Identifying Performance Anomalies in Fluctuating Cloud Environments: A Robust Correlative-GNN-based Explainable Approach

Yujia Song[a,1], Ruyue Xin[b,1], Peng Chen[a,*], Rui Zhang[a], Juan Chen[a], Zhiming Zhao[b,*]

[a]School of Computer and Software Engineering, Xihua University, Chengdu, China.
[b]Multiscale Networked Systems (MNS), University of Amsterdam, Amsterdam, Netherlands.

Abstract

Cloud computing provides scalable and elastic resources to customers as a low-cost, on-demand utility service. Multivariate time series anomaly detection is crucial to promise the overall performance of cloud computing systems. However, due to the complexity and high dynamics of cloud environments, anomaly detections caused by irre gular fluctuations in data and the robustness of models are a challenge. To address these issues, we propose a deep learning-based anomaly detection method for multivariate time series for real-world operational clouds: Correlative-GNN with Multi-Head Self-Attention and Auto-Regression Ensemble Method (CGNN-MHSA-AR). Our method utilizes two parallel graph neural networks (GNN) to learn the time and feature inter-dependencies to achieve fewer false positives. Our approach leverages a multi-head self-attention, GRU, and AR model to capture multiple-dimensional information, leading to better detection robustness. CGNN-MHSA-AR can also provide an abnormal explanation based on the prediction error of its constituent univariate series. We compare the detection performance of CGNN-MHSA-AR with seven baseline methods on seven public datasets. The evaluation shows that the proposed CGNN-MHSA-AR outperforms its competitors with an F1-Score of 0.871 on average and is 19.9% better than state-of-the-art baseline methods. In addition, CGNN-MHSA-AR also offers to correctly identify the root cause of detected anomalies with up to 74.1% accuracy.

Keywords: Deep Anomaly Detection, Multivariate Time Series, Graph Neural Networks, Multi-head Self-attention, Cloud Computing, Anomaly Explanation

1. Introduction

Cloud computing can store, aggregate, and configure resources on demand and provide users with personalized services. Therefore, providing reliable performance to users and ensuring service level agreements (SLAs) is essential for cloud computing systems[1]. However, because of the complexity and high dynamics of underlying infrastructures in clouds, anomalies such as physical resource breakdown may occur in cloud computing systems. With monitoring tools, performance data such as resource usage of cloud computing systems can be collected[2]. At the same time, anomaly detection to build a profile of performance data and detect deviations from the profile for cloud computing systems can be developed[3]. Considering it is tedious and time-consuming to label data manually because various anomalies exist, unsupervised learning involves picking up interesting structures in the data, and learning features without labels is popular[4]. As a result, unsupervised anomaly detection to identify abnormal behaviors and predict anomalies to forestall future incidents is required in cloud computing systems.

Performance data of cloud computing systems, such as CPU and memory usage, are usually represented as multivariate time series. Multivariate time series reflects the health status of a cloud computing system and can be used to identify abnormal behavior or events in real-time[5]. The main target of real-time anomaly detection models is to improve detection accuracy, reduce false positive rate (FPR), and achieve better performance[6]. However, the rapid increase of resources, sensors, and applications in cloud computing systems will cause irregular data fluctuations and increase the FPR of anomaly detection. For example, sudden changes in a specific feature, e.g., CPU usage, do not necessarily mean anomalies in the system. Figure 1 shows the normal fluctuations of a cloud platform server. The red box part may be detected as an anomaly due to fluctuating CPU and memory usage. Still, the system is healthy during this period, and disk IO and network traffic fluctuate steadily. To avoid high FPR caused by this situation, it is vital to consider correlations between variables in multivariate time series to differentiate normal fluctuations from anomalies.

Furthermore, improving detection robustness is essential to meet changes in data patterns in multivariate time series and keep detection performance consistent. Deep learning-based methods for multivariate time series anomaly detection have been developed recently. For example, STGCN[7] is a novel GNN-based model tackling the time series prediction problem in the traffic domain and improving detection accuracy with multi-head self-attention. OmniAnomaly[8] utilizes a stochas-

*Corresponding author
Email addresses: chenpeng@mail.xhu.edu.cn (Peng Chen), z.zhao@uva.nl (Zhiming Zhao)

[1]These authors contributed to the work equally and should be regarded as co-first authors.

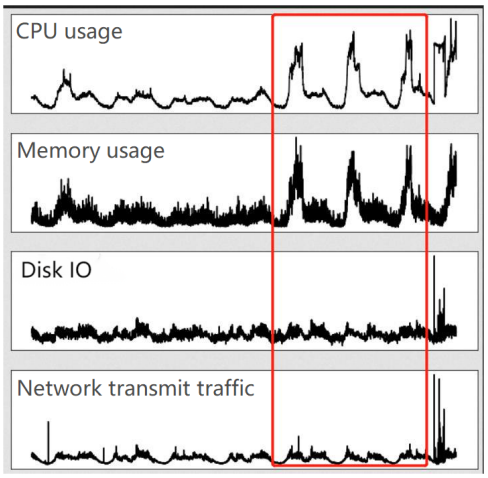


Figure 1: An example of multivariate time series. The red box represents normal fluctuations.

tic recurrent neural network to capture long-term temporal information and a planar normalizing flow to generate reconstruction probabilities. However, existing deep learning-based detection methods mainly target improving detection accuracy for specific scenarios, which cannot meet the requirements of complex and dynamic cloud computing systems. Therefore, the need for an unsupervised deep learning-based method for multivariate time series anomaly detection that can detect anomalies accurately and keep detection robustness becomes urgent.

Deep learning makes it possible to extract information from unstructured data. The graph neural network (GNN) is good at processing graph structure data and mining inter-dependencies between nodes. We can use GNN to mine inter-dependencies from feature and time dimensions for multivariate time series. We need to extract as much information as possible from multivariate time series to improve detection robustness. Multi-head self-attention has been developed to extract context information in sequence data, and gate recurrent unit (GRU) is good at capturing long-term temporal dependencies. In addition, the autoregressive (AR) model can be used to maintain linear relations in time series. Based on these ideas, we propose a Correlative-GNN with Multi-Head Self-Attention and Auto-Regression Ensemble Method (CGNN-MHSA-AR) for unsupervised multivariate time series anomaly detection in cloud computing systems, and efficient abnormal explanation results are achieved on five public datasets.

By using two parallel graph neural networks, we are able to distinguish normal from abnormal fluctuations and to derive a correlation between sequences, thereby reducing the likelihood of false positives. In addition, to further extract context information and enhance the robustness of the anomaly detection model, we integrated multiple deep learning methods, such as the multi-head self-attention mechanism and GRU. Finally, in order to speed up troubleshooting, we have added the function of abnormal interpretation and quickly find the features that may cause abnormalities based on the abnormal score.

The contributions of this paper are summarized as follows:

1. We present two parallel GNNs that can analyze correlations between features and time in a multivariate time series to avoid fluctuations in normal data being falsely identified as anomalies.
2. We ensemble different deep learning techniques, such as multi-head self-attention and GRU, which learn data features from multiple dimensions to enhance the robustness of detection.
3. We perform the interpretation task by identifying features that may cause anomalies. We design the forecast error of our model as a generic indicator to detect and interpret anomalies in multivariate time series.
4. We conduct experiments on public datasets to evaluate the detection performance and interpretation ability of CGNN-MHSA-AR. Results show that our model outperforms other detection models in anomaly detection accuracy, robustness, and interpretation.

The rest of the paper is organized as follows: Section 2 reviews existing research about multivariate time series anomaly detection in cloud computing systems. In section 3, we propose the CGNN-MHSA-AR model and provide a detailed description of each module. In section 4, we conduct experiments and provide experimental results and analysis. Finally, we draw our conclusion in section 5.

2. Related Work

This section explores existing unsupervised deep-learning methods for multivariate time series anomaly detection and explanation, and then we introduce correlation discovery in multivariate time series.

2.1. Unsupervised Multivariate Time Series Anomaly Detection and Explanation

Researchers usually develop unsupervised multivariate time series anomaly detection methods with deep learning methods for high-dimensional and unlabeled data. Long short-term memory (LSTM)-based VAE-GAN[9] uses LSTM as an encoder, generator, and discriminator to detect anomalies via reconstruction difference and discrimination results. USAD[10] uses an autoencoder with two decoders and an adversarial game-like training framework to classify normal and abnormal data. LSTM-VAE[11] integrates LSTM into variational autoencoder (VAE) to perform multivariate time series anomaly detection. Ensemble Learning-Based Detection (ELBD)[12] framework which integrates four existing well-selected detection methods for performance anomaly detection and prediction of cloud applications. HTA-GAN[13] model is a novel heterogeneous BIGAN-based anomaly detection model that uses popular GAN-based generative models and one-class classification to improve unsupervised anomaly detection. A novel anomaly score, DR-score, is employed by MAD-GAN[14] to

detect anomalies by discrimination and reconstruction using GAN's generator and discriminator outputs.

[15] proposed a new GAN-based framework for anomaly detection and localization and a transformation method for time-series imaging called range images. The technique converts multivariate time series into two-dimensional images through the structure of an encoder and decoder. It utilizes residual images and anomaly-scoring functions to detect and explain anomalies. OmniAnomaly[8] proposed a stochastic model for multivariate sequence anomaly detection. It captures the normal patterns of data by learning robust representations of multivariate time series with random variable connections and a plane regularization process. OmniAnomaly also provides anomaly interpretation capabilities based on time series reconstruction probabilities. TranAD[16] is a deep transformer network-based anomaly detection and diagnosis model that uses an attention-based sequence encoder to infer information about temporal trends quickly. CAT-IADEF[17] proposes a novel convolutional adversarial model, adopting three convolutional neural networks to learn sequence features and adversarial training to amplify "slight" anomalies while enhancing the robustness of the model. In addition, CAT-IADEF compares the number of anomalies that may occur on each dimension of the time series data for the explanation of anomalies.

These detection algorithms enable unsupervised multivariate anomaly detection with advanced deep learning methods and target improving detection accuracy. However, due to the high data volatility in cloud computing systems, anomaly detection by these algorithms may need to be more accurate. Therefore, combining deep learning-based unsupervised anomaly detection methods with correlation discovery to improve detection accuracy can be considered.

2.2. Correlation Discovery of Multivariate Time Series

Previous studies have shown that correlation discovery in data is crucial for time series anomaly detection[18]. MSCRED[19], CCG-EDGAN[20] utilize the inner product between vectors to generate a signature matrix to extract correlations between different features in time series. The MTAD-GAT[21] considers each univariate time series as an individual feature and includes two graph attention layers in parallel to learn the complex dependencies of multivariate time series in both temporal and feature dimensions.

To tackle real-time anomaly detection in operational cloud environments, especially differentiating rare abnormal performance issues (anomalies) from frequent normal fluctuations, we found that the existing methods proposed above could not adequately extract correlation information, and the linear dependence among the data needs to be extracted properly. Moreover, anomaly explanation is also necessary for further troubleshooting and system auto-healing. Therefore, our method is proposed to effectively improve the capability of the model to distinguish normal fluctuations from abnormality. At the same time, we add a multi-head self-attention mechanism to extract context information and use the AR model to compensate for the performance degradation of the neural network when processing the linear part of the data regularities. In addition, we selected some representative datasets to demonstrate that our method can distinguish normal fluctuations from abnormality more effectively. Furthermore, our approach adds the function of anomaly interpretation to speed up cloud computing troubleshooting. In this function, we can find features that cause anomalies as the root causes.

3. Methodology

In this section, we provide the problem statement of unsupervised multivariate time series anomaly detection and introduce the overall architecture of our model in detail.

3.1. Problem Statement

We define multivariate time series in a cloud computing system as $X = \{x_1, x_2, \cdots x_n\}$, where n is the number of timestamps in a sliding window. We also define $x_t = \left\{v_t^1, v_t^2, \cdots v_t^m\right\}$ as a vector at time t, where m represents the number of features. For data $X \in R^{n \times m}$, the task of multivariate time series anomaly detection is to learn the characteristics of data X and determine whether an observation x_{n+1} is anomalous or not.

For multivariate time series anomaly explanation, our goal is to find the root cause of the anomaly. Having located the abnormal time point $x_t = \left\{v_t^1, v_t^2, \cdots v_t^m\right\}$ in the test set, we have to determine which feature at that time point is abnormal.

Typical unsupervised deep learning methods usually train and model normal data of multivariate time series and identify abnormal points through high reconstruction errors, such as the LSTM-VAE[11]. LSTM-VAE extracts dependencies between time series by replacing the feedforward network in VAE with LSTM and exploits the reconstruction error to detect anomalies. However, the LSTM-VAE ignores feature inter-dependencies and contextual information in multivariate time series, which may increase the false positive rate in anomaly detection.

To improve the detection accuracy and robustness of multivariate time series anomaly detection, we first provide two parallel GNNs to learn inter-dependencies in both feature and time dimensions. We then integrate multi-head self-attention to capture context information, GRU to extract long-term dependency and AR model to maintain linear relations in multivariate time series. We will introduce the overall architecture and detailed modules next.

3.2. Overall Architecture

The overall architecture of CGNN-MHSA-AR is shown in Figure 2. The graph neural network can mine the relationship between different nodes, so we use correlative GNN. In addition, the multi-head self-attention mechanism captures the information of the same sequence in different representation subspaces by combining multiple parallel self-attention calculations and then obtains more comprehensive related features from various angles and levels, so we chose multi-head self-attention to extract the information's contextual information further. Finally, the neural network's performance will decrease when processing linear features. To maintain the linear dependence of the data, we integrated the AR model so that the entire

model can maintain the linear and nonlinear relationship of the data simultaneously. The specific module structure is as follows:

Data preprocessing: we perform data preprocessing with data normalization and denoising for original multivariate time series.

1D-CNN feature extraction: we use the one-dimensional convolutional layer to extract local patterns of each feature in preprocessed data.

Correlation calculation based parallel GNNs: we provide parallel GNNs to learn inter-dependencies in multivariate time series from feature and time dimensions, and correlation calculation is used in this module.

Context learning via multi-head self-attention: we use a multi-head self-attention to extract context information in multivariate time series.

Time-series forecasting via GRU: we use a GRU to capture the dependencies between different time series.

Autoregressive model ensembling: we utilize an AR model to maintain linear relations in original data.

Anomaly Detection and Explanation: we utilize anomaly scoring functions for anomaly detection and anomaly explanation.

3.3. Modules of CGNN-MHSA-AR

Data preprocessing. We apply *min-max* data normalization for original data to ensure that all data has the same scale. In addition, we adopt the Fourier transform to denoise data as shown in Figure 2.(1). We treat each row of time series as univariate time series and use Fast Fourier Transform (FFT) to denoise and replace detected noise with zero.

1D-CNN feature extraction. The one-dimensional convolution neural network (1D-CNN) is widely used in sequence processing because it can recognize local patterns of sequences. For preprocessed multivariate time series, we use the 1D-CNN with kernel size 7 to extract information along the time dimension, as shown in Figure 2.(2).

Correlation calculation based parallel GNNs. Generally, given a graph, we can use GNN to get new representations of nodes by considering inter-dependencies between nodes. For multivariate time series, inter-dependencies in data can be exploited with correlation calculation in both feature and time dimensions. However, traditional correlation calculation methods, such as Pearson correlation, do not work when a cloud computing platform is stable because multivariate time series will keep certain values unchanged[22]. In most previous studies, the inner product between vectors has been used to calculate the correlation and has achieved good results. Therefore, the correlation in time series is calculated using the vectors' inner product, and we will consider both feature and time correlation calculations.

For feature correlation calculation, we define a feature as a vector $x_i = \left\{{v_i}^1, v_i^2, \cdots, {v_i}^n\right\}$, where n is the number of timestamps in a sliding window. We utilize the inner product between different vectors to get correlations of different features. We use $c_{ij}^{feature}$ to represent the correlation between feature i and j, and the formula is as follows:

$$c_{ij}^{feature} = \frac{\sum_{t=0}^{n} v_i^t v_j^t}{k} \tag{1}$$

where k represents the rescaling factor, and equal to the length of a sliding window in this paper. We calculate feature correlations of all features in the multivariate time series and finally obtain the feature correlation matrix with a shape of $m \times m$. m represent the number of features in the multivariate time series.

For time correlation calculation, we define the vector at time i as $x_i = \left\{{v_i}^1, v_i^2, \cdots, {v_i}^m\right\}$, where m is still the number of features in the multivariate time series. We can use c_{ij}^{time} to represent the correlation between time i and j, and the formula is as follows:

$$c_{ij}^{time} = \frac{\sum_{k=0}^{m} v_i^k v_j^k}{\lambda} \tag{2}$$

where λ represents the rescaling factor, and equal to the length of a sliding window. We calculate time correlations between of all timestamps and finally obtain the time correlation matrix with a shape of $n \times n$. n represents the length of a sliding window.

As shown in Figure 2.(3.1), we use feature correlations between a vector i with others as weights of a GNN and calculate the output representation of the vector as follows:

$$h_i = \sigma(\sum_{j=1}^{m} c_{ij}^{feature} v_i^j) \tag{3}$$

where σ represents the sigmoid activation function. Similarly, for Figure 2.(3.2), we use time correlations between a vector i with others as weights of a GNN and calculate the output representation of the vector as follows:

$$h_i = \sigma(\sum_{j=1}^{n} c_{ij}^{time} v_i^j) \tag{4}$$

We obtained the correlation between the time series and the feature sequence through the parallel graph neural network. Then, to gather the extracted information, we spliced the two parts of the data together. Finally, in order to preserve the original characteristics of the data, we concatenate the data processed by the convolutional neural network with the data processed by two graph neural networks, resulting in a shape of $n \times 3m$, where each row represents a $3m$ dimensional feature vector for each timestamp.

Context learning via multi-head self-attention. As shown in Figure 2.(4), to make the network's complexity scales with the input size, we set the multi-head self-attention mechanism with m heads and $3m$ embedding dimension to learn different contextual information from data where m is the number of features in the multivariate time series.

Time series forecasting via GRU. After multi-head self-attention, we use a GRU to extract time dependence in time series, as shown in Figure 2.(5). We set the neurons of GRU to 150 and the number of layers to 1. Finally, We use a fully

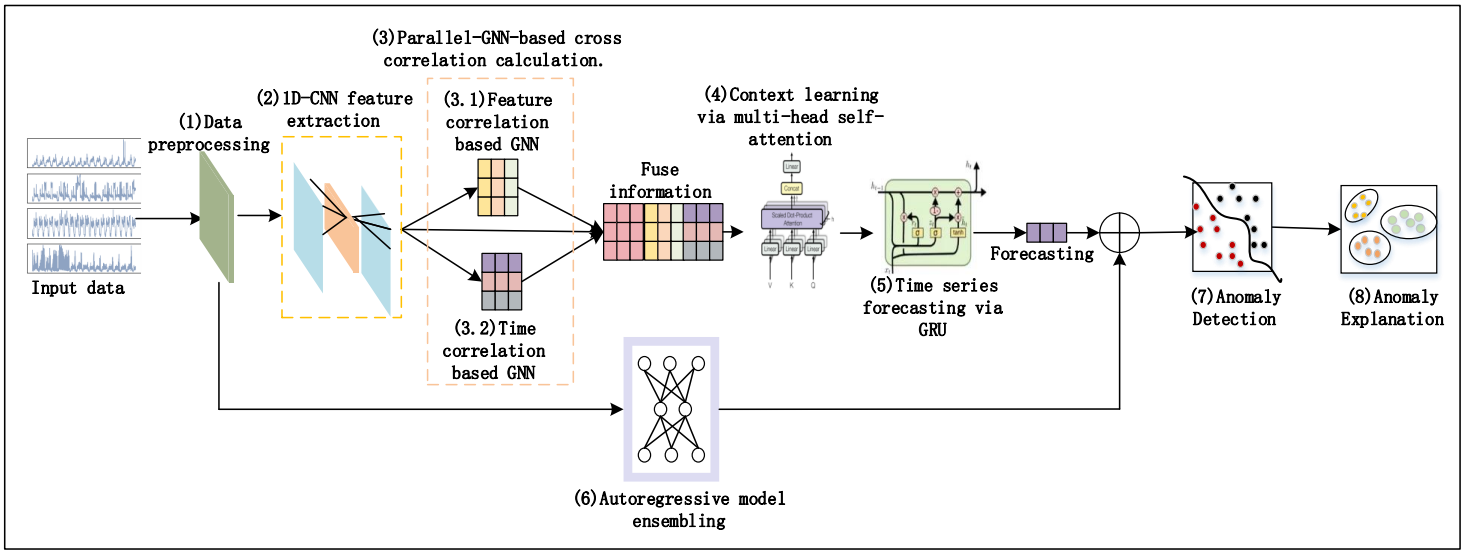


Figure 2: Overall architecture of CGNN-MHSA-AR.

connected layer to predict the value of the next timestamp to get the first predicted value $x_{n+1,i}^{forecast}$. We use $x_{n+1,i}^{forecast}$ to represent the predicted value of the i-th feature at the time n+1.

Autoregressive model ensembling. Due to the nonlinearity of convolutional, multi-head self-attention, and GRU modules, the output is not sensitive to the original input[23]. To address this drawback, we apply a first-order autoregressive (AR) model to the preprocessed data to obtain the second predicted value. We use $x_{n+1,i}^{AR}$ to represent the predicted value of the i-th feature at the time n+1 after the AR model. A final prediction result is obtained by integrating the results from the first and second predictions. The formula is as follows:

$$\hat{x}_{n+1,i} = \alpha x_{n+1,i}^{forecast} + (1-\alpha) x_{n+1,i}^{AR} \tag{5}$$

where α is to adjust the nonlinear and AR prediction results. In this paper, we set $\alpha = 0.5$.

Finally, we define the root mean square error (RMSE) as loss function:

$$Loss = \sqrt{\sum_{i=1}^{m} (x_{n+1,i} - \hat{x}_{n+1,i})^2} \tag{6}$$

Anomaly detection. After training the model, we get the predicted value $\hat{x}_{n+1}$ at time n+1. We follow [24] to use the error between the actual value x_{n+1} and the predicted value as the anomaly score, and the formula is as follows:

$$S_{n+1} = \frac{1}{m}\sum_{i=1}^{m} s_{n+1}^{i} = \frac{1}{m}\sum_{i=1}^{m} \sqrt{(x_{n+1,i} - \hat{x}_{n+1,i})^2} \tag{7}$$

We identify a timestamp as an anomaly if its anomaly score is larger than a threshold.

Anomaly explanation. Abnormal explanation aims to identify what features may cause anomalies at a timestamp. Consequently, abnormal explanation in cloud computing systems is highly significant in practice to speed up troubleshooting. For a detected anomaly x_t, we sort $\hat{x}_t^i$ in descending order, where $\hat{x}_t^i$ represents the anomalous score of the i-th dimension in x_t. A smaller ranking indicates a greater likelihood of the feature causing anomalies. Finally, we select the top k features as abnormal root causes.

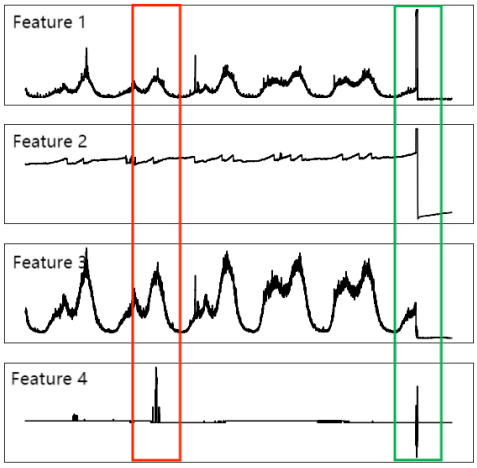
4. Experiments and Analysis

In this section, we conduct experiments to evaluate the anomaly detection accuracy and performance of abnormal explanations of CGNN-MHSA-AR. We first compare the detection performance of CGNN-MHSA-AR with baseline methods on seven public datasets. Then we provide ablation experiments to analyze the importance of modules in CGNN-MHSA-AR. Finally, we utilize five datasets to test the ability of abnormal explanation of CGNN-MHSA-AR.

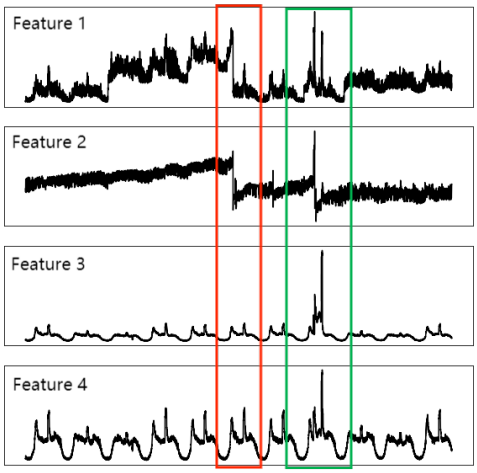
4.1. Datasets

We use seven public datasets in this paper. SMD (Server Machine Dataset) is a five-week-long real-time dataset of 28 cloud platform servers[8], which contain 708405 data points from the training set and 708420 data points from the testing set. The anomaly rate of SMD is 4.16%. In this dataset, we found that the four datasets, machine-1-3 (the training set has 23702 data points and the testing set has 23703 data points), machine-1-8 (23698 data points in the training and test sets each), machine-2-6 (28743 data points in the training and test sets each), and machine-3-5 (the training set has 23690 data points and the testing set has 23691 data points), did not perform well in many detection models because the irregular fluctuations in the data would lead to false positives in abnormal detection. We analyze that irregular fluctuations in data cause false positive anomaly detection, as shown in Figure 3. Figure

3(a) and Figure 3(b) illustrate four-feature segments for machines 1-3 and 1-8, respectively. It is possible to detect anomalies in the red box in Figure 3(a) because of fluctuations in Feature 4. While features 1, 2, and 3 fluctuate steadily, the system operates in a healthy state. When we look at the green box, each feature has apparent fluctuations, representing abnormal data segments. Figure 3(b) shows a similar scenario. Due to fluctuations in features 1 and 2, anomalies may be detected in the red box, which will lead to lower detection accuracy. Therefore, we use these four datasets to prove that our model can effectively resolve the false positive issue and improve detection accuracy.



(a) A sample segment in SMD machine-1-3 dataset.



(b) A sample segment in SMD machine-1-8 dataset.

Figure 3: Typical segment in datasets that has both normal fluctuations and anomalies. An anomalous data set is displayed in green box while a normal data set is displayed in red box.

SMAP (Soil Moisture Activate Passive satellite) and MSL (Mars Science Laboratory rover) are spacecraft datasets provided by NASA[8]. The abnormal rate is 13.13% and 10.72%, respectively. Our method proposed in this paper is intended to effectively distinguish normal and abnormal fluctuations for high-dimensional and complex data in the operational cloud environment. Therefore, in MSL and SMAP, we did not choose the complete datasets. Instead, we selected a section of the datasets containing a number of normal and anomalous ones as experimental datasets to prove that our method can better distinguish between normal and abnormal fluctuations. For MSL, we choose 28317 data points from the training set and 20000 from the testing set. And 20000 data points are selected from the training set and 20000 from the testing set in SMAP.

4.2. Evaluation Metric

4.2.1. Anomaly Detection

We use precision, recall, and F1-score and F1 Average Rank to validate anomaly detection performance of models.

$$precision = \frac{TP}{TP + FP} \tag{8}$$

$$recall = \frac{TP}{TP + FN} \tag{9}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{10}$$

with TP the True Positives, FP the False Positives, and FN the False negatives. F1 Average Rank represents the average Rank of F1-Score of each model in the seven datasets. To get the best F1-score, we enumerate all possible anomaly thresholds to search for the best F1-score, denoted as best-f1[8].

4.2.2. Abnormal Explanation

We use two general metrics, HitRate@P% and normalized discounted cumulative gain (NDCG), to evaluate the model's anomaly explanation ability and the correct rate of anomaly explanation.

$$HitRate@P\% = \frac{Hit@\lfloor P\% \times |GT_t| \rfloor}{|GT_t|} \tag{11}$$

$$CR@k = \frac{number\ of\ correct\ location\ in\ topk}{number\ of\ anomalies} \tag{12}$$

where GT_t is the set of ground truth, and $|GT_t|$ is the length of GT_t.

We quantify roots causes ranking accuracy using NDCG, a popular measure for relevance evaluation.

4.3. Experimental Settings

We use Python 3.7 and CPU-only PyTorch 1.11.0. We set the sliding window size as $n = 100$, the 1D-CNN with kernel size 7. We set the neurons of GRU to 150 and the number of layers to 1, the multi-head self-attention mechanism with m heads and $3m$ embedding dimension. To train CGNN-MHSA-AR, we set epochs as 10, batch size as 256, learning rate as 0.001, and dropout as 0.4, and we use the Adam optimizer.

4.4. Experimental Results

4.4.1. Performance of Anomaly Detection

We provide comparison results between CGNN-MHSA-AR with baseline methods on seven public datasets and an analysis of their detection performance.

Table 1: Comparison of anomaly detection performance for multiple detection methods on seven datasets. The best F1-Score are highlighted in bold.

Method	machine-1-3			machine-1-8			machine-2-6			machine-3-5			SMD			MSL			SMAP			F1 Average Rank
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	
LODA	0.348	0.581	0.435	0.476	0.460	0.468	0.996	0.651	0.787	0.725	0.694	0.709	0.555	0.729	0.631	0.931	0.647	0.764	0.509	0.999	0.674	5.85
CBLOF	0.974	0.925	**0.949**	0.663	0.424	0.517	0.997	0.959	**0.978**	0.838	0.938	0.885	0.600	0.761	0.671	0.854	0.925	0.888	0.435	0.999	0.607	3.42
HBOS	0.987	0.401	0.571	0.252	0.463	0.326	0.851	0.660	0.743	0.507	0.999	0.340	0.992	0.198	0.330	0.205	0.999	0.341	0.396	0.999	0.568	8.14
IForest	0.947	0.924	0.935	0.472	0.853	0.608	0.911	0.898	0.904	0.614	0.704	0.656	0.601	0.505	0.549	0.669	0.925	0.775	0.302	0.999	0.464	5.00
DeepSVDD	0.468	0.806	0.592	0.709	0.854	0.775	0.702	0.905	0.791	0.832	0.971	0.897	0.788	0.334	0.470	0.981	0.730	0.837	0.462	0.272	0.342	5.42
GDN	0.285	0.558	0.377	0.461	0.250	0.324	0.874	0.294	0.440	0.444	0.617	0.518	0.227	0.151	0.181	0.205	0.942	0.345	0.197	0.917	0.343	9.42
MTAD-GAT	0.449	0.873	0.593	0.599	0.433	0.503	0.885	0.620	0.703	0.725	0.584	0.647	0.642	0.773	0.702	0.974	0.944	**0.959**	0.427	0.999	0.598	5.00
MAD-GAN	0.303	0.594	0.401	0.594	0.918	0.721	0.679	0.999	0.809	0.790	0.584	0.649	0.791	0.395	0.527	0.935	0.944	0.940	0.608	0.999	0.756	4.85
LSTM-VAE	0.274	0.873	0.418	0.283	0.338	0.308	0.616	0.660	0.637	0.824	0.584	0.684	0.667	0.738	0.701	0.843	0.944	0.891	0.658	0.727	0.691	6.00
CGNN-MHSA-AR	0.837	0.870	0.853	0.712	0.956	**0.816**	0.777	0.999	0.875	0.918	0.955	**0.936**	0.839	0.867	**0.853**	0.906	0.944	0.925	0.727	0.999	**0.842**	**1.85**

Comparison results. We selected five statistical models and four deep learning models as baseline methods in order to demonstrate that our model outperforms linear and nonlinear models. The specific methods including five statistical methods(from PYOD[25]): LODA, IForest, CBLOF, HBOS and DeepSVDD; and four deep learning methods: MTAD-GAT[21], GDN[26], MAD-GAN[14], LSTM-VAE[11]. We calculate the best-f1 of each model and present the comparison results in Table 1.

Table 1 shows that CGNN-MHSA-AR outperforms all other methods on machine-1-8, machine-3-5, SMD, and SMAP. The average F1 on these four datasets can reach 86.1%. The F1 of CGNN-MHSA-AR is slightly lower than the best baselines on machine-1-3, machine-2-6, and MSL. CGNN-MHSA-AR outperforms the state-of-the-art method (MTAD-CAT) on all six datasets except the MSL dataset, and 26%, 31%, 17%, 28%, 15%, 24% relatively increase F1-scores. The robustness of CGNN-MHSA-AR is much better than all baselines because the precision of CGNN-MHSA-AR is above 0.7, and the recall is above 0.85 on all seven datasets, while no baseline can achieve this. In terms of the average ranking of the F1-score, CGNN-MHSA-AR also performs best.

As shown in Figures 4, we can see that the CGNN-MHSA-AR performs well in the F1-score, and the fluctuations of CGNN-MHSA-AR on seven datasets are all small, which proves the excellent robustness of CGNN-MHSA-AR. Furthermore, on average, as shown in Figure 5, CGNN-MHSA-AR has the best ranking across the three evaluation metrics for all datasets.

Performance analysis. Baseline methods have different performances on these public datasets. LODA is a lightweight anomaly detector that is very practical in sensor failure. LODA consists of multiple one-dimensional histograms, which approximate the probability density of the input data and project it into a single vector. A low density indicates an enormous outlier in the sample. However, LODA's insufficient dependency extraction between time series results in its poorer performance than CGNN-MHSA-AR.

Based on the characteristics of a sample, HBOS divides it into multiple intervals, and intervals with fewer samples are more likely to be outliers. As a result, HBOS performs well in global anomaly detection but cannot detect local outliers. GDN analyzes sensor relationships based on a graph and then identifies deviations from learned patterns. However, GDN ignores correlations in the time dimension, which makes it hard to predict various behaviors. CGNN-MHSA-AR extracts correlations between different times and features in parallel, making it perform better than GDN and HBOS.

CBLOF is a cluster-based local outlier detector that uses clusters to identify dense regions in data and then performs a density estimate for each cluster. When a data point deviates significantly from most data, it is considered abnormal. IForest defines anomalies as sparsely distributed points far away from groups with high density. Due to the small density of abnormal points, the tree model can easily detect abnormal points. Consequently, the anomalous data will be closer to the root of the isolation tree when it is established, while the normal data will be farther from it. The performance of CBLOF and IForest on machine-1-3 and machine-2-6 is better than CGNN-MHSA-AR because of the low density of outliers in these two datasets.

The MAD-GAN uses an LSTM-based GAN model to model the time-series distribution with generators. As well as using prediction error, this study uses discriminator loss to calculate anomaly scores. An LSTM-VAE performs multivariate anomaly detection using LSTM in combination with a variational autoencoder (VAE). It is significant to note that these methods do not consider feature interdependencies or contextual information when analyzing multivariate time series. This may result in a higher rate of false positives.

MTAD-GAT considers each univariate time series as a feature. It uses two graph attention layers simultaneously to learn the temporal and feature dimension dependencies of multivariate time series. However, MTAD-GAT ignores the maintenance of linear relationships for dynamic periodic data, decreasing prediction accuracy. In contrast, the CGNN-MHSA-AR addresses this problem by adding the AR model's prediction results. For the MSL dataset, MTAD-GAT extracts correlations between data with a better weight matrix, leading to a slightly better performance than CGNN-MHSA-AR.

4.4.2. Ablation Experiments

We conduct ablation experiments to analyze the influence of convolutional layers, correlation calculation of feature and time dimensions, GRU, and AR model in CGNN-MHSA-AR. Table 2 shows experimental results.

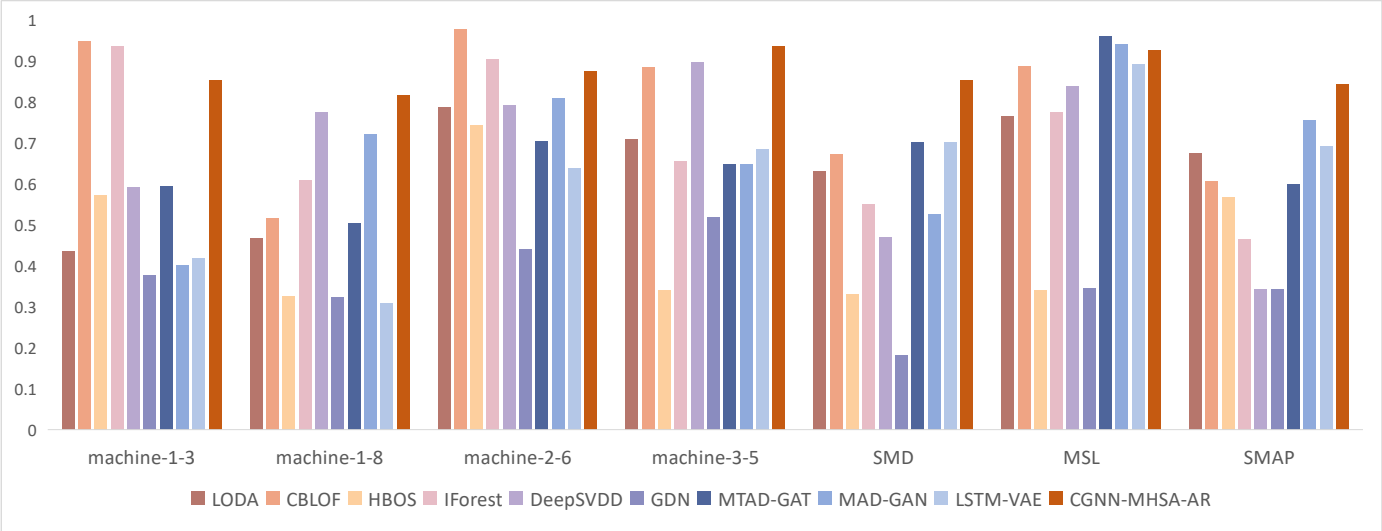


Figure 4: F1-score of *CGNN-MHSA-AR* and all baseline models.

Table 2: Ablation Study:F1-Score for CGNN-MHSA-AR and its ablated versions

Method	machine-1-3	machine-1-8	machine-2-6	machine-3-5	SMD	MSL	SMAP
CGNN-MHSA-AR	**0.853**	**0.816**	**0.875**	**0.936**	**0.853**	**0.925**	**0.842**
w/o conv	0.851	0.794	0.842	0.929	0.849	0.883	0.811
w/o time-correlation	0.778	0.790	0.827	0.929	0.850	0.917	0.701
w/o feature-correlation	0.616	0.758	0.847	0.927	0.852	0.882	0.820
w/o GRU	0.763	0.789	0.801	0.911	0.851	0.923	0.815
w/o ar	0.473	0.487	0.770	0.649	0.681	0.793	0.636

Table 3: Anomaly detection results with various α for seven datasets.

α	machine-1-3			machine-1-8			machine-2-6			machine-3-5			SMD			MSL			SMAP		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
$\alpha = 0.2$	0.823	0.870	0.846	0.691	0.956	0.802	0.686	0.999	0.813	0.870	0.964	0.915	0.834	0.855	0.844	0.902	0.944	0.923	0.601	0.999	0.751
$\alpha = 0.4$	0.781	0.870	0.823	0.669	0.956	0.787	0.707	0.999	0.828	0.922	0.955	0.938	0.835	0.853	0.843	0.903	0.882	0.892	0.761	0.727	0.744
$\alpha = 0.6$	0.829	0.870	0.849	0.654	0.956	0.777	0.717	0.999	0.835	0.920	0.955	0.937	0.839	0.867	0.853	0.894	0.944	0.918	0.662	0.727	0.693
$\alpha = 0.8$	0.833	0.870	0.851	0.570	0.956	0.714	0.803	0.999	0.890	0.730	0.962	0.830	0.850	0.868	0.859	0.871	0.736	0.798	0.645	0.999	0.784

The influence of convolutional layers. The impact of convolutional layers on the anomaly detection ability of our proposed model can be seen in Table 2 w/o conv, and an average of 2% reduces the best-f1. For machine-2-6, the best-f1 is reduced most, 3.3%. Therefore, these results show that convolutional layers can help CGNN-MHSA-AR better extract correlations between the temporal and feature dimensions via convolution.

The influence of correlation calculation. We verify the effect of correlation calculation on CGNN-MHSA-AR by removing the time and feature correlations, denoting w/o time-correlation and w/o feature-correlation in Table 2. We can discover that in w/o time-correlation, the best-f1 value is reduced by 4.4% on average. For SMAP, the correlation calculation of the time dimension has the most significant impact, and 14.1% reduces the best-f1. In w/o feature correlation, the best-f1 is reduced by 5.6% on average. For machine-1-3, the correlation calculation for the feature dimension has the most significant impact, and 23.7% reduces the best-f1. From these results, we can see that feature and time correlation calculations can extract interdependencies between non-adjacent vectors, which plays a crucial role in the final performance of our model.

The influence of GRU. We verify the effect of GRU in our model by removing GRU, denoting w/o GRU in Table 2. We can see that the average best-f1 value drops by 3.5% when we remove the GRU. The GRU has the most significant impact on machine-2-6, and the best-f1 drops by 7.4% after removing the

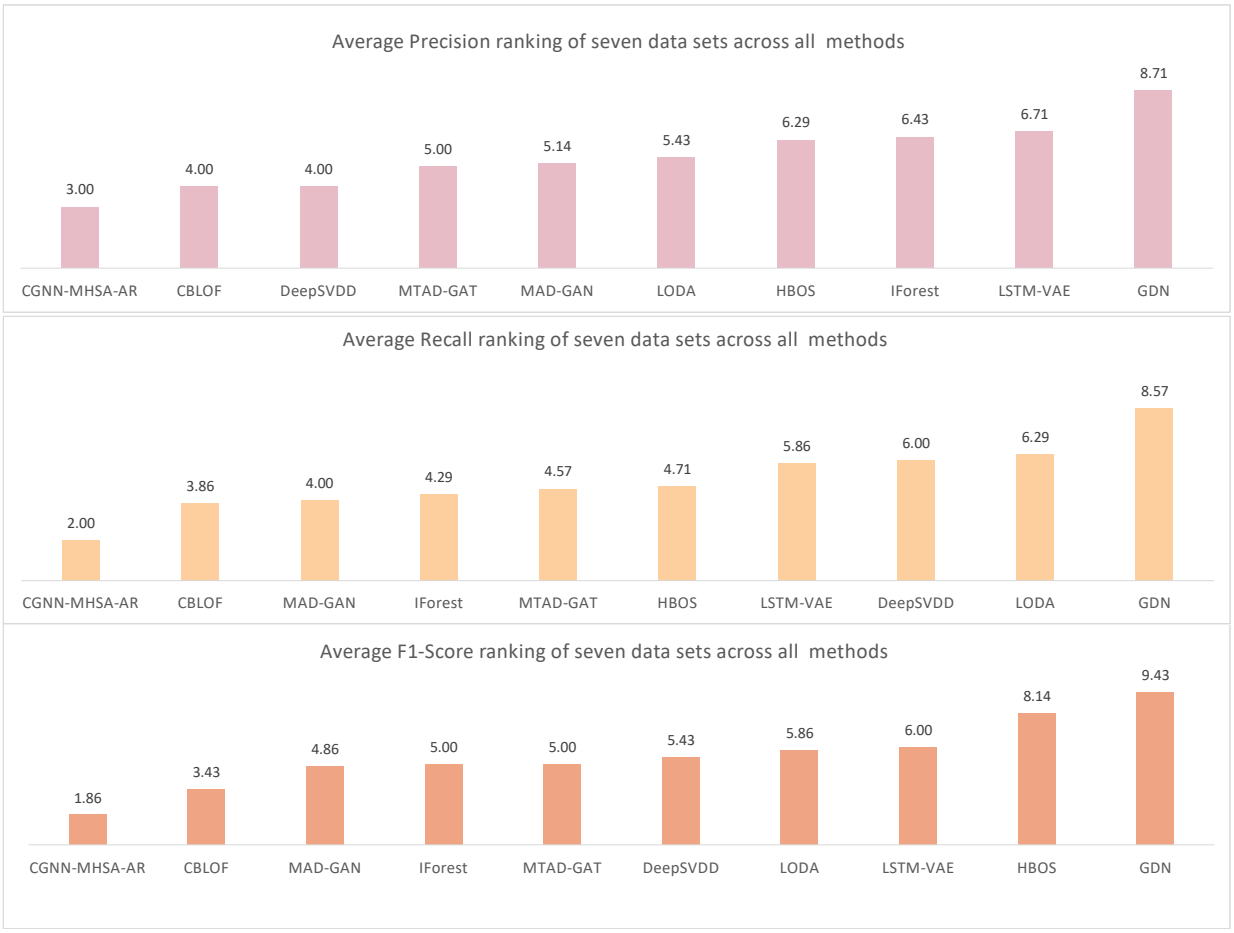


Figure 5: Average Precision, Recall, F1-score ranking of seven data sets across all methods.

Table 4: Comparison of anomaly explanation performance for multiple detection methods on five datasets. The best performance are highlighted in bold.

Metric	machine-1-3			machine-1-8			machine-2-6			machine-3-5			SMD		
	GDN	MTAD-GAT	CGNN-MHSA-AR	GDN	MTAD-GAT	CGNN-MHSA-AR	GDN	MTAD-GAT	CGNN-MHSA-AR	GDN	MTAD-GAT	CGNN-MHSA-AR	GDN	MTAD-GAT	CGNN-MHSA-AR
NDGC@100%	0.311	0.302	**0.501**	0.349	0.408	**0.413**	0.334	0.441	**0.551**	0.559	0.541	**0.593**	0.388	0.420	**0.512**
NDGC@150%	0.350	0.337	**0.556**	0.381	0.447	**0.450**	0.394	0.495	**0.604**	0.644	0.644	**0.705**	0.458	0.497	**0.587**
HitRate@100%	0.301	0.299	**0.502**	0.316	0.356	**0.367**	0.322	0.450	**0.530**	0.524	0.493	**0.552**	0.363	0.400	**0.488**
HitRate@150%	0.365	0.359	**0.592**	0.371	0.421	**0.429**	0.421	0.538	**0.618**	0.668	0.666	**0.741**	0.482	0.530	**0.613**
CR@16	0.458	0.446	**0.800**	0.425	0.446	**0.577**	0.583	0.563	**0.625**	0.611	0.576	**0.685**	0.513	0.495	**0.628**
CR@20	0.567	0.560	**0.899**	0.510	0.501	**0.678**	0.681	0.677	**0.704**	0.702	0.688	**0.782**	0.634	0.595	**0.741**

GRU. The GRU can extract the time dependency in time series and output the predicted value considering the hidden status of previous data. The gated structure ensures that important information will not disappear during long-term propagation, which makes GRU improve the abnormal detection ability of CGNN-MHSA-AR.

The influence of AR model. We show the effect of an AR model on the anomaly detection ability of CGNN-MHSA-AR in Table 2 w/o ar. We can see that the best-f1 is reduced by 23% on average, with the most significant impact on machine-1-3, which has a 38% reduction in the best-f1 value. The AR model predicts data based on previous data linearly. We add the AR model because the output after GRU in CGNN-MHSA-AR is not sensitive to the original data. Experimental results show that the AR model improves the anomaly detection performance of our model.

4.4.3. *Effect of parameters*

We study the role of the sensitivity threshold (equation 5). A Large α corresponds to predictors that emphasize nonlinearity in the anomaly score. In comparison, a small α corresponds to predicting results that put more emphasis on linearity. Table 3 reports the effect of varying α in the number of detected precision, recall, and the F1 score.

We can observe that the value of α has a more significant influence on precision. When α is changed, there can be an in-

crease or decrease in the number of true samples among the samples that are predicted to be positive. Consequently, by choosing a reasonable α, the accuracy of the anomaly detection model can be increased. It is possible to reduce α when the collected data has more linear characteristics. Conversely, the anomaly detection accuracy can be improved with a higher α if the data has more nonlinear characteristics. At the same time, from the experimental results(Table 1) of the statistical model in the baseline method, only linear feature prediction cannot achieve acceptable anomaly detection accuracy for the datasets we chose. Therefore, the accuracy of anomaly detection and the robustness of the model can be improved by integrating the prediction of the nonlinear characteristics of the data by the neural network.

4.4.4. Performance of Abnormal Explanation

We evaluate the abnormal explanation capability of CGNN-MHSA-AR on four sub-datasets and SMD because they have the ground truth of abnormal explanation while SMAP and MSL do not. As shown in Table 4, CGNN-MHSA-AR can detect anomalous features with an accuracy of 36.7%-74.1%. Compared to the state-of-the-art baseline methods, CGNN-MHSA-AR can improve explanation accuracy by up to 23.3% for machine-1-3, 1.1% for machine-1-8, 11% for machine-2-6, 7.5% for machine-3-5 and 9% for SMD. 9.64% improves the explanation accuracy on average.

At the same time, we selected the top 16 and 20 result rankings to evaluate the correct rate of abnormal explanation of CGNN-MHSA-AR. Table 4 shows that CGNN-MHSA-AR achieves the highest correct rate of 0.8 at the top 16 features and 0.89 at the top 20 features. In contrast to state-of-the-art baseline methods, CGNN-MHSA-AR can increase the correct rate by an average of 15.7% in the top 16 features and 15.6% in the top 20 features.

5. Conclusion

Anomaly detection is crucial to ensure the reliability of cloud computing systems. To detect anomalies in cloud computing systems, we propose an unsupervised anomaly detection method based on multivariate time series, known as CGNN-MHSA-AR. In this paper, to effectively distinguish between normal and abnormal fluctuations, we have proposed several innovations, and their connections are as follows:

1. In order to effectively distinguish normal and abnormal fluctuations in time series, we used two parallel graph neural networks to extract the correlation between nodes. Then, we fused the data of these two parts with the data processed only by the convolutional layer to obtain a matrix. This matrix fuses the relationship between different time series, the relationship between different features, as well as the characteristics of the original data. To further extract the contextual information and time dependence, we choose the multi-head self-attention mechanism and GRU to learn the features of the matrix so as to improve the accuracy and robustness of the model's anomaly detection.
2. We use a parallel graph neural network combined with a multi-head self-attention mechanism and GRU to learn the nonlinear characteristics of the data, find out the nonlinear dependency between the data, and predict the value of the next time step to obtain the first predicted value. However, the neural network has a robust nonlinear mapping ability, but its performance will decline to a certain extent when dealing with the linear features of the data. So we use the AR model to learn the linear features of the data and predict the second predicted value. Finally, the model's anomaly detection accuracy and robustness are improved by integrating the neural network results and the AR model.
3. From the perspective of anomaly detection and anomaly interpretation, when an anomaly is detected, it is impossible to quickly determine which component caused the anomaly in the multivariate time series, resulting in a slowdown in troubleshooting. Therefore, we designed the anomaly interpretation module to determine the characteristics that may cause abnormalities by calculating and sorting the anomaly scores of each feature in the multivariate time series and to speed up troubleshooting in cloud computing systems.

In experiments, we use seven public datasets to evaluate our model. Regarding the best-f1 score, CGNN-MHSA-AR outperforms all baseline methods in seven datasets. Compared with the state-of-the-art baseline method, CGNN-MHSA-AR increases the best-f1 up to 31.3%. Furthermore, on the seven datasets, the precision of CGNN-MHSA-AR is above 0.7, and the recall is above 0.85, reflecting the excellent robustness of our model. The model has also been shown to correctly determine the root causes of 74.1% of detected anomalies, a higher percentage than the state-of-the-art models.

In future work, we consider two more aspects that can be explored to improve the model CGNN-MHSA-AR. First, the optimization of neural networks in CGNN-MHSA-AR to reduce model training time will be researched in the future. Second, we will apply our model to more complex situations in cloud-edge computing systems, such as multiple anomalies, to provide effective anomaly detection.

Acknowledgment

This work has been partially supported by the European Union's Horizon 2020 research and innovation programme by the ARTICONF project grant agreement No.825134, by the ENVRI-FAIR project grant agreement No.824068, by the BLUECLOUD project grant agreement No.862409, by the LifeWatch ERIC, by China Scholarship Council, Science

and Technology Program of Sichuan Province under Grant No.2020JDRC0067 and No.2020YFG0 326 and Talent Program of Xihua University under Grant No.Z202047.

References

[1] C. Huang, G. Min, Y. Wu, Y. Ying, K. Pei, Z. Xiang, Time series anomaly detection for trustworthy services in cloud computing systems, IEEE Transactions on Big Data (2017).

[2] M. S. Aslanpour, S. S. Gill, A. N. Toosi, Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research, Internet of Things 12 (2020) 100273.

[3] N. Moustafa, M. Keshk, K.-K. R. Choo, T. Lynar, S. Camtepe, M. Whitty, Dad: a distributed anomaly detection system using ensemble one-class statistical learning in edge networks, Future Generation Computer Systems 118 (2021) 240–251.

[4] B. A. NG, S. Selvakumar, Anomaly detection framework for internet of things traffic using vector convolutional deep learning approach in fog environment, Future Generation Computer Systems 113 (2020) 255–265.

[5] R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, arXiv preprint arXiv:1901.03407 (2019).

[6] A. Sgueglia, A. Di Sorbo, C. A. Visaggio, G. Canfora, A systematic literature review of iot time series anomaly detection solutions, Future Generation Computer Systems (2022).

[7] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, arXiv preprint arXiv:1709.04875 (2017).

[8] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, D. Pei, Robust anomaly detection for multivariate time series through stochastic recurrent neural network, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2828–2837.

[9] Z. Niu, K. u, X. Wu, Lstm-based vae-gan for time-series anomaly detection, Sensors 20 (13) (2020) 3738.

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

[11] D. Park, Y. Hoshi, C. C. Kemp, A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder, IEEE Robotics and Automation Letters 3 (3) (2018) 1544–1551.

[12] R. Xin, H. Liu, P. Chen, Z. Zhao, Robust and accurate performance anomaly detection and prediction for cloud applications: a novel ensemble learning-based framework, Journal of Cloud Computing 12 (1) (2023) 1–16.

[13] P. Chen, H. Liu, R. Xin, T. Carval, J. Zhao, Y. Xia, Z. Zhao, Effectively detecting operational anomalies in large-scale iot data infrastructures by using a gan-based predictive model, in: The Computer Journal, Vol. 65, 2022, pp. 2909–2925.

[14] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, S.-K. Ng, Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks, in: International conference on artificial neural networks, Springer, 2019, pp. 703–716.

[15] Y. Choi, H. Lim, H. Choi, I.-J. Kim, Gan-based anomaly detection and localization of multivariate time series data for power plant, in: 2020 IEEE international conference on big data and smart computing (BigComp), IEEE, 2020, pp. 71–74.

[16] S. Tuli, G. Casale, N. R. Jennings, Tranad: Deep transformer networks for anomaly detection in multivariate time series data, arXiv preprint arXiv:2201.07284 (2022).

[17] P. Wen, Z. Yang, L. Wu, S. Qi, J. Chen, P. Chen, A novel convolutional adversarial framework for multivariate time series anomaly detection and explanation in cloud environment, Applied Sciences 12 (20) (2022) 10390.

[18] D. Song, N. Xia, W. Cheng, H. Chen, D. Tao, Deep r-th root of rank supervised joint binary embedding for multivariate time series retrieval, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2229–2238.

[19] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, N. V. Chawla, A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 33, 2019, pp. 1409–1416.

[20] H. Liang, L. Song, J. Du, X. Li, L. Guo, Consistent anomaly detection and localization of multivariate time series via cross-correlation graph-based encoder–decoder gan, IEEE Transactions on Instrumentation and Measurement 71 (2021) 1–10.

[21] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, Q. Zhang, Multivariate time-series anomaly detection via graph attention network, in: 2020 IEEE International Conference on Data Mining (ICDM), IEEE, 2020, pp. 841–850.

[22] H. Liang, L. Song, J. Wang, L. Guo, X. Li, J. Liang, Robust unsupervised anomaly detection via multi-time scale dcgans with forgetting mechanism for industrial multivariate time series, Neurocomputing 423 (2021) 444–462.

[23] S. Huang, D. Wang, X. Wu, A. Tang, Dsanet: Dual self-attention network for multivariate time series forecasting, in: Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 2129–2132.

[24] G. Pang, C. Shen, L. Cao, A. V. D. Hengel, Deep learning for anomaly detection: A review, ACM Computing Surveys (CSUR) 54 (2) (2021) 1–38.

[25] Y. Zhao, Z. Nasrullah, Z. Li, Pyod: A python toolbox for scalable outlier detection, arXiv preprint arXiv:1901.01588 (2019).

[26] A. Deng, B. Hooi, Graph neural network-based anomaly detection in multivariate time series, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 4027–4035.