mathbf{X}_{1:T}|\mathbf{X}_0) = \prod_{t=1}^{T} q(\mathbf{X}_t|\mathbf{X}_{t-1})$$
$$q(\mathbf{X}_t|\mathbf{X}_{t-1}) = \mathcal{N}\left(\mathbf{X}_t;\ \sqrt{1-\beta_t}\mathbf{X}_{t-1}, \beta_t \mathbf{I}\right) \tag{14}$$

where $\beta_t \in (0, 1)$ can be defined as a linear sequence increasing with t to control the rate of added noise. $\mathbf{X}_t$ is noised by $\mathbf{X}_t = \sqrt{\overline{\alpha}_t}\mathbf{X}_0 + (1 - \overline{\alpha}_t)\epsilon$, where $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, $\alpha_t = 1 - \beta_t$, and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is the sampled noise. We add noise to both the masked and unmasked parts of $\mathbf{X}_0$.

The reverse process of a Markov chain, defined by the parameter θ, can be formulated as

$$p_\theta(\mathbf{X}_{0:T-1}|\mathbf{X}_T) = p(\mathbf{X}_T) \prod_{t=1}^{T} p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t)$$
$$p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t) = \mathcal{N}\left(\mathbf{X}_{t-1}; \mu_\theta(\mathbf{X}_t, t, c), \tilde{\beta}_t \mathbf{I}\right) \tag{15}$$

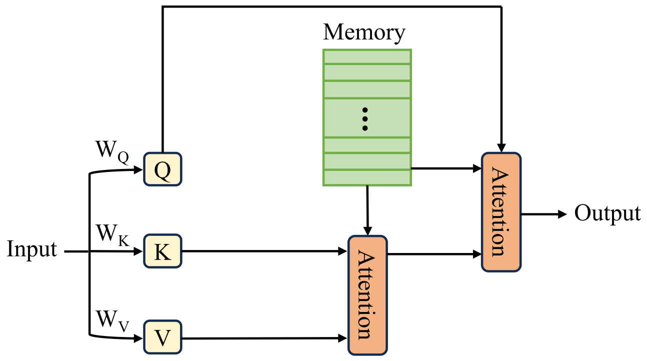

Fig. 3. Memory block.

where $\tilde{\beta}_t = (1 - \overline{\alpha}_{t-1})/(1 - \overline{\alpha}_t)\beta_t$ and $\tilde{\beta}_1 = \beta_1$. μ_θ is parameterized as

$$\mu_\theta(\mathbf{X}_t, t, c) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{X}_t \frac{\beta_t}{\sqrt{1 - \overline{\alpha}_t}} \epsilon_\theta(\mathbf{X}_t, t, c)\right) \tag{16}$$

where c represents conditional information, which is concatenated by the mask matrix $\mathbf{M}$ and the focus score S. ϵ_θ is the denoising neural network used to predict the noise from $\mathbf{X}_t$. We build ϵ_θ based on a dedicated memory block to better adapt to anomaly detection on the revised remainders.

The proposed memory block is shown in Fig. 3. Similar to the TFG-Attention block, the input is mapped to $\mathbf{Q} \in \mathbb{R}^{L \times d_q}$, $\mathbf{K} \in \mathbb{R}^{L \times d_k}$, and $\mathbf{V} \in \mathbb{R}^{L \times d_v}$ through linear projections. Define a memory matrix $\mathbf{J} \in \mathbb{R}^{n \times d_q}$, which stores n memory items with dimension d_q, and n is much smaller than L. Note that since we use the masking strategy and the memory block, performing a full attention computation may include many unnecessary or invalid information interactions. To address these limitations, we keep only the top-K scores with the highest weights. K is a variable parameter that dynamically controls the sparsity level, calculated as the weighted average of specific proportions like 0.75. Then, we can calculate the output O of the memory block as follows:

$$\text{Top}k\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\text{top}k\left(\frac{\mathbf{Q}\mathbf{K}^{\mathbf{T}}}{\sqrt{d_k}}\right)\right)\mathbf{V}$$
$$O = \text{Top}k\text{Att}\left(\mathbf{Q}, \mathbf{J}, \text{Top}k\text{Att}(\mathbf{J}, \mathbf{K}, \mathbf{V})\right) \tag{17}$$

where top$k$(·) is the learnable top-K selection operator.

We customize a two-step training approach based on the memory block. In the first step, memory block $\epsilon_{\theta 1}$ generate an output $O_1 = \epsilon_{\theta 1}(\mathbf{X}_t, t, c)$ to predict the noise ϵ, where the focus score S contained in c is a zero vector. In the second step, we update the focus score $S = \|O_1 \epsilon\|_2$ with the reconstruction loss of $\epsilon_{\theta 1}$ and get the output $O_2 = \epsilon_{\theta 2}(\mathbf{X}_t, t, c)$ of memory block $\epsilon_{\theta 2}$. In this way, the model pays more attention to where the reconstruction loss is higher, thus improving the sensitivity to anomalies. The model is trained to predict the noise of the entire corrupted series

$$L_{\text{dif}} = \gamma^{-m}\|O_1 - \epsilon\|_2 + \left(1 - \gamma^{-m}\right)\|O_2 - \epsilon\|_2 \tag{18}$$

where γ is a training parameter close to one and m represents the training epoch. As the number of training epochs increases, the focus score becomes gradually more reliable, and the weight of the loss in the second step also progressively increases. At the same time, the memory matrix can be updated through the loss function, forcing it to record normal features. Since the normal fluctuations of the remainders are centered around 0, the memory matrix is randomly initialized within a small range around 0.

E. Training and Inference

Algorithm 1 Training Process

```
Input: Training data $\mathbf{X} = \{x_1, x_2, \dots x_L\}$, training epoch $m_1, m_2$, number of diffusion steps $T$, hyperparameter $\gamma$.
Output: The trained model $ReTSDec$ and $ADDiff$.
1:  Initialize weights of the model $ReTSDec$ and $ADDiff$;
2:  Train the model $ReTSDec$.
3:  for $m = 1$ to $m_1$ do
4:      for mini-batch in $\mathbf{X}$ do
5:          Decompose $\mathbf{X}$ into $\mathbf{T}_X, \mathbf{S}_X, \mathbf{R}_X$;
6:          $\mathbf{X}_{in} \leftarrow \mathbf{T}_X + \mathbf{S}_X$;
7:          $\mathbf{X}_{out} \leftarrow ReTSDec(\mathbf{X}_{in})$;
8:          $L_{dec} \leftarrow \|\mathbf{X}_{in}\mathbf{X}_{out}\|_2$;
9:          $ReTSDec \leftarrow$ update weights using $L_{dec}$;
10:         $\mathbf{X}_{re} \leftarrow \mathbf{X}ReTSDec(\mathbf{T}_X + \mathbf{S}_X)$;
11:     end for
12: end for
13: Train the model $ADDiff$.
14: for $m = 1$ to $m_2$ do
15:     for mini-batch in $\mathbf{X}_{re}$ do
16:         $\mathbf{X}_0 \leftarrow \text{Mask}(\mathbf{X}_{re})$;
17:         Sample $t \sim \text{Uniform}(1, \dots, T)$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$;
18:         $\mathbf{X}_t \leftarrow \sqrt{\overline{\alpha}_t}\mathbf{X}_0 + (1 - \overline{\alpha}_t)\epsilon$;
19:         $S \leftarrow \vec{0}$, $O_1 \leftarrow \epsilon_{\theta 1}(\mathbf{X}_t, t, c)$;
20:         $S \leftarrow \|O_1\epsilon\|_2$, $O_2 \leftarrow \epsilon_{\theta 2}(\mathbf{X}_t, t, c)$;
21:         $L_{dif} \leftarrow \gamma^{-m_2}\|O_1 - \epsilon\|_2 + (1 - \gamma^{-m_2})\|O_2 - \epsilon\|_2$;
22:         $ADDiff \leftarrow$ update weights using $L_{dif}$;
23:     end for
24: end for
25: return: The optimized model $ReTSDec$ and $ADDiff$.
```

The training process consists of two phases. In the first phase, the model ReTSDec is trained aiming to provide a revised remainder for anomaly detection by minimizing the loss term L_{dec}. In the second phase, training the model ADDiff for anomaly detection involves minimizing the loss term L_{dif}. The training process of DeAnomaly is shown in Algorithm 1.

In the inference phase, we compute the anomaly score for timestamp t as follows:

$$\text{Score}(t) = \sum_{i=1}^{d} \|s_t^i \tilde{s}_t^i\|_2 \tag{19}$$

where $s_t = \hat{\mathbf{X}}_0^t$ and $\tilde{s}_t = \tilde{\mathbf{X}}_0^t$ are the observed value and predicted value, respectively, and d refers to the number of variables. If the anomaly score Score(t) exceeds a defined threshold, the corresponding timestamp t is identified as an anomaly. The threshold is established by the proportion of anomalies in the validation dataset. Finally, Algorithm 2 summarizes the inference process of DeAnomaly.

Algorithm 2 Inference Process

```
Input: Testing data $\hat{\mathbf{X}} = \{\hat{x}_1, \hat{x}_2, \dots \hat{x}_{L'}\}$, number of diffusion steps $T$, threshold $\lambda$.
Output: The predicted label list of $\mathbf{Y} = \{y_1, y_2, \dots y_{L'}\}$.
1:  for mini-batch in $\hat{\mathbf{X}}$ do
2:      Decompose $\hat{\mathbf{X}}$ into $\mathbf{T}_{\hat{X}}, \mathbf{S}_{\hat{X}}, \mathbf{R}_{\hat{X}}$;
3:      $\hat{\mathbf{X}}_{re} \leftarrow \hat{\mathbf{X}}ReTSDec(\mathbf{T}_{\hat{X}} + \mathbf{S}_{\hat{X}})$;
4:      $\hat{\mathbf{X}}_0 \leftarrow \hat{\mathbf{X}}_{re}$;
5:      Sample noise $\tilde{\mathbf{X}}_T \sim \mathcal{N}(0, \mathbf{I})$;
6:      for $t = T$ to 1 do
7:          $S \leftarrow \vec{0}$, $O_1 \leftarrow \epsilon_{\theta 1}(\tilde{\mathbf{X}}_t, t, c)$;
8:          $S \leftarrow \|O_1\tilde{\mathbf{X}}_t\|_2$, $O_2 \leftarrow \epsilon_{\theta 2}(\tilde{\mathbf{X}}_t, t, c)$;
9:          $z \sim \mathcal{N}(0, \mathbf{I})$ if $t > 1$ else $z = 0$;
10:         $\tilde{\mathbf{X}}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(\tilde{\mathbf{X}}_t \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}} \frac{O_1+O_2}{2}\right) + \tilde{\beta}_t z$;
11:     end for
12:     Calculate the anomaly score $\varphi = \|\hat{\mathbf{X}}_0\tilde{\mathbf{X}}_0\|_2$;
13:     for each element $\varphi_i$ of $\varphi$ do
14:         if $\varphi_i > \lambda$ then
15:             $y_i \leftarrow 1$; // anomaly
16:         else
17:             $y_i \leftarrow 0$; // normal
18:         end if
19:     end for
20: end for
21: return: The predicted label list $\mathbf{Y}$.
```

TABLE I
STATISTICS OF THE DATASETS

Dataset	Train	Test	Dimensions	Anomalies(%)
PSM	132481	87841	25	27.8
MSL	58317	73729	55	10.7
SMAP	135183	427617	25	13.1
SWaT	495000	449919	51	12.1

V. EXPERIMENTS

A. Datasets and Evaluation Metrics

To validate the effectiveness of the DeAnomaly method, we utilize four well-known public datasets: Pooled Server Metrics (PSM), Mars Science Laboratory (MSL), Soil Moisture Active Passive (SMAP), and Secure Water Treatment (SWaT). The dataset statistics are detailed in Table I. The PSM dataset [35] is sourced from multiple application server nodes at eBay, covering metrics such as CPU load and memory usage that are essential for evaluating server performance. The MSL rover and SMAP satellite datasets [36] are publicly available from NASA. MSL offers extensive data from the MSL's study of Martian geology and climate, with a total of 55 dimensions. SMAP contains satellite-derived data related to soil moisture and freeze-thaw conditions, featuring 25 dimensions. The SWaT dataset [37] was gathered from 51 sensors in a real-world water treatment facility, encompassing seven days of normal operations for training and four days of simulated attack scenarios for testing.

We use precision (P), recall (R), and $F1$-score ($F1$) as evaluation metrics to compare the performance between

DeAnomaly and other baselines. They are defined as follows:

$$P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \; F1 = \frac{2 \times P \times R}{P+R} \tag{20}$$

where TP denotes the true positives, FP represents the false positives, and FN is the false negatives. In practical applications, time series data may exhibit consecutive anomalous data points due to external interference or system errors. In response, this article adopts a widely used point-adjust strategy [38], which assumes that if any observation within a detected continuous anomalous segment is flagged as an anomaly, all anomalies within that segment are deemed to be accurately detected. Note that no adjustments are made to the detection results for point anomalies outside of the anomalous segments. We also adopt the R-AUC-PR evaluation metric proposed in [39], which is applicable to range-based anomaly detection. This evaluation method provides a new perspective for performance analysis of anomaly detection algorithms and effectively reduces the bias that may arise from threshold selection in traditional evaluation methods.

B. Experimental Setup

In our experiments, both the ReTSDec and ADDiff models are trained using a sliding window W set to 100 across all datasets. We use the SGD optimizer with an initial learning rate of 1e^{-4}, and we adopt a learning rate adjustment strategy. Each model undergoes training for 100 epochs, utilizing an early stopping strategy with a patience level of 5 epochs. The batch size for training is 128. For the model ReTSDec, we configure the number of GAT layers to be 1. The hidden layer dimension d_1 of the TFG-Attention block is 128, and the number of heads h is set to 4. The dimension d_2 of the guidance matrix is configured to 25. In the model ADDiff, we employ 150 diffusion steps for the diffusion model. The number of memory items n is set to 16, and the value of K is set to three proportions: 0.67, 0.75, 0.8. Our method is implemented using PyTorch [40] version 1.13.1 with CUDA 11.6. All experiments are conducted on a laptop equipped with 32 GB of memory, an Intel Core i7 CPU, and an NVIDIA GeForce RTX 3060 GPU. The source code is publicly available at https://github.com/bbuu000/DeAnomaly.

C. Overall Performance

1) Baseline Methods: To evaluate the performance of DeAnomaly, we compare it with 13 state-of-the-art models in MTS-AD, including the classic method IF [41] and deep learning-based methods. Deep learning-based methods can be further categorized into the prediction-based methods: MTAD-GAT [8], and GDN [9]; the reconstruction-based methods: BeatGAN [42], USAD [43], OmniAnomaly [44], InterFusion [11], TranAD [12], ATF-UAD [16], DTAAD [45], and CATCH [46]; the decomposition-based method DecompTran [14]; the diffusion-based method ImDiffusion [23].

2) Results and Analysis: We compare the overall performance of DeAnomaly with other baselines and report the average values of six individual runs in Table II, with the best $F1$ and R-AUC-PR results in bold and the second best underlined. As shown in the table, DeAnomaly demonstrates the best overall performance across all datasets, with the highest average $F1$-score of 0.9850 for PSM, 0.9321 for MSL, 0.9578 for SMAP, and 0.9673 for SWaT.

We observe that the classical method, IF, has the worst detection performance compared to other deep learning models. IF only attempts to separate anomaly points from other points for detection. Although it is simple and fast, it fails to effectively model the complex information of the MTS. Two prediction-based methods focus on analyzing and modeling the MTS from a spatial perspective. MTAD-GAT constructs a fully connected graph for all variables, while GDN learns the topological graph structure between variables through a top-K nearest connection strategy. These approaches have certain limitations and do not align with real-world scenarios. BeatGAN and USAD aim to better capture the complex distribution of normal data through adversarial learning, amplifying the reconstruction errors of anomalies. However, they overlook the temporal dynamics of MTS data and the correlations between variables. OmniAnomaly achieves relatively good results by utilizing GRU to model the temporal dependencies in sequences, but it neglects the potential relationships between variables. InterFusion, as an extension of it, takes into account both temporal and spatial dependencies, resulting in an improvement in detection outcomes. However, LSTM-based models have limitations in modeling long-term sequences because of the characteristics of their recurrent learning mechanisms. To overcome this limitation, the Transformer architecture has been widely used to extract time series features in recent years. TranAD amplifies the reconstruction errors of anomalies through adversarial training of two Transformer architectures. However, it only processes information in the time domain, which may result in the inability to handle certain complex anomalies. In contrast, ATF-UAD utilizes a dual-view adversarial learning mechanism in both the time domain and frequency domain, effectively enhancing its anomaly detection capabilities. Similarly, CATCH captures fine-grained frequency features through frequency domain chunking and utilizes a channel fusion module based on a two-layer optimization algorithm to achieve efficient anomaly detection in MTS. DTAAD uses causal convolution and dilated convolution as local TCN and global TCN, respectively, and combines them with a Transformer and feedback mechanism to improve detection accuracy. DecompTran designs a frequency attention module and integrates it with a Transformer and time series decomposition to explicitly learn complex temporal patterns. ImDiffusion integrates time series imputation with diffusion models to achieve precise modeling of spatio-temporal dependencies, thereby enhancing the robustness of anomaly detection.

However, the fundamental idea of the aforementioned methods is based on detecting anomalies in the original sequences, which are easily disturbed by complex temporal patterns within them. Although DecompTran unravels complex temporal patterns and explicitly models them separately, it does not consider the potential inaccuracies in decomposition when anomalies and noise are present in the original sequences. This leads to ineffective modeling of the individual components.

TABLE II
PERFORMANCE OF DEANOMALY AND BASELINES

Method	PSM				MSL			
	P	R	F1	R-AUC-PR	P	R	F1	R-AUC-PR
IF	0.6630	0.4919	0.5641	0.2058	0.6059	0.5328	0.5334	0.0942
MTAD-GAT	0.8763	0.8725	0.8744	0.4116	0.7321	0.7616	0.7432	0.1278
GDN	0.8750	0.8385	0.8564	0.3230	0.8668	0.8072	0.8360	0.1295
BeatGAN	0.9204	0.8767	0.8975	0.3453	0.7782	0.8512	0.8102	0.1421
USAD	0.9399	0.7348	0.8244	0.3369	0.8449	0.9641	0.8999	0.1176
OmniAnomaly	0.9551	0.8859	0.9191	0.3718	0.8321	0.8125	0.8221	0.1290
InterFusion	0.9533	0.9128	0.9326	0.1896	0.7688	0.9464	0.8442	0.1083
TranAD	0.9506	0.8951	0.9220	0.3994	0.8951	0.9297	0.9115	0.1057
ATF-UAD	0.9530	0.8974	0.9240	0.4125	0.5082	**0.9999**	0.6739	0.1722
DecompTran	0.9659	0.8633	0.9117	0.4093	**0.9412**	0.9093	0.9250	0.1326
ImDiffusion	0.9811	0.9753	0.9781	**0.4711**	0.8930	0.8638	0.8779	0.2381
DTAAD	0.9535	0.9812	0.9677	0.4223	0.8733	0.9997	0.9303	0.2443
CATCH	0.9819	0.9621	0.9722	0.4559	0.9111	0.9365	0.9230	0.2501
DeAnomaly	**0.9837**	**0.9864**	**0.9850**	0.4633	0.9389	0.9254	**0.9321**	**0.2576**

Method	SMAP				SWaT			
	P	R	F1	R-AUC-PR	P	R	F1	R-AUC-PR
IF	0.2886	0.7671	0.4163	0.1096	0.9764	0.6650	0.7907	0.0685
MTAD-GAT	0.9718	0.5259	0.6824	0.1083	0.8468	0.8224	0.8344	0.3196
GDN	0.9689	0.5401	0.6936	0.0961	0.9822	0.6932	0.8127	0.1318
BeatGAN	0.8915	0.6781	0.7663	0.1303	0.9606	0.7020	0.8107	0.3215
USAD	0.7582	0.9694	0.8509	0.1051	0.9675	0.7474	0.8415	0.2598
OmniAnomaly	0.8407	0.9674	0.8995	0.0978	0.9749	0.7500	0.8470	0.3722
InterFusion	0.8788	0.7704	0.8204	0.1457	0.8683	0.8530	0.8600	0.1477
TranAD	0.8224	0.8502	0.8360	0.1077	0.7025	0.7266	0.6886	0.1670
ATF-UAD	0.8863	**0.9999**	0.9397	0.1021	**0.9927**	0.6887	0.8131	0.2054
DecompTran	0.9704	0.8656	0.9146	0.1137	0.9504	0.7952	0.8659	0.1728
ImDiffusion	0.8771	0.9618	0.9175	0.1105	0.8988	0.8465	0.8709	0.1939
DTAAD	0.8364	0.9998	0.9206	0.1396	0.9690	0.6859	0.8177	0.1678
CATCH	**0.9826**	0.9137	0.9489	**0.1523**	0.9830	0.8803	0.9356	0.3306
DeAnomaly	0.9346	0.9822	**0.9578**	0.1486	0.9702	**0.9644**	**0.9673**	**0.3915**

Part of the results are from [23].

TABLE III
*F*1 AND R-AUC-PR PERFORMANCE COMPARISON OF DIFFERENT DECOMPOSITION METHODS ON FOUR DATASETS

Method		PSM		MSL		SMAP		SWaT		
		F1	R-AUC-PR	F1	R-AUC-PR	F1	R-AUC-PR	F1	R-AUC-PR	Points/Second
Moving Average	With model ReTSDec	**0.9850**	**0.4633**	**0.9321**	**0.2576**	**0.9578**	**0.1486**	**0.9673**	**0.3915**	9.3
	W/O model ReTSDec	0.9325	0.4438	0.8705	0.2392	0.8806	0.0916	0.9028	0.3804	**12.5**
Fast RobustSTL	With model ReTSDec	**0.9718**	**0.5108**	**0.9386**	**0.2425**	**0.9506**	**0.1417**	**0.9667**	**0.3746**	7.1
	W/O model ReTSDec	0.9511	0.4601	0.9196	0.2398	0.9375	0.1236	0.9456	0.3628	**9.9**
OneShotSTL	With model ReTSDec	**0.9863**	0.4867	**0.9343**	**0.2590**	**0.9568**	**0.1423**	**0.9652**	**0.3874**	7.8
	W/O model ReTSDec	0.9437	**0.4985**	0.9023	0.2411	0.9234	0.1339	0.9279	0.3525	**11.4**

Conversely, our proposed method uses the model ReTSDec to extract accurate structural patterns and employs remainders that genuinely represent anomalies for anomaly detection.

D. Effectiveness Evaluation

1) Contribution of the Model ReTSDec: We propose a model, ReTSDec, to address the first challenge. To verify its performance, we replace our decomposition method (i.e., moving average-based) with two advanced time series decomposition methods, Fast RobustSTL [47] and OneShotSTL [48], which can quickly and robustly extract various components from time series contaminated by anomalies or noise. In addition, we remove the model ReTSDec from each decomposition scheme to independently assess their respective decomposition

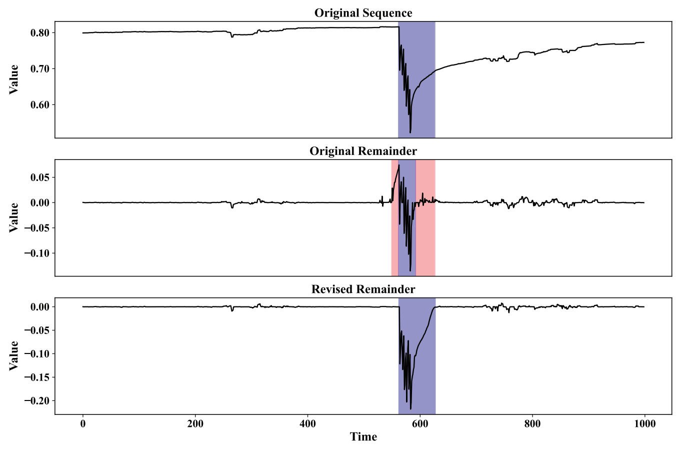

Fig. 4. Visualization of how model ReTSDec helps decompose the remainder.

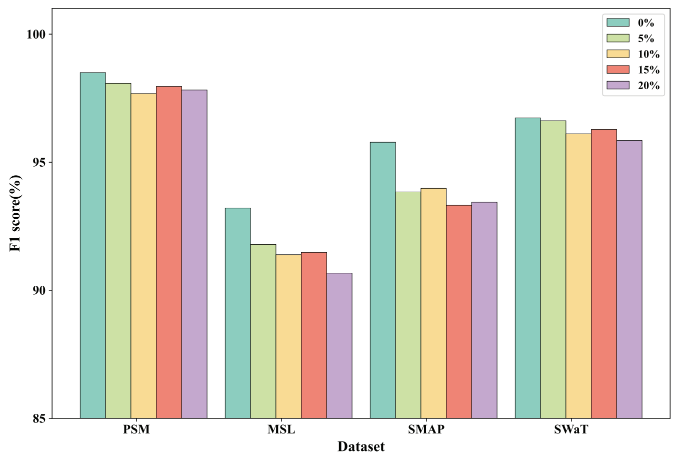

Fig. 5. Performance with different injection noise ratios.

performance. In Table III, we provide the $F1$ and R-AUC-PR performance comparison across all datasets over six runs, along with their average values. We also give the average inference efficiency (points/second) over all datasets. We can observe that both substitution approaches achieve better results in the absence of the model ReTSDec, which is related to their robust decomposition. When the model ReTSDec is added, the performance of all methods improves, with our model showing the most significant progress and performing best on some datasets. This further demonstrates the effectiveness of the model ReTSDec for improving various decomposition methods. In addition, the moving average-based decomposition method performs inference at the fastest rate of 9.3 data points per second. Considering both efficiency and performance, we ultimately choose the simplest decomposition method.

Furthermore, we visually compare different remainder components of our method on the PSM dataset. As shown in Fig. 4, the top-to-bottom subplots show the original time series, the remainder component without ReTSDec processing, and the remainder component after ReTSDec processing, respectively. The blue shaded areas represent the labeled anomalies in the first subplot and the predicted anomalies in the other subplots. It is apparent that when anomalies are present in the original series, some artificial anomalies are introduced in the remainder component obtained after decomposition, and some anomalies are overlooked, as illustrated by the red shading in the second subplot. Since we use additive decomposition, inaccurate remainders imply that the trend and seasonality components are also inaccurate. This corroborates our previous view that decomposition will be inaccurate when the original series are contaminated. However, the remainder component obtained after normal pattern extraction of trend and seasonal components by the model ReTSDec can accurately represent anomalies.

2) Contribution of the Model ADDiff: We introduce a diffusion-based model, ADDiff, to address the second challenge. When gathering time series data, noise is unavoidable due to various internal and external interferences. Some of the noise contained in the original dataset has already proved the robustness of our method to some extent. To further test the robustness of our model against noisy data, we randomly select $r\%$ of the values in the training set and substitute them with random numbers that do not follow any specific distribution, which will be closer to reality. Note that this approach can evaluate the model's robustness under both regular numerical noise and partial data loss scenarios. For the case where the data dimensions change, since our model requires the data dimensions to be consistent during training and testing, this limits its applicability in this scenario.

Fig. 5 shows the performance of our model on four datasets with different injection noise ratios. The results in the figure are the average of six independent runs. We can observe that on all datasets, as the noise ratio increases, the $F1$-scores of DeAnomaly slightly decline but still maintain relatively high performance. Even on the MSL dataset, where the performance drop is most pronounced, the $F1$-score span is only 2.54%. The possible reason is that the model ADDiff is designed based on diffusion models, which introduce noise during the training process and learn an inverse diffusion process to remove this noise. This makes the model tolerant to the presence of noise. In addition, through the storage of normal noise prototypes by the memory block, the model marks data points as anomalies only when they deviate from the normal pattern by a certain degree.

E. Ablation Experiment

To verify the effectiveness and necessity of each component of DeAnomaly, we conduct ablation studies on four multivariate datasets, and the results are summarized in Table IV. Note that all results shown in the table are averaged from six independent runs. The specific ablation studies are as follows.

1) *DeAnomaly^{-}:* The model directly uses the original sequences for anomaly detection without decomposition.
2) *W/O Graph:* Model ReTSDec without graph learning and graph attention.
3) *W/O Frequency:* Model ReTSDec without frequency-domain branch.
4) *W/O Guidance:* Model ReTSDec without time–frequency interaction layer.
5) *W/O Feature Fusion:* The time–frequency feature fusion layer in the model ReTSDec is replaced with a concatenation operation.

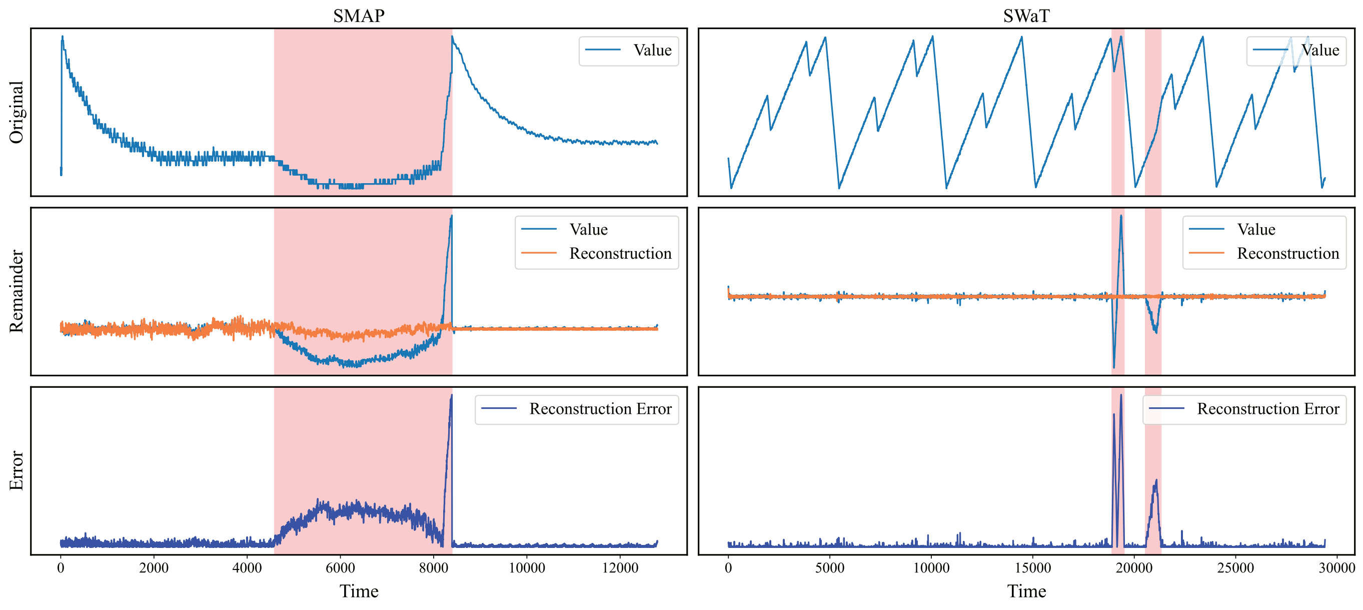

Fig. 6. Visualization analysis of anomaly detection. The first row illustrates the raw time series with anomalies, the second row depicts the revised remainders and their reconstruction results, and the final row displays the reconstruction error. Anomalies are highlighted with a red shaded background.

6) *W/O Memory:* The memory block removes the memory matrix.
7) *W/O Top-K Attention:* The memory blocks in the model ADDiff use full attention.
8) *W/O Focus Score:* The model ADDiff is simplified to use a one-step training approach.

Comparing the nine variants, we can observe that the complete model performs best. The significant performance degradation of DeAnomaly$^-$ compared to DeAnomaly indicates the necessity of avoiding complex temporal pattern interference in MTS-AD. The model's effectiveness diminishes when it does not incorporate spatial relationship modeling, especially for MSL datasets with a high number of sensors. This observation demonstrates the importance of capturing dependencies between sensors. The frequency-domain branch brings significant improvements, indicating that modeling sequences from a frequency domain perspective is meaningful. The effectiveness without the guidance matrix is dramatically reduced, illustrating that time–frequency interactions can facilitate sequence modeling. Attention-based feature fusion enables more comprehensive time–frequency integration, which helps model ReTSDec to extract accurate structural patterns. The memory matrix amplifies the reconstruction error of anomalies by storing normal prototypes, thereby improving the effectiveness of anomaly detection. Compared to standard full attention, our top-K attention strategy reduces irrelevant context during feature extraction to produce high-quality output. Finally, the focus score brings about a certain performance improvement by focusing more on parts with larger reconstruction errors.

F. Case Study

To demonstrate DeAnomaly's ability to detect anomalies using remainders, we present the revised remainders, reconstructed remainders, and reconstruction error results of two real-world datasets (SMAP and SWaT) in Fig. 6. The point-wise anomalies are bounded by red boxes. We can observe that the anomalies in the original time series are effectively mapped to the revised remainders. In addition, without the interference of structural patterns, these anomalies become more prominently exposed, resulting in significantly higher reconstruction errors, while the reconstruction errors of other points remain stable. This validates that our remainder detection method can highlight anomalies, thereby aiding in anomaly detection and reducing the false positive rate.

G. Sensitivity Analysis

In this section, we explore how various parameters affect the performance of DeAnomaly. We adjust each parameter setting individually, and the average results of six independent runs are shown in Fig. 7.

1) Dimension Size of Guidance Matrix: Fig. 7(a) illustrates the $F1$-scores under five different guidance matrix dimension size choices, i.e., $d_2 = \{9, 16, 25, 36, 49\}$. As d_2 increases, the model's ability to detect anomalies progressively enhances. For the SMAP dataset, the optimal $F1$-score is achieved when $d_2 = 36$, while for other datasets, $d_2 = 25$ produces the best results. For most datasets, setting d_2 too large can lead to excessive information interference and higher computational costs. As a result, we set $d_2 = 25$ in our experiments.

2) Length of Diffusion Step: Fig. 7(b) demonstrates the performance of our model under different diffusion steps. The length of the diffusion step affects both the performance and duration of anomaly detection. If the length is too short, it may result in insufficient denoising by the model and the inability to accurately capture the detailed features of the input series. Conversely, a step length that is too long might cause excessive denoising, making it unable to accurately reflect the anomalous

TABLE IV
ANOMALY DETECTION ACCURACY IN TERMS OF $F1$-SCORE AND R-AUC-PR OF DEANOMALY AND ITS VARIANTS ON FOUR DATASETS

Method	PSM		MSL		SMAP		SWaT	
	F1	R-AUC-PR	F1	R-AUC-PR	F1	R-AUC-PR	F1	R-AUC-PR
DeAnomaly	**0.9850**	0.4633	**0.9321**	0.2576	**0.9578**	0.1486	**0.9673**	**0.3915**
DeAnomaly$^-$	0.9286	0.3972	0.8553	0.2268	0.8711	0.1072	0.8642	0.3514
W/O Graph	0.9802	0.4423	0.9125	0.2389	0.9498	0.1405	0.9544	0.3821
W/O Frequency	0.9373	0.4031	0.8793	0.2106	0.8908	**0.1497**	0.9216	0.3682
W/O Guidance	0.9449	0.3901	0.8927	0.2202	0.8995	0.1133	0.9283	0.3657
W/O Feature Fusion	0.9423	**0.4727**	0.9037	0.2364	0.9112	0.1277	0.9322	0.3688
W/O Memory	0.9322	0.4284	0.9003	0.2071	0.9025	0.1149	0.9184	0.3501
W/O Top-K Attention	0.9409	0.4272	0.9014	**0.2754**	0.9317	0.1356	0.9376	0.3902
W/O Focus Score	0.9528	0.4575	0.9139	0.2419	0.9406	0.1367	0.9301	0.3629

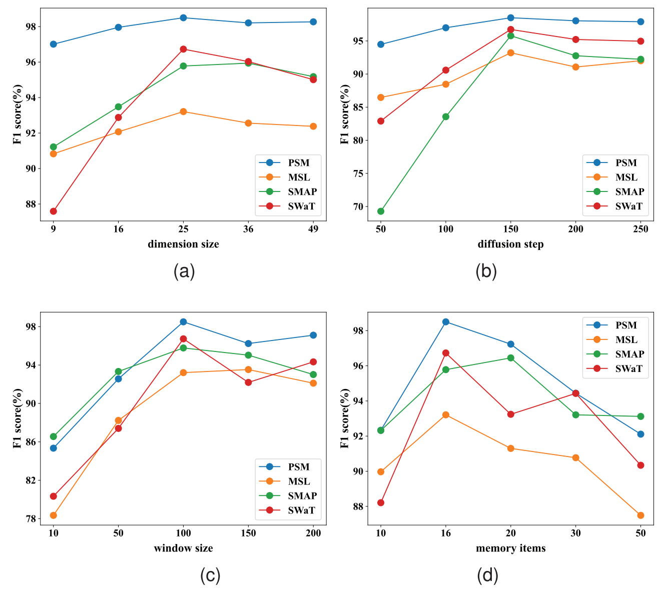

Fig. 7. Effect of different parameters on DeAnomaly's performance. (a) Dimension size of the guidance matrix. (b) Diffusion step. (c) Window size. (d) Number of memory items.

features in the original data. In addition, this will significantly increase the model's computational load and processing time. Taking the above analysis into account, we select a diffusion step length of 150 in the experiments.

3) Window Size: Fig. 7(c) illustrates the performance of DeAnomaly under different sliding window sizes. The window size has a crucial impact on MTS modeling. It can be observed that if the window size is too small, the limited data within the window is unable to effectively capture the relationships between the variables and the local temporal context, thus limiting the processing capability of the model, ReTSDec. However, if the window size is too large, the interactions between variables become more complex and bring higher computational cost and memory consumption. Considering this, we choose a window size of 100 in our experiments.

4) Number of Memory Items: Fig. 7(d) demonstrates the impact of different numbers of memory items on anomaly detection results. We can find that the optimal performance is achieved when the number of memory items is 16. The main reason is that if the number of memory items is too small, it becomes difficult to retain the essential features of the remainder sequence, resulting in a larger error for normal samples. On the contrary, a larger number of memory items increases the likelihood of the memory items being mixed with anomalous information, making it difficult for our model to effectively distinguish between anomalous and normal samples. In summary, the experiment chooses 16 memory items.

TABLE V
MODEL EFFICIENCY ON THE SWAT DATASET

Method	Inference efficiency (Points/Second)	Model size(MB)	F1
TranAD	**36.1**	51.42	0.6886
ATF-UAD	23.7	38.90	0.8131
DecompTran	26.2	62.13	0.8659
ImDiffusion	6.1	89.28	0.8709
DTAAD	12.7	56.78	0.8177
CATCH	32.4	**33.15**	0.9356
DeAnomaly	9.8	112.36	**0.9673**

H. Model Efficiency Analysis

We conduct a comprehensive comparison of the inference efficiency and model size of recent deep learning-based models on the SWaT dataset to validate the practicality of DeAnomaly in production environments. The results are presented in Table V. Despite requiring two-stage processing and multiple iterations of the diffusion model to obtain the final results, our method achieves a sufficient inference speed of 9.8 data points per second to meet the online requirements of most application scenarios. In addition, compared to the ImDiffusion detector, which is also based on diffusion models, our model demonstrates a faster inference speed. This is primarily due to our use of a more lightweight denoising neural network. Regarding model size, our model employs attention-related operations in the backbone networks of both stages, resulting in a relatively large size. However, considering the improvement in detection accuracy we have achieved, the model size of 112.36 MB remains manageable given the continuous expansion of hardware resources today. In summary, although our model is inferior in terms of real-time efficiency and model size, it has higher detection accuracy, which is critical for subsequent tasks.

VI. Conclusion

In this work, we propose DeAnomaly, a decomposition-based two-phase unsupervised anomaly detection framework. It employs graph-structure-based variable relationship modeling and time–frequency domain-based temporal relationship modeling to learn normal structural patterns, thereby avoiding interference from complex temporal patterns. It further utilizes diffusion models to perform anomaly detection on the remainders to overcome the impact of noise. Comprehensive experiments conducted on four real-world datasets highlight the advantages of our method. Our proposed DeAnomaly is a highly generalized model that is well-suited for modern industrial and IT systems that require accurate and robust anomaly detection.

DeAnomaly also has certain limitations, and our future work will focus on three aspects. First, DeAnomaly does not incorporate domain-specific prior knowledge during the model training process, which may lead to misjudgments of anomalies in some cases. In the future, we hope to further improve the model's performance by introducing relevant expert knowledge during the training process. Second, the iterative process of the diffusion model is relatively time-consuming. Therefore, we plan to utilize accelerated sampling algorithms for diffusion models to further enhance the detection efficiency of our method. Finally, our model is unable to effectively detect anomalies when there are changes in data dimensions. In the future, we will explore how to address this issue through methods such as adaptive input.

References

[1] S. Schmidl, P. Wenig, and T. Papenbrock, "Anomaly detection in time series: A comprehensive evaluation," *Proc. VLDB Endow.*, vol. 15, no. 9, pp. 1779–1797, 2022.

[2] G. Li and J. J. Jung, "Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges," *Inf. Fusion*, vol. 91, pp. 93–102, Mar. 2023.

[3] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," in *Int. Workshop Mach. Learn. Data Mining Pattern Recognit.* Cham, Switzerland: Springer, 2007, pp. 61–75.

[4] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proc. Int. Joint Conf. Neural Netw.*, 2003, pp. 1741–1745.

[5] W. A. Chaovalitwongse, Y.-J. Fan, and R. C. Sachdeo, "On the time series *k*-nearest neighbor classification of abnormal brain activity," *IEEE Trans. Syst., Man, Cybern. A, Syst. Hum.*, vol. 37, no. 6, pp. 1005–1016, Jun. 2007.

[6] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120043–120065, 2021.

[7] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 1–33, 2021.

[8] H. Zhao et al., "Multivariate time-series anomaly detection via graph attention network," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 841–850.

[9] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, May 2021, pp. 4027–4035.

[10] Y. Bai, J. Wang, X. Zhang, X. Miao, and Y. Lin, "CrossFuN: Multiview joint cross-fusion network for time-series anomaly detection," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–9, 2023.

[11] Z. Li et al., "Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 3220–3230.

[12] S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep transformer networks for anomaly detection in multivariate time series data," *Proc. VLDB Endow.*, vol. 15, no. 6, pp. 1201–1214, Feb. 2022.

[13] J. Wang, S. Shao, Y. Bai, J. Deng, and Y. Lin, "Multiscale wavelet graph AutoEncoder for multivariate time-series anomaly detection," *IEEE Trans. Instrum. Meas.*, vol. 72, pp. 1–11, 2023.

[14] S. Qin, J. Zhu, D. Wang, L. Ou, H. Gui, and G. Tao, "Decomposed transformer with frequency attention for multivariate time series anomaly detection," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Jun. 2022, pp. 1090–1098.

[15] C. Zhang, T. Zhou, Q. Wen, and L. Sun, "TFAD: A decomposition time series anomaly detection architecture with time–frequency analysis," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2022, pp. 2497–2507.

[16] J. Fan, Z. Wang, H. Wu, D. Sun, J. Wu, and X. Lu, "An adversarial time–frequency reconstruction network for unsupervised anomaly detection," *Neural Netw.*, vol. 168, pp. 44–56, Nov. 2023.

[17] Z. Zhang, R. Wang, R. Ding, and Y. Gu, "Unravel anomalies: An end-to-end seasonal-trend decomposition approach for time series anomaly detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2024, pp. 5415–5419.

[18] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, and S. Zhu, "RobustSTL: A robust seasonal-trend decomposition algorithm for long time series," in *Proc. AAAI*, vol. 33, Jun. 2019, pp. 5409–5416.

[19] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2256–2265.

[20] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, Jun. 2020, pp. 6840–6851.

[21] I. Pintilie, A. Manolache, and F. Brad, "Time series anomaly detection using diffusion-based models," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Dec. 2023, pp. 570–578.

[22] C. Yang, T. Wang, and X. Yan, "DDMT: Denoising diffusion mask transformer models for multivariate time series anomaly detection," 2023, *arXiv:2310.08800*.

[23] Y. Chen et al., "ImDiffusion: Imputed diffusion models for multivariate time series anomaly detection," 2023, *arXiv:2307.00754*.

[24] E. B. Dagum and S. Bianconcini, *Seasonal Adjustment Methods and Real Time Trend-cycle Estimation*. Cham, Switzerland: Springer, 2016.

[25] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A seasonal-trend decomposition," *J. Off. Stat.*, vol. 6, no. 1, pp. 3–73, 1990.

[26] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.

[27] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*. Melbourne, VIC, Australia: OTexts, 2018.

[28] S. Lim et al., "Long-term time series forecasting based on decomposition and neural ordinary differential equations," in *Proc. IEEE Int. Conf. Big Data (BigData)*, Dec. 2023, pp. 748–757.

[29] C. Ding, S. Sun, and J. Zhao, "MST-GAT: A multimodal spatial–temporal graph attention network for time series anomaly detection," *Inf. Fusion*, vol. 89, pp. 527–536, Jan. 2023.

[30] Y. Zheng et al., "Correlation-aware spatial–temporal graph learning for multivariate time-series anomaly detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 11802–11816, Jan. 2023.

[31] X. Zhou, C. Dai, W. Wang, and T. Qiu, "Global–local association discrepancy for multivariate time series anomaly detection in IIoT," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 11287–11297, Apr. 2024.

[32] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, Apr. 2022.

[33] D. Han et al., "Agent attention: On the integration of softmax and linear attention," 2023, *arXiv:2312.08874*.

[34] F. Pukelsheim, "The three sigma rule," *Amer. Statistician*, vol. 48, no. 2, pp. 88–91, May 1994.

[35] A. Abdulaal, Z. Liu, and T. Lancewicki, "Practical approach to asynchronous multivariate time series anomaly detection and localization," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 2485–2494.

[36] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2018, pp. 387–395.

[37] A. P. Mathur and N. O. Tippenhauer, "SWaT: A water treatment testbed for research and training on ICS security," in *Proc. Int. Workshop Cyber Phys. Syst. Smart Water Netw. (CySWater)*, 2016, pp. 31–36.

[38] H. Xu et al., "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications," in *Proc. World Wide Web Conf.*, 2018, pp. 187–196.

[39] J. Paparrizos, P. Boniol, T. Palpanas, R. S. Tsay, A. Elmore, and M. J. Franklin, "Volume under the surface: A new accuracy evaluation measure for time-series anomaly detection," *Proc. VLDB Endowment*, vol. 15, no. 11, pp. 2774–2787, Jul. 2022.

[40] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2019, pp. 8026–8037.

[41] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Min.*, May 2008, pp. 413–422.

[42] B. Zhou, S. Liu, B. Hooi, X. Cheng, and J. Ye, "BeatGAN: Anomalous rhythm detection using adversarially generated time series," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4433–4439.

[43] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: Unsupervised anomaly detection on multivariate time series," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2020, pp. 3395–3404.

[44] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2019, pp. 2828–2837.

[45] L.-R. Yu, Q.-H. Lu, and Y. Xue, "DTAAD: Dual TCN-attention networks for anomaly detection in multivariate time series data," *Knowl.-Based Syst.*, vol. 295, Jul. 2024, Art. no. 111849.

[46] X. Wu et al., "Catch: Channel-aware multivariate time series anomaly detection via frequency patching," in *Proc. Int. Conf. Learn. Represent.*, Apr. 2025.

[47] Q. Wen, Z. Zhang, Y. Li, and L. Sun, "Fast RobustSTL: Efficient and robust seasonal-trend decomposition for time series with complex patterns," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 2203–2213.

[48] X. He, Y. Li, J. Tan, B. Wu, and F. Li, "OneShotSTL: One-shot seasonal-trend decomposition for online time series anomaly detection and forecasting," *Proc. VLDB Endowment*, vol. 16, no. 6, pp. 1399–1412, Feb. 2023.

Hui Dou received the Ph.D. degree from the Department of Computer Science, Xi'an Jiaotong University, Xi'an, China, in 2017.

He was a Senior Engineer at Huawei, Shenzhen, China. From August 2018 to August 2020, he undertook post-doctoral research at Sun Yat-sen University, Guangzhou, China. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. He has published more than 20 papers in some international conferences and journals, such as ASE, ICDCS, ICPP, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, and JPDC. He is currently interested in AI-driven performance diagnosis and optimization for distributed systems.

Pengcheng Shi received the B.E. degree in information management and information systems from Yancheng Institute of Technology, Yancheng, China, in 2023. He is currently pursuing the master's degree with the School of Computer Science and Technology, Anhui University, Hefei, China.

His current research topics mainly focus on anomaly detection and root cause localization in microservice systems.

Yiwen Zhang received the Ph.D. degree in management science and engineering from Hefei University of Technology, Hefei, China, in 2013.

He is currently a Professor with the School of Computer Science and Technology, Anhui University, Hefei. Meanwhile, he is a Ph.D. Advisor. He has published more than 70 papers in some international conferences, including SIGIR, ICSOC, and ICWS, and journals, including IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, ACM TOIS, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON MOBILE COMPUTING, ACM TKDD, and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. His research interests include service computing, cloud computing, and big data analytics.

Pengfei Chen received the Ph.D. degree from the Department of Computer Science, Xi'an Jiaotong University, Xi'an, China, in 2016.

He was aResearch Scientist at IBM Research, Beijing, China. He is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-sen University, Zhuhai, China. He has published more than 50 papers in some international conferences, including ACM/IEEE ICSE, ACM ESEC/FSE, IEEE INFOCOM, WWW, IEEE ICSOC, and IEEE ICWS, and journals, including IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON RELIABILITY, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, and IEEE TRANSACTIONS ON CLOUD COMPUTING. He is currently interested in distributed systems, AIOps, cloud computing, microservices, and blockchain.

Zibin Zheng (Fellow, IEEE) is currently a Professor and the Deputy Dean of the School of Software Engineering, Sun Yat-sen University, Guangzhou, China. He authored or co-authored more than 200 international journal and conference papers, including one ESI hot paper and ten ESI highly cited papers. According to Google Scholar, his papers have more than 36 000 citations. His research interests include blockchain, software engineering, and services computing.

Prof. Zheng was a recipient of several awards, including the Top 50 Influential Papers in Blockchain of 2018 and the ACM SIGSOFT Distinguished Paper Award at ICSE 2010.