Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

MFGAD-INT: in-band network telemetry data-driven anomaly detection using multi-feature fusion graph deep learning

Yunfeng Duan[1†], Chenxu Li[2†], Guotao Bai[1], Guo Chen[1], Fanqin Zhou[2*], Jiaxing Chen[1], Zehua Gao[3*] and Chun Zhang[1]

Abstract

As the cloud services market grows, cloud management tools that detect network anomalies in a non-intrusive manner are critical to improve users' experience of cloud services. However, some network anomalies, such as Microburst, in cloud systems are very discreet. Network monitoring methods, e.g., SNMP, Ping, are of coarse temporal granularity or low-dimension metrics, have difficulty to identify such anomalies quickly and accurately. Network telemetry is able to collect rich network metrics with fine temporal granularity, which can provide deep insight into network anomalies. However, the rich features in the telemetry data are insufficient exploited in existing research. This paper proposes a Multi-feature Fusion Graph Deep learning approach driven by the In-band Network Telemetry, shorted as MFGAD-INT, to efficiently extract and process the spatial-temporal correlation information in telemetry data and effectively identify the anomalies. The experimental results show that the accuracy performance of the proposed method improves about 10.56% compared to the anomaly detection method without network telemetry and about 9.73% compared to the network telemetry-based method.

Keywords Anomaly detection, Time series analysis, In-band network telemetry, Deep learning, Data stream mining, Cloud computing

Introduction

With the Digital transformation of various industries, an increasing number of AI-driven services [1] that rely heavily on data and computing power are being run on edge and center cloud facilities [2]. The pooling and virtualization of computing and storage resources in clouds [3–5], as well as smart scheduling and maintenance techniques [6, 7], have shielded the management details of the infrastructure, providing convenience for enterprises' digital transformation, but also making network anomalies more concealed. At the same time, emerging intelligent businesses [8, 9], such as splitting learning for large neural network models and federated learning models [10], put more stringent requirements on cloud data center networks. Network anomalies that are insensitive to traditional businesses, such as Microburst, can cause delays or packet loss during the parameter transfer process of split learning, severely affecting the efficiency and convergence of model training. Common network anomaly monitoring methods mainly utilize simple network manage protocol (SNMP), active network probes, etc., to obtain network status data [11]. The frequency of obtaining network status data using such methods is very

†Yunfeng Duan and Chenxu Li contributed equally to this work.

*Correspondence:
Fanqin Zhou
fqzhou2012@bupt.edu.cn
Zehua Gao
zhgao@bupt.edu.cn
[1] China Mobile Information Technology Co., Ltd., 102206 Beijing, China
[2] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, 100876 Beijing, China
[3] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, 100876 Beijing, China

low (minute level), and the dimensions of the detected indicators are also very limited. It is also difficult to continuously monitor the overall network performance. Therefore, it is difficult to effectively monitor more concealed network anomalies, which is one of the emerging research directions of network anomaly detection [12–14].

Data plane network telemetry techniques, such as in-band network telemetry (INT), offer high-precision stream monitoring data that can uncover potential network anomalies. However, the increased richness of information provided by network telemetry significantly increases the volume of data that needs to be processed, posing new challenges to anomaly detection. To ensure real-time detection of network anomalies, detection methods need to process telemetry data streams with utmost efficiency. Furthermore, since the data is continuously streamed, it is not possible to assume its distribution or length. Lastly, network telemetry data contains a wealth of network information, and detection algorithms must capture as many features from the information as possible to maximize the capabilities of INT telemetry. Although deep learning has been widely used in a variety of network problems, such as resource scheduling [15], health prediction [16, 17], and shows great potential in dealing with problems with large solution space, but the use of deep learning in telemetry-aided anomaly detection is rare. Therefore, it is crucial to develop efficient and effective detection methods that can extract essential information from incoming measurements using the simplest possible operation.

Considering the above challenges, this paper proposes the in-band network telemetry data-driven anomaly detection method using multi-feature fusion graph deep learning (MFGAD-INT). This method extracts multimodal features from different perspectives using graph neural networks to process high-density telemetry data streams obtained from telemetry systems, and performs multimodal feature fusion and learning via deep neural networks. To achieve fast network anomaly identification based on the latest network state, MFGAD-INT constructs an anomaly scoring model and determination model based on network state prediction. To effectively extract features from different perspectives, the proposed method uses the GAT [18] mechanism to extract temporal and spatial feature information of the input data respectively [19]. Additionally, the proposed method performs level-by-level data fusion of multimodal data and long-time information learning via the gating recurrent unit (GRU) [20] to model network state changes and predict future network states. To validate the effectiveness of the proposed method, we conducted experiments on a cloud data center network in Zhejiang, China, using its INT capability and controlled injection of anomalous states. The contributions of this paper can be summarized as follows.

(1) This paper proposes an anomaly feature learning framework for effectively processing INT telemetry data for anomaly detection. It can extract features from both spatial and temporal aspects of network telemetry data. At the same time, the framework realizes the gradual fusion of multimodal data and further complements the learning of long-term memory.
(2) Based on the above feature learning framework, this paper implements a graph learning based network anomaly detection method MFGAD-INT. By combining GAT [18] and GRU to extract multimodal feature information from INT data, MFGAD-INT can maximize the utilization of rich information in telemetry data when processing INT network telemetry data.
(3) We evaluated MFGAD-INT in a real network environment and compared it with other anomaly detection algorithms. The results show that MFGAD-INT outperforms other algorithms in terms of detection accuracy, demonstrating stronger generalization in detecting multiple types of anomalies, such as Microburst [21] or QoS anomaly [22–24], while providing a degree of anomaly localization.

The rest of this paper is structured as follows. Related work section reviews the various techniques related to network anomaly detection. In Framework of telemetry-based anomaly detection system section, the system model construction of this paper is presented, along with a general introduction of the proposed MFGAD-INT. In Method section, the anomaly detection framework of MFGAD-INT based on GAT and GCN is presented. Simulation and results analysis section shows the results of our tests implemented in a real environment by INT with programmable switches. Finally, in Conclusion section, the paper is summarized and future work is discussed.

Related work

This section provides an overview of related work focusing on the anomaly detection of streaming data in cloud data center networks, while meticulously dividing the related work according to its characteristics and algorithmic principles, as shown in Table 1. In particular, we first introduce network anomaly detection methods based on

Table 1 Related work classification table

Method	Data	Algorithm	Technical characteristics
HS-Trees [25]	No reliance	ML	The detection of anomaly is quickly achieved by decision tree that does not require changes in tree structure.
RShash [26]		ML	Using random hashing to detect subspace anomaly
LSTM [27]		CNN	Using LSTM AutoEncoder pairs to extract Feature, and use SVM to complete the binary classification of input.
ICNAD [28]		GNN	Network anomaly detection is achieved using GNN that fuse the nodes' attributes and neighboring nodes'.
GDN [29]		GNN	Improving the accuracy of anomaly detection by using GAT with graph neural network.
Snappy [21]	Network Telemetry	Statistical Analysis	A Microburst detection method implemented inside programmable switches.
BurstRadar [30]		Statistical Analysis	A Microburst monitoring algorithm inside programmable switches for real-time detection.
INT-DETECT [31]		ML	Grey fault detection and localization method based on INT.
PacketScope [32]		Statistical Analysis	Determine whether abnormal packet loss events occur within network based on INT data.
INT-detector [33]		GAAL	Fast network anomaly detection based on INT and GAAL.
LossSight [34]		GAN	Packet loss complementary based on INT and GAN.
ODS [35]		Clustering	Detection of BGP anomalies based on clustering algorithm.

common network detection means, which mostly acquire network data via SNMP, active network probes, and then perform anomaly detection on the acquired available data through machine learning or some deep learning algorithms. Then, we introduce the methods for network state detection based on network telemetry.

Anomaly detection methods

The purpose of anomaly detection is to find patterns in the data that deviate from other observations [36]. The purpose of using anomaly detection algorithms is to analyze the data obtained from telemetry to monitor the network status instead of network operations and maintenance personnel, to discover network anomalies [37], and even to locate the source of network anomalies [38]. Network information tends to be streaming data that contains a large amount of normal data, and this streaming data changes dynamically over time; most early anomaly detection algorithms were based on supervised learning algorithms, among which the streaming half-space tree [25] (HS-Tree) achieves the classification of normal and anomalous by a decision tree that does not require changing the tree structure. HS-Trees use the quality [39] as a judgment marker for ranking anomalies to achieve fast and accurate anomaly determination. Another way to detect abnormal network states is through outlier detection. The abnormal data latent in the network state data can be represented as outliers. The random subspace hashing algorithm [26] (RSHash) uses random hashing to achieve a fast and stable subspace outlier anomaly state determination effect, and the algorithm also gives a more reasonable anomaly score for outliers. These supervised learning-based methods use labeled data to train algorithm models for the detection of normal and abnormal events. However, the biggest problem with such methods is that the model needs to be trained with a balanced data set constructed. In network anomaly detection, the periods when anomalies occur are few and a balanced data set cannot be constructed. In addition, making labels for the data set for this particular problem of network anomaly detection is also a tricky problem.

With the rapid development of neural networks in recent years, they have been able to do predictions for some complex problems, at this time, some researchers have found graph neural networks to be well suited for the graph-based prediction of network state information and thus discriminating anomalies. Mahmoud et al. in [27] proposed a LSTM AutoEncoder and one-class support vector machine (OC-SVM [40]) based approach to train the model by using only normal class examples. Liu et al. in [28] proposed a graph neural network-based anomaly detection algorithm for industrial control networks that fuses the network nodes' own attributes and the information of neighboring nodes in the network topology to achieve the detection of network anomalies. Deng et al. in [29] proposed a method GDN for anomaly detection in industrial sensor networks based on graph attention mechanism and graph neural network. It improves the accuracy of anomaly detection in multidimensional data by introducing GAT in the time dimension. Most of these neural network-based network anomaly detection methods are based on unsupervised learning (e.g., Hidden Markov Models, K-means clustering [41]) or semi-supervised learning (i.e., training

using only normal data. The trained models are then applied to methods containing both normal and abnormal event test data) neural network algorithms. These algorithms are dedicated to mining the high-dimensional feature relationships in the input data, and discriminating outliers (i.e., abnormal times) by learning the high-dimensional feature change patterns of the input data. Therefore, the performance of neural network-based algorithms depends on both the algorithm's ability to extract high-dimensional features and the amount of information in the input data.

Telemetry-based network anomaly detection

Network measurement means cannot accurately reflect the state changes of the network, that is, they cannot provide rich enough input data for the subsequent discriminant algorithm. The emergence of network telemetry has solved this problem. In-band network telemetry is a typical representative of the new network telemetry technology in recent years, which can accurately query the internal state of the switch and perform fine-grained, real-time monitoring of the network by inserting metadata into each packet through intermediate switching nodes in the path, and embedding the network information into the packet.

Some anomaly detection algorithms based on different telemetry methods have also been proposed in recent years, which focus on faster data processing speed and more accurate anomaly determination for network state information obtained by different telemetry methods. Andrian et al. in [35] proposed a stream pattern anomaly detection algorithm ODS, which is suitable for manipulating telemetry data. Andrian performed an exhaustive evaluation of the available data sets, comparing ODS with classical offline (e.g., DBScan [42], local outlier [43]) and online methods (windowed variants of Robust Random Cut Forest [44], ExactStorm [45], and continuous outlier detection [46]) to validate the reliability and timeliness of ODS. Tan et al. in [34] proposed a packet loss monitoring system for in-band network telemetry. From the incomplete in-band network telemetry data, the lost telemetry information is automatically inferred and filled, and the complete telemetry information is output. Experimental results show that this method has high detection and recovery accuracy and very low overhead, which can further improve network monitoring, control, and management performance. Ross et al. in [32] proposed a network telemetry system PacketScope. This system is also designed around system packet loss, and obtains packet loss information and information such as delay and forwarding queue inside the switch to determine whether abnormal packet loss events have occurred in the current network. Jia et al. in [31] proposed a fast gray fault detection and localization mechanism based on the recently proposed in-band network telemetry. Using INT probe packets for network-wide telemetry, all feasible paths between the source and the target are obtained. However, this method can only identify impassable breaks in the network environment, and cannot effectively identify other network anomalies such as congestion and Microburst. To address the problem, Chen et al. in [21] proposed Snappy, an algorithm that can identify Microburst in real time. Snappy maintains multiple snapshots of queue occupants over time. When each new packet arrives, Snappy updates a snapshot and estimates the score of queue occupancy. However, Snappy's detection of Microburst flows is inherently probabilistic, and the probability (recall) of identifying all Microburst flows increases with the number of switching pipeline phases Snappy uses. Snappy also requires division and rounding operations, which are currently not supported by high-speed programmable switch ASIC. Joshi et al. in [30] proposed BurstRadar to achieve continuous and efficient monitoring of Microburst by capturing telemetry information of only the packets involved in a Microburst using a programmable switch ASIC. However, since it is a switch programming algorithm implemented through P4, it focuses only on Microburst monitoring and cannot perform normal monitoring for other kinds of anomalies. Based on this state of affairs, Zhang et al. in [33] proposed an INT-detector, an automatic and fast network anomaly detection system that combines in-band network telemetry and deep learning to detect anomalies using Generative Adversarial Active Learning. However, this approach is more focused on providing low latency detection speed compared to giving more accurate anomaly detection results.

In summary, according to our latest survey, the current detection methods for handling network telemetry data are still under development. Existing methods are often limited to specific kinds of problems, and there is currently no effective detection algorithm for more widespread network anomalies. Therefore, how to better apply the well-developed neural network-based anomaly detection algorithms to the detection of network anomalies based on telemetry data, to achieve efficient, accurate, and widespread network anomaly detection, remains a key focus that requires further exploration and research.

Framework of telemetry-based anomaly detection system

In this section, a systematic overview of the network anomaly detection problem addressed by MFGAD-INT is presented, and mathematical modeling is performed to formulate the optimization problem. Specifically, in Overall structure of MFGAD-INT section, the input

data of network anomalies processed by MFGAD-INT is defined and corresponding mathematical calculation methods and data acquisition methods are given. In Data preprocessing section, the definition of network anomaly detection is provided, and the mathematical expressions for input and output are given. In Network anomaly detection problem description and construction section, optimization problem analysis is carried out based on the mathematical definitions given in Data preprocessing section.

Overall structure of MFGAD-INT

As shown in Fig. 1, MFGAD-INT is consisted of two main parts: offline training and online testing. The network telemetry module, which is common to both parts, is normalized in the data preprocessing module and further divided into sequences by a sliding window of size N. The telemetry data is divided into segments of multivariate time series for offline training. During offline training, the model learns the metric change patterns during healthy network operation and gives anomaly scores to the network state based on the learned patterns. These scores are used to select appropriate score thresholds. This offline model training session can be included as a regular training schedule to accommodate the network load at different time periods, such as weekly or every other week. For online detection, MFGAD-INT invokes the model to score anomalies on the new telemetry data $X_t = \{x_1, x_2, \ldots, x_m\}$ after pre-processing, and then makes anomaly judgments and alerts by thresholds. When the anomaly score exceeds the threshold, the current moment is evaluated as abnormal and the output $Y_t = \{y_0, y_1, y_2, y_3, y_4, \ldots, y_M\}, y_i \in \{0, 1\}$, where $y_0 = 1$ indicates the presence of anomalies in the current network, followed by $\{y_1, y_2, y_3, y_4, \ldots, y_M\}$, $y_i \in \{0, 1\}$

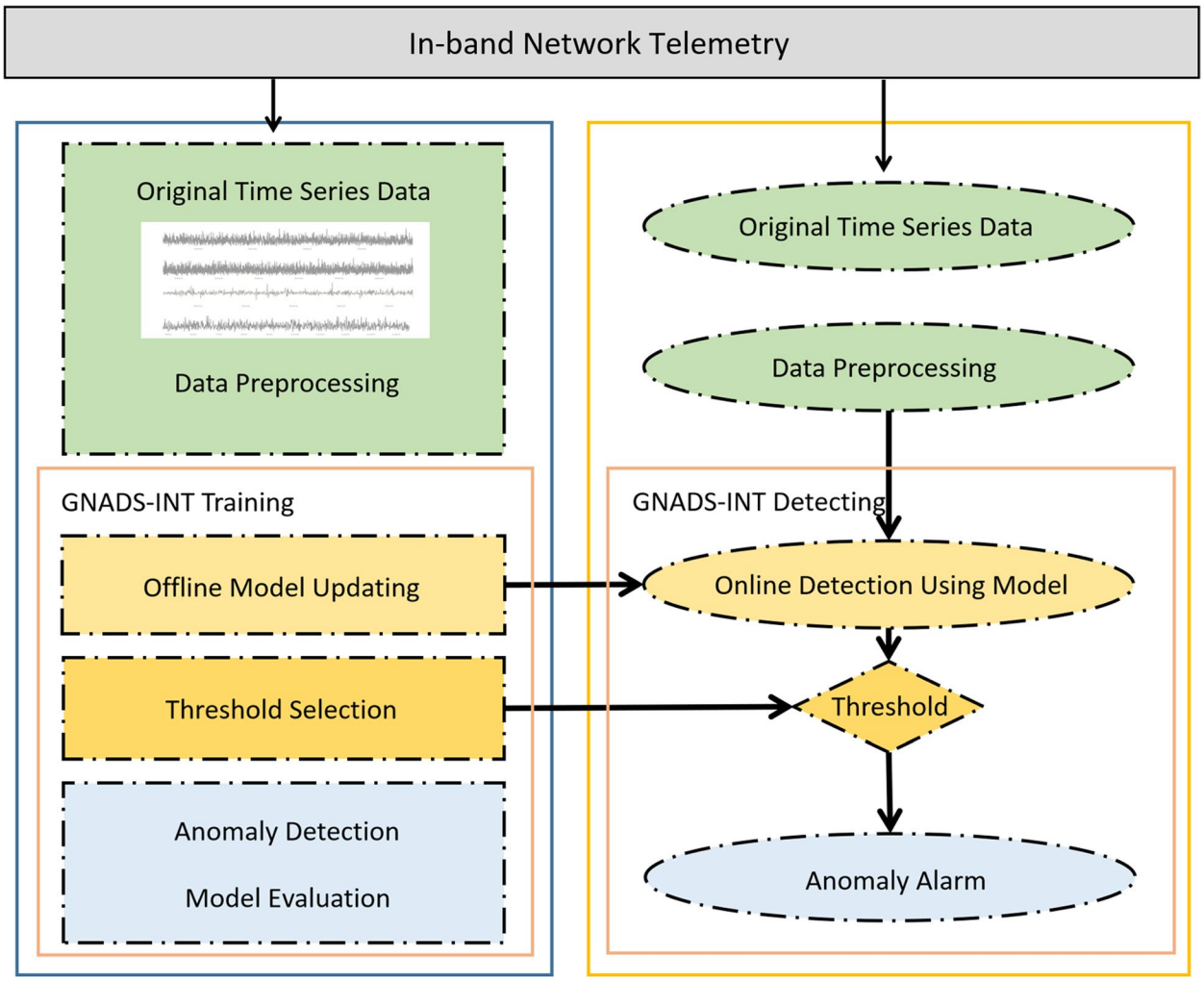

Fig. 1 Overall framework of MFGAD-INT

indicating which specific metric triggered the anomaly. This allows the algorithm to detect not only if there are anomalies in the network at this time, but also which specific metrics are anomalous as a way to explain why the anomaly occurred.

Data preprocessing

MFGAD-INT initially obtains network state metadata through telemetry, and network indicators can be obtained by calculating these metadata. During the computation, some additional categorical metadata (e.g., port number, switch number, etc.) will be cleaned up after being used for categorical data dimensions; some data that do not meet the range of values specified in this section will also be cleaned up and invalidated. Finally, standard normalization is applied to the whole data set to complete the data preprocessing. All the metrics in the network, such as delay, packet loss, queue length, link bandwidth utilization and other key indicators, are telemetry with a fixed frequency.

Switch processing delay

There is a period of time between when a packet enters the switch and when the switch sends it out of the switch, called the switch processing delay X_{nodal}. Theoretically, the switch processing delay $X_{nodal} = t_{process} + t_{queue}$, where $t_{process}$ is the processing delay, which includes the time taken for routing and the time required for checking, and t_{queue} is the queuing delay, whose size depends on the current traffic in the network. There is actually also the time between the start of the packet and its complete delivery called transmission delay, denoted as $t_{transmission}$, but it is negligible as it is often very short. The method of obtaining this value in MFGAD-INT can be simplified by telemetry, and the exact value can be quickly calculated by the timestamps X_{in} and X_{out} of the packets entering and leaving the switch port, which is calculated as

$$X_{nodal} = X_{out} - X_{in}. \tag{1}$$

Link delay

The time from when a packet is sent by the previous switch to when it is received by the next switch is called the link delay, denoted as X_{chain}. The actual value of the link delay is calculated by the difference between the Egress timestamp of the previous switch and the Ingress timestamp of the current switch. The minimum value of link delay should be its theoretical value and is calculated as

$$X_{chain\_min} = \frac{Channel\ length}{Channel\ program\ rate}. \tag{2}$$

Packet loss rate

Generally, packet loss occurs at the switch node and previous means are not very easy to obtain the packet loss rate in the network. Using telemetry, the packet loss number is obtained by calculating the difference between the total number of packets received by the switch and the total number of packets sent to obtain the current packet loss rate X_{drop} at the switch node.

Link bandwidth utilization

Link bandwidth utilization is usually defined as the actual data transfer on a physical link as a percentage of the channel capacity. Using telemetry, the incoming and outgoing port utilization X_{in_used} and X_{out_used} of the switch can be directly obtained, and the port utilization of the same physical port can be summed to obtain the actual bandwidth utilization of the port and thus can refer to the link bandwidth utilization. Therefore, the link bandwidth utilization X_{chain_used} is calculated as

$$X_{chain\_used} = X_{in\_used} + X_{out\_used}. \tag{3}$$

Network anomaly detection problem description and construction

In the network anomaly detection problem, the network indicator information can be regarded as a time series, and the telemetry information of all indicators in the network constitutes a multivariate time series, where each sequence affects each other. By definition, the multivariate time series can be expressed as $X = \{X_1, X_2, X_3, X_4, , , X_M\}$, where M denotes the number of network performance indicators. Each univariate time series $X_n \in R^N$ is a vector representing the data of one network performance metric after N telemetry at a fixed frequency. Therefore, the multivariate time series is finally represented as $X \in R^{M \times N}$. For a given multivariate time series input $X \in R^{M \times N}$, a sliding window of size T is used to generate a fixed length input. Network anomaly detection is used to evaluate the anomalous state of the input at each moment, and this evaluation can be quantified as $Anom_t = \{anom_0, anom_1, anom_2, anom_3, anom_4, , , anom_M\}$, where $anom_0$ is the network state anomaly evaluation and the other parts are the network indicators of each anomaly evaluation. The task of anomaly detection is to generate the output vector $Y \in R^{(M+1) \times T}$ by computing the quantified anomaly evaluation, which can be expressed as $Y = \{Y_1, Y_2, Y_3, Y_4, \ldots, Y_T\}$. where $Y_t = \{y_0, y_1, y_2, y_3, y_4, , , y_M\}, y_i \in \{0, 1\}$, indicates whether the network as a whole is anomalous and whether the network indicator is anomalous at the t-th timestamp.

Based on the above definition, the objective optimization equation can be obtained. The network anomaly detection problem can be carried out in two steps. First, for the prediction of the network state, the input data is predicted by a deep learning model, and the first step is optimization by its loss function. The optimization equation is as

$$\begin{aligned} &\min \frac{1}{n}\sum(\overline{x_i}-x_i)^2, \\ &\text{s.t. } C_1: x_i > X_{chain\_min}, x_i \in X_{chain} = \{x_1, x_2, x_3, x_4, \ldots, , x_T\}, \\ &C_2: x_i < 1\,, x_i \in X_{drop} = \{x_1, x_2, x_3, x_4, \ldots, x_T\}, \end{aligned} \tag{4}$$

where x_i and $\bar{x_i}$ are the original state information and predicted state information for the i-th timestamp in the current input data, respectively. Some indicators in the input data have constraints in the values, and data cleaning is performed when the input values are not in the given range.

When the algorithm starts anomaly detection, the predicted state data needs to be scored based on the true values and the classification algorithm is optimized by the following optimization problem.

$$\begin{aligned} &\max \frac{1}{n}\sum(\overline{x_{anom}}-x)^2 - u, \\ &\text{s.t.} \quad \overline{x_{anom}} \geq 0, x \geq 0, \end{aligned} \tag{5}$$

where x denotes the true state value of the network at the current timestamp and $\overline{x_{anom}}$ is the anomaly score given by the prediction algorithm when the anomaly occurs, respectively. u is the threshold value given by the SPOT algorithm, which is mathematically described as

$$u = SPOT(\overline{x}_l, x_i), \tag{6}$$

For the predictions $\overline{x}_l$ given by deep learning, the anomaly score is calculated by the telemetry value x_i according to a fixed formula SPOT and a suitable threshold is chosen to maximize the difference between the score and the threshold when an anomaly occurs. We will explain the processing process of the SPOT algorithm in detail later.

Method

The anomaly detection methods before are mainly used to generate clusters rather than detecting the anomalies, which is the goal of this paper. In this section, we will show how to move from clustering to outlier detection. Next, the details of the components of the algorithm are described. Finally, the usage steps of the algorithm for online detection are presented.

Detection program design

In our work, we use INT to obtain network status information, and the telemetry data itself can be regarded as a data source containing multimodal information. Each feature in the data set contains the information from different telemetry metadata with its own periodical changes; meanwhile, the correlation between different telemetry metadata constitutes the spatial information of network state; The change pattern presented by the features composed of all telemetry metadata under the same time reflects the temporal information of network state. The CNN and different GAT layers are used to separately extract multimodal information from the same data set. The advantage of separate refinement is that different neural networks can focus on only part of the state information in the data set during the training process to achieve better feature extraction. Combined with the idea of multimodal learning, the fusion of multimodal data is completed in the model. Since the multimodal information of MFGAD-INT is obtained by the algorithm's feature extraction of the same data set, feature fusion can be done simply. The temporal density of INT telemetry data is high, up to millisecond time density, so the long-time memory in the algorithm's input data also retains rich real time patterns, which is crucial for predicting network state information. For the fused multi-state data, the GRU is used to achieve the complementary extraction of long-time information. Finally, the GRU Auto-Encoder is used to compress the multimodal data for prediction. The data dimension is compressed to achieve the prediction of the network state while preserving the high-dimensional feature information as much as possible. After the above algorithm, we chose SPOT threshold selection algorithm to judge the anomaly scores. Summarizing the algorithm composition of MFGAD-INT.

(1) Data feature learning for different states in INT telemetry data is implemented using a 1-D convolution layer and two GAT layers. Realizing the separation of multimodal information based on telemetry data and the generation of multimodal data.
(2) Fusion learning of underlying features of the multimodal data at the shallow level of the model. The multimodal data are stitched together one by one, and feature learning of long-time information is achieved by GRU.
(3) Finally, to predict the network state, an AutoEncoder is used to implement multimodal transitions. In this paper, we use GRU Auto-Encoder to complete this step, which can achieve lower information loss data compression by convolution neural network based on attention mechanism.

The overall flow of the anomaly detection algorithm is shown in Fig. 2. In the remainder of this subsection, the

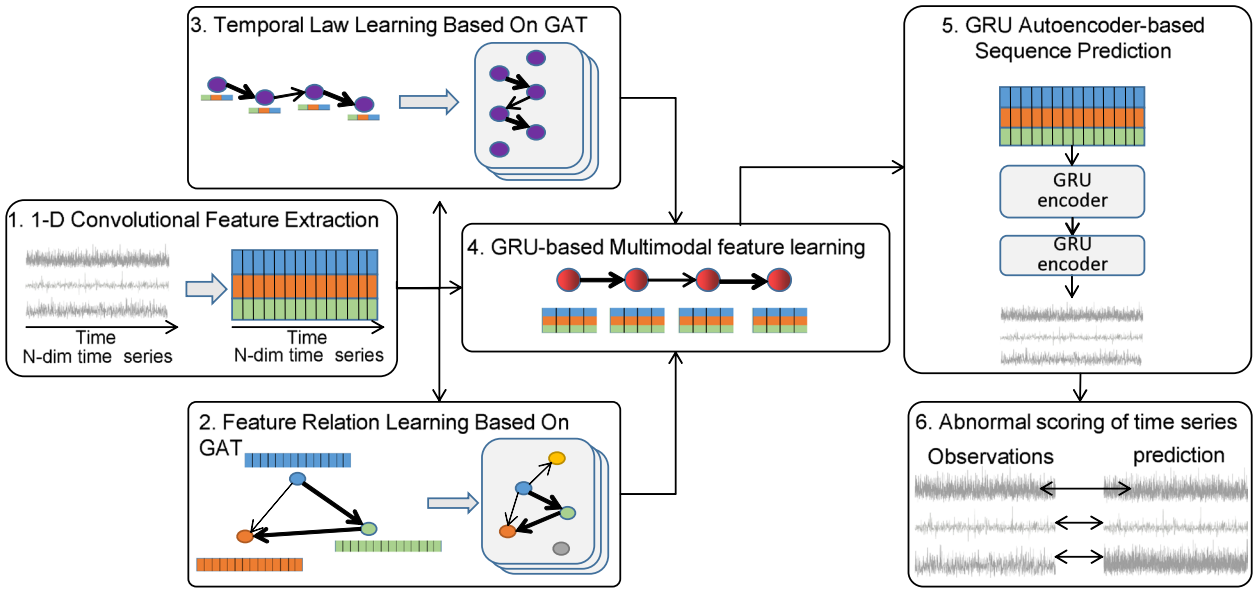

Fig. 2 General structure of multivariate timing prediction model

principles and roles of each component in the prediction algorithm are further elaborated.

The use of graph attention

In a real network environment, the values of each network metric are not independent, and the variation patterns among them affect each other. The changes of the same metrics at different times in the same network environment should also have regularity. So how to make the algorithm discover and quantify these existing multimodal patterns is the focus of our concern.

GAT can be the key to solve this problem. GAT is proposed to discover the influence relationship between connected nodes in a graph. For our problem, two graphs can be constructed for the input data according to the time dimension and the spatial dimension. In the spatial dimension, each univariate time series $X_i \in R^T$ is considered as a node to construct a feature graph of T-dimensional feature vectors; in the temporal dimension, the indicator data under the same time is considered as a node to construct a feature graph of M-dimensional feature vectors. Since the network indicators affect and depend on each other, and also show strong correlation between the time stamps of network activities in a short period of time, both of these graphs are complete graphs. GAT calculates the attention coefficients and updates the node feature vectors based on the neighbor nodes X_j of each node X_i. The output is updated as

$$h_i = \text{sigmiod}(\sum_{1}^{N} \alpha_{ij} X_j), \tag{7}$$

where h_i denotes the node feature vector updated by GAT, N denotes the number of neighboring nodes of v_i, v_j is the feature vector of neighboring nodes, and α_{ij} is the attention score of v_j to v_i, which is used to represent the correlation between two nodes and further used to update the node feature vector. The attention score α_{ij} can be calculated by

$$e_{ij} = \text{LeakerReLU}(\omega^T \cdot (X_i \oplus X_j)), \tag{8}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{n=1}^{N} exp(e_{in})}, \tag{9}$$

where ω^T is a learnable row vector parameter and $\oplus$ denotes the Hadamard product of two feature vectors. LeakyReLU is a nonlinear mapping activation function used to add a nonlinear mapping to the model.

As described above, GAT is applied to learn the two relations and achieve the generation of multimodal data. After the multimodal data is generated, the outputs of the two GAT layers $h_{space} = \{h_1, h_2, h_3, h_4, , , h_M\}$, $h_{time} = \{h_1, h_2, h_3, h_4, , , h_T\}$ are spliced with the original input $X = \{X_1, X_2, X_3, X_4, , , X_M\}$ to achieve the multimodal data The splicing formula is as

$$h = h_{space} + h_{time}^T + X. \tag{10}$$

The use of GRU

By using GAT, we have achieved the extraction and fusion of multimodal information, and the next step is to learn the long-term memory information hidden in it. In general, this can be achieved by RNNs. However, standard RNN networks are not good at handling long time sequence data due to the gradient vanishing problem. Therefore, in this paper, GRU is chosen to capture the long-time memory information in multimodal data.

GRU selectively retains and uses long-time memory by introducing RNN-based update and reset gates to solve the gradient disappearance problem of long-time sequences. The update gate is calculated by

$$z = sigmoid(h_t U^Z + s_{t-1} W^Z), \tag{11}$$

where h_t is the input for the current timestamp t, s_{t-1} is the retained information for the previous timestamp $t-1$, and U^Z, W^Z are two weight matrices, which are used to make a linear change to the input information. The update gate determine how much of the historical information needs to be retained for further transmission. The reset gate is calculated by the following equation

$$r = sigmoid(h_t U^r + s_{t-1} W^r). \tag{12}$$

As with the update gate, the input to the reset gate is linearly varied and then the activation result is compressed using the Sigmoid function. The role of the reset gate is to determine how much historical information should be forgotten, so new memory content will use the reset gate to store past relevant information. The formula for this step is

$$h = tanh(h_t U^h + (s_{t-1} \oplus r) W^r), \tag{13}$$

where the meanings of h_t, s_{t-1}, U^h and W^h remain unchanged and r is the activation result of the reset gate. The Hadamard product of s_{t-1} and r can be calculated to determine the previous information to be retained versus forgotten. Finally, the input x_t is activated using hyperbolic tangent activation function after linear transformation with the determined historical information vector respectively.

The network also needs to retain the information about the current timestamp to be passed, and this is where the result of the update gate is used. This step can be expressed as

$$s_t = (1-z) \oplus h + z \oplus s_{t-1}. \tag{14}$$

The Hadamard product of z and s_{t-1} represent the information retained in the previous step, and this information plus the information retained in the current memory is the output of GRU.

The above algorithmic flow completes the learning of the multimodal data. The output is put into the AutoEncoder and compressed to obtain the prediction result, denoted as $\bar{X} = \{\bar{x_1}, \bar{x_2}, \bar{x_3}, \bar{x_4}, , , \bar{x_M}\}$. Here we choose GRU Auto-Encoder. This prediction result will be used as an important component in the calculation of the anomaly score, which is described in detail in the next subsection.

Anomaly score and SPOT threshold selection

After the prediction results are obtained, further abnormal scores are needed. For the anomaly scores of individual indicators, the normal state score and the anomaly score need to be maximized, and each score is guaranteed to be greater than zero. Equation (15) is used to evaluate the anomalies of the network metrics at the specified timestamps.

$$anom_i = (\bar{x}_i - x_i)^2, \tag{15}$$

where $\bar{x_i}$ is the predicted value and x_i is the real network indicator state value observed through network telemetry. The network anomaly score can be obtained by averaging the anomaly scores of all indicators, calculated by

$$anom_0 = \frac{\sum_{i=1}^{n} s_i}{n}, \tag{16}$$

where n is the number of all indicators obtained by telemetry, and the final total output $Anom_t = \{anom_0, anom_1, anom_2, anom_3, anom_4, , , anom_M\}$ is obtained. By averaging, it ensures that the abnormal state of each indicator is reflected when determining the abnormal state of the network; it also makes it possible that when the number of network indicators is large, there is no problem of scoring too large values for network abnormalities.

Finally, anomaly judgments are made based on anomaly score pairs. Here the threshold value can be used to judge the network state. Since the state of the network fluctuates more frequently, some networks have small values of change from normal to abnormal states. Therefore, the common SVM dichotomous threshold selection method is not suitable for this algorithm. MFGAD-INT uses SPOT algorithm to dynamically generate thresholds that determine anomalies in network state and indicators, obtaining the final system judgement output $Y_t = \{y_0, y_1, y_2, y_3, y_4, , , y_T\}, y_i \in \{0, 1\}$.

```
A ← ∅
z_q, t ← POT(anom_n, ..., anom_n, q)
k ← n
for i > n do
    if anom_i > z_q then
        Add(i, anom_i) in A
    else if anom_i > t then
        X_i ← anom_i − t
        Add X_i in X_T
        N_t ← N_t + 1
        k ← k + 1
        γ̂, σ̂ ← Grimshaw(X_T)
        z_q ← CalcThreshold(q, γ̂, σ̂, k, N_t, t)
    else
        k ← k + 1
    end if
end for
```

Algorithm 1 Calculate SPOTFor the fraction $anom_i$ of all anomaly samples, the initial threshold z_q is calculated by POT, after which the outlier data is traversed to determine whether the current outlier value exceeds this threshold, and if it does, it is judged to be abnormal and added to A. If it does not exceed, the current data is judged to be a peak, and if it is a peak, the data values exceeding the current peak are added to the set X_T used to store the anomaly peak, calculate the optimization parameters $\hat{\gamma}$, $\hat{\sigma}$, and finally update the threshold value.

The pseudocode of the SPOT algorithm is shown above. Here, $anom_i$ is the network anomaly value predicted by the prediction model and scored, while A is the data set used to store the data judged as anomalies in the algorithm. The initial threshold z_q is obtained through executing POT operator. Then, the algorithm traverses the anomaly value data and determines whether the current anomaly value exceeds the threshold. If it exceeds, it is judged as an anomaly. At this point, it is also necessary to determine whether the current data is a peak. If it is a peak, the values exceeding the current peak are added to the set X_T which is used to store the excess peaks, and the optimization parameters $\hat{\gamma}$ and $\hat{\sigma}$ are calculated, and finally the threshold is updated.

Simulation and results analysis

In this section, we evaluate the proposed MFGAD-INT using data sets collected in a real data center network environment. We first present the experimental setup, including the simulation environment, data collection approach, and the metrics and compared methods chosen for performance evaluation. The results of different are then analyzed to validate the advancement of MFGAD-INT.

Experimental setup

The data set was collected from a testbed replicating the legacy topology of the CSP data center, as shown in Fig. 3.

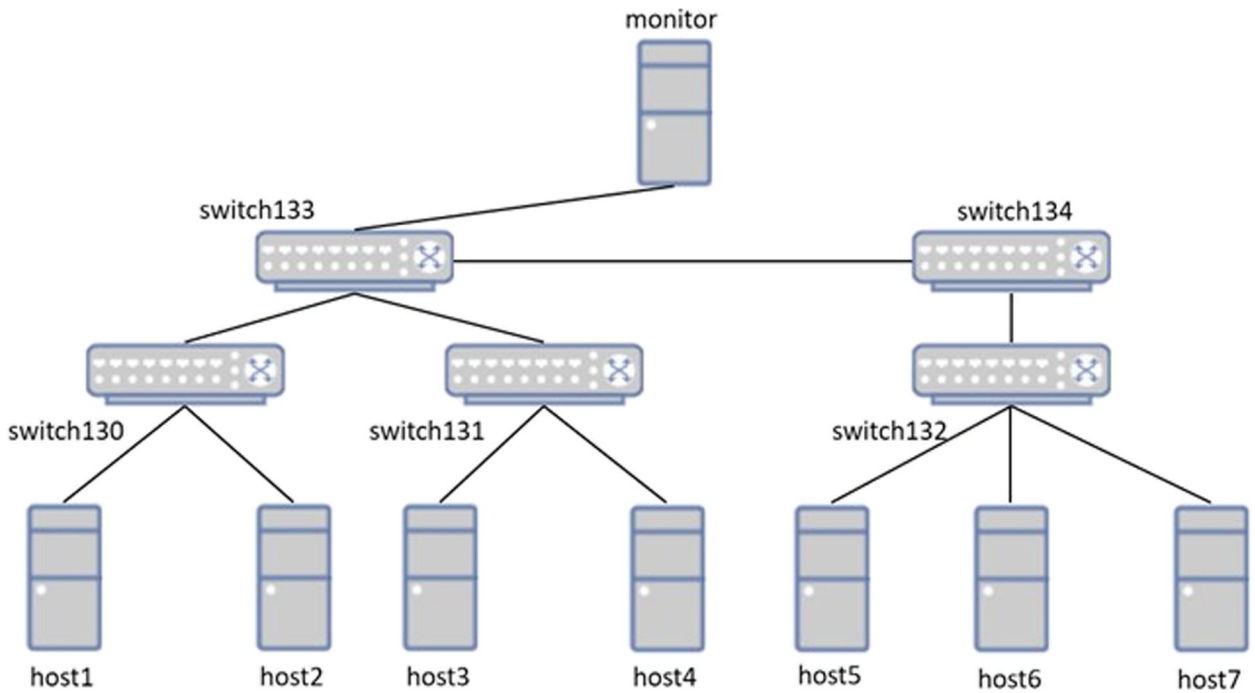

Fig. 3 Real testbed network topology

Although the testbed does not involve real users, it does use real devices, protocols, and applications typical of production networks. We use a server to generate real application data streams and are controlled to add random anomalous events to the streams. In the context of the study, we chose to inject both Microburst, which occur more commonly in modern data centers, and elephant flow anomalies.

Modern data center networks operate at high speeds (bigger than 10 Gbps) and have ultra-low end-to-end latency (10 microseconds) [30]. As a result, even a small amount of queuing (called Microburst) occurring over a short period of time can have a significant impact on application performance. This manifests itself in the form of a very large number of bursts of data received by switch ports in a very short period of time (millisecond level). We generate Microburst randomly in the network environment by sending short bursts of high traffic data streams at random.

Anomalous occurrences of large elephant flows can also significantly impact network performance and thus trigger network anomalies. However, if a service that generates a stream is misconfigured and enters the normal network environment, it may cause severe congestion in the network environment. In this paper, anomalous events are injected by randomly sending elephant stream data among different servers.

Different anomalous events are selected for injection under different network load conditions, resulting in multiple experiments as shown in Table 2 below.

As to the data set, we first collect data under normal conditions as the training set and thereafter collect the test set based on the injection of anomaly events as labels. We measure high and low traffic loads in terms of the number of running services. Specifically, the services in the network are simulated by pushing video streams. At the same time, specific anomalous service streams are randomly injected into the network environment using idle servers to keep the occurrence of network anomalies in a human-controllable state. All experiments lasted for about half an hour. To verify the effectiveness of INT, data sets were collected for each group of experiments using the SNMP network protocol and different telemetry frequencies of INT.

Based on the latest INT 2.1 protocol, we select nine of these standard metadata as shown in Table 3. These telemetry metadata can be uniquely identified by the name of their YANG model. According to the specification of the protocol, the selected metadata can be classified into 4 categories, i.e., node information, ingress information, egress information, buffer information, which are all data plane information.

Evaluation Metrics: Since the data set for anomaly detection is an unbalanced data set [47], i.e., most is normal data while only a little is anomaly data. In this case, if we only focus at the correct rate, we can easily imagine a scenario where 90% of the test data set is normal, when the predicted results are all normal, and we can expect the correct rate is about 90%. But in fact, this kind of metrics is not meaningful for the anomaly detection problem which is more concerned with the low-probability anomalies. Therefore, we use Accuracy, Recall and F1 to evaluate the performance of the algorithm model.

We compare MFGAD-INT with other network anomaly detection methods, including ODS, GDN, HSTree, and RShash. all of these algorithms are described in the subsection on related work, and they are representative of typical algorithms at various stages of the anomaly detection field.

We implemented our approach using Python version 3.8. All experiments were run on a server with Intel Core i9-12900K Processor (5.20 GHz) CPU and NVIDIA GeForce RTX 3090. We use the same sliding window size of 100 for all models. In MFGAD-INT, we set the hidden size of GRU and GRU AutoEncoder to 150 and use the Adam optimizer to train the MFGAD-INT model. The

Table 2 Description of the experimental data set

No.	Traffic	Anomaly Events	Duration
E1	High load	Microburst	0.30h
E2	Low load	Microburst	0.35h
E3	High load	Elephant Flow	0.30h
E4	Low load	Elephant Flow	0.35h
E5	High load	Both	0.30h
E6	Low load	Both	0.35h

Table 3 Available INT telemetry features

Node	Ingress	Egress	Buffer Information
Node-id	Ingress-identifier	Egress-identifier	Queue-id
	Ingress-timestamp	Egress-timestamp	
	Ingress-RX-byte-count	Egress-TX-byte-count	Instantaneous-queue-length
	Ingres-RX-utilization	Egress-TX-utilization	

initial learning rate is 0.001 and the number of training epochs is 50. For each data set, we independently conducted 5 repeated experiments.

In addition, telemetry-based means of data collection is an important component of MFGAD-INT. To analyze the effectiveness of this component, for MFGAD-INT, additional comparative experiments were performed using SNMP-based data sets obtained.

Results and analysis

In this subsection, the experimental results are analyzed to verify the validity and reliability of MFGAD-INT. First, the impact of the anomaly detection effect on telemetry parameters will be discussed mainly. Then, the performance of the proposed algorithm is compared with other anomaly detection algorithms. Also the generalization and reliability of the proposed algorithm is verified by the performance differences under multiple data sets. Finally, an anomaly case is analyzed to show that MFGAD-INT has the ability to locate the location of anomalies at the same time to a certain extent.

Influence of telemetry parameters on the effect of anomaly detection

To demonstrate the important role played by telemetry components in MFGAD-INT, a component validity analysis was performed. The data set was reconstructed in the same network environment using SNMP. Also, to test the performance difference of MFGAD-INT under different frequency telemetry, we also collected network information in this environment using 25 Hz, 15 Hz, and 5 Hz telemetry frequencies respectively. Since SNMP and low-frequency telemetry methods inevitably suffer from missing information, the mean substitution method is used to supplement the missing state information. The performance comparison is shown in Table 4.

Table 4 Performance comparison of algorithmic models

Method	Precision	Recall	F1
SNMP	0.5964	0.6274	0.6115
5Hz INT	0.8895	0.7399	0.8079
15Hz INT	0.9469	0.7624	0.8447
25Hz INT	0.9779	0.8987	0.9366
50Hz INT	0.9886	0.9974	0.9942

The following conclusions can be drawn from analyzing the data in the above table.

First, observing the data in Table 4 with and without INT telemetry shows that all metrics of the algorithm model are much higher than SNMP when telemetry is used. Focus on Recall, it can be seen that the full power INT component can detect basically everything that should be detected and 99.74% of the abnormal time periods can be detected by the algorithm. Also observing the five sets of performance comparisons it can be seen that faster frequency telemetry corresponds to higher performance. This is because telemetry brings more available information and MFGAD-INT is designed to handle high density data.

With the addition of telemetry, the accuracy of the network state index assessment given by the algorithm is significantly improved. Many network anomalies of short duration can be reasoned and identified by the algorithm, as shown in the comparison Fig. 4. Figure 4(a) and (b) show the impact of anomaly detection on the processing delay of the same switch under different network detection means, and the part above the dotted line shows the index anomalies detected by the algorithm. In this example, there is one elephant flow anomaly and 20 Microburst in the network. The algorithm without telemetry detects only one elephant flow anomaly and five short-time Microburst, and the anomaly detection results lag in elephant flow anomaly detection; while the algorithm based on the INT telemetry component completes the identification of all anomalies without lag in the time period of anomaly detection, which can quickly provide more time-sensitive detection results.

In summary, compared with SNMP, telemetry can enrich the temporal information in the acquired network state information, thus maximizing the performance of MFGAD-INT and enabling high-performance and highly interpretable network anomaly determination.

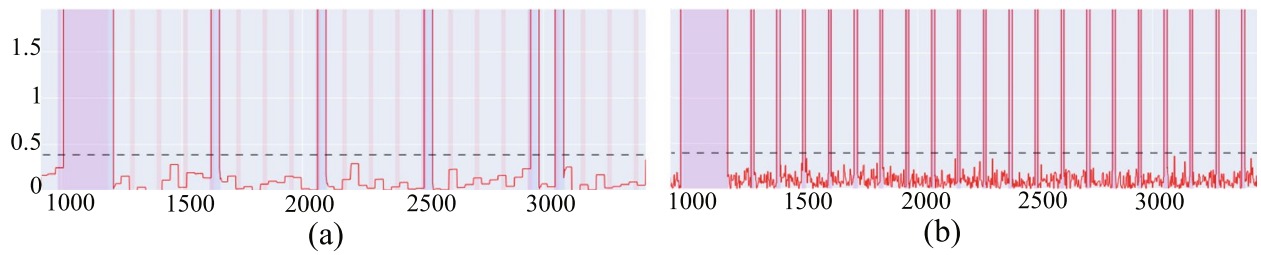

Fig. 4 Anomaly scores with and without telemetry components, **a** without telemetry **b** with telemetry

Table 5 Performance comparison of model and baseline

Method	Precision	Recall	F1
RShash	0.7614(0.4444)	0.9999(0.4880)	0.8645(0.4652)
HSTree	0.5475(0.4030)	0.9993(0.4759)	0.7074(0.4367)
ODS	0.8263(0.4967)	0.9999(0.4987)	0.8969(0.4976)
GDN	0.8375(0.5023)	0.9418(0.4965)	0.8866(0.4993)
MFGAD-INT	0.9886(0.5964)	0.9974(0.6274)	0.9942(0.6115)

Meanwhile, the frequency of telemetry also significantly affects the performance of the algorithm, and a higher telemetry frequency will bring more accurate detection results, which is consistent with intuition.

Validity of MFGAD-INT

As shown in Table 5, MFGAD-INT performs significantly better than other network anomaly detection algorithms. MFGAD-INT improves about 10.56% compared to GDN and about 9.73% compared to ODS. HSTree and RShash are more early period anomaly detection methods, which are not as effective when dealing with fine-grained, highly oscillatory network indicators with high time-density data. GDN also uses GAT to learn the correlation of temporal features, but it is also less effective than MFGAD-INT in terms of detection because it mainly learns the temporal patterns hidden in the time series and is not as explanatory for the interrelationships between network indicators.

Figure 5 shows the anomaly score comparison results of the four algorithms. The experimental results in Fig. 5(a) show that RShash is an algorithm that is sensitive to the increase of data anomalies. However, for dynamic network states, the fluctuation of network indicators within a certain range does not imply the presence of anomalies in the network. In addition, when the network operation state reaches an extremely high load, the network state has changed to an abnormal state, but the values of some network indicators do not change drastically, and there are also cases of missed detection. Meanwhile, when the network enters a longer period of abnormal state, RShash will consider that the current state tends to be normal, which is obviously unreasonable. The effect of HSTree has been greatly improved compared with RShash. When facing a long period of abnormal state, the algorithm can accurately determine the current abnormal situation. However, the algorithm cannot give an accurate judgment for the current state when the anomaly has just ended, which leads to many false alarms and affects the performance of the algorithm.

GDN and MFGAD-INT are based on GAT learning data, so they both achieve better results than earlier network anomaly detection algorithms when scoring anomalies on network states. The advantages of the new method incorporating GAT can be seen through Fig. 5(c) and (d). The anomaly score can be stabilized at a high level when facing the same type of persistent anomaly states, while there are more significant differences in the anomaly score values when facing different anomaly states. This indicates that this algorithm enhances the interpretability of GAT for anomaly judgment. At the same time, when the network is in a normal state, the anomaly score can also be kept at a lower state with less fluctuation, which is beneficial for the algorithm to distinguish between anomalies and normal states.

Since the ODS algorithm achieves the detection of anomalies through clustering, it is different from other algorithms in terms of judgment methods. However, it can also be seen from Table 5 that the algorithm does not perform as well as the MFGAD-INT algorithm using GNN. This is a performance-for-time detection algorithm when dealing with larger amounts

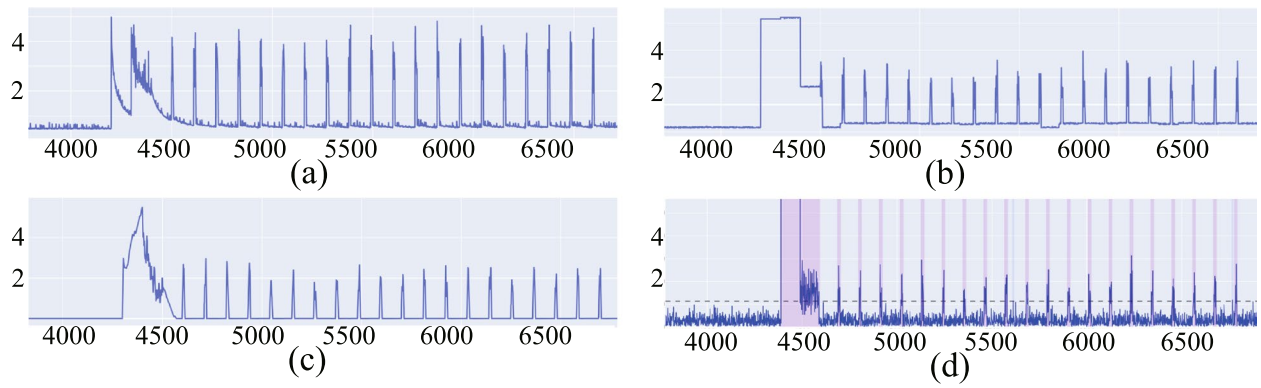

Fig. 5 Algorithm anomaly scores of different algorithms, **a** RShash, **b** HSTree, **c** GDN, **d** MFGAD-INT

of data, as the clustering algorithm dynamically adjusts the central clusters.

Figure 5(c) and (d) also show that MFGAD-INT based on multimodal information extraction learning performs much better than GDN with only spatial dimensional feature learning in terms of interpretability of anomaly scoring. For the same type of anomaly, MFGAD-INT can give similar anomaly score, and for different types of anomalies, MFGAD-INT's anomaly scores also have enough variability. This is not achievable in the other algorithms.

Generalizability of MFGAD-INT

In this subsection the generalizability of the proposed algorithm is analyzed. Through a side-by-side comparison with other algorithms using the same experimental scenarios, the volatility of the MFGAD-INT algorithm is lowest for different anomaly injection methods under different loads, and MFGAD-INT maintains excellent detection performance under different network states. The experimental comparison results are shown in Table 6 below.

The performance of the anomaly detection algorithms fluctuates with the difficulty of injecting anomalies through the above experimental analysis. In the E3/E4 task, the performance of all algorithms is optimal. In the E1/E2 task, MFGAD-INT is the only algorithm with all performance metrics above 0.9. Meanwhile, MFGAD-INT achieves the highest value in all five metrics of all experiments. Although it has a lower recall than RShash and HSTree in the face of E5/E6, these two algorithms are far below MFGAD-INT in the evaluation of other metrics, so MFGAD-INT proves to be highly reliable. Since the ODS algorithm is a clustering algorithm, its detection performance is better in the face of single anomaly injection, but the performance fluctuates greatly when facing multiple anomalies injected at the same time. In contrast, MFGAD-INT performs well in the face of different situations, which proves that MFGAD-INT has good generality.

Table 6 Experimental validation table for generalizability

No.	Method	Precision	Recall	F1
E3/E4	MFGAD-INT	0.9943	0.9995	0.9971
	GDN	0.9053	0.9949	0.9504
	ODS	0.9547	0.9991	0.9767
	RShash	0.7339	0.9508	0.8322
	HSTree	0.6666	0.9994	0.7999
E5/E6	MFGAD-INT	0.9886	0.9974	0.9942
	GDN	0.8375	0.9418	0.8866
	ODS	0.8263	0.9999	0.8969
	RShash	0.7614	0.9999	0.8645
	HSTree	0.5475	0.9993	0.7074
E1/E2	MFGAD-INT	0.9496	0.9999	0.9741
	GDN	0.7893	0.9939	0.8822
	ODS	0.8992	0.9999	0.9469
	RShash	0.6606	0.8727	0.7520
	HSTree	0.3788	0.9593	0.5432

Case study of anomaly location

Since the MFGAD-INT pair achieves more information extraction and thus gives a more explanatory anomaly judgment, it can locate the location and start time of the anomaly to a certain extent. The example is detailed in Fig. 6.

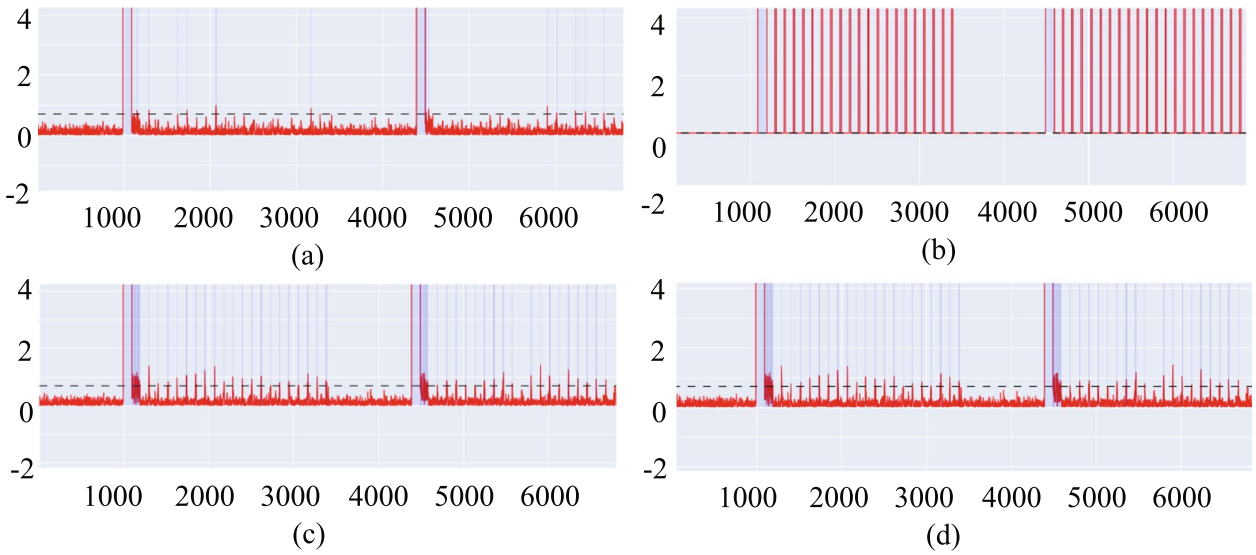

Fig. 6 Anomaly score of internal processing delay of switches **a** Switch 130, **b** Switch 131, **c** Switch 132, and **d** Switch 133

In the example in Fig. 6, there is an elephant flow in the network environment that flows through the switches 130, 132, and 133 during the time periods from 1000 to 1100. The four graphs correspond to the processing delays of each of the four switches in the experimental network, and it can be seen that no anomalies were detected at the time period 1000 in switch 131, while the other four switches all detected persistent anomalies, thus also locating a network link and achieving some degree of anomaly location.

Conclusion

In this paper, we present a graph attention-based deep learning method for INT data-driven anomaly detection in cloud data center networks. Our proposed method, MFGAD-INT, can accurately identify anomalies and locates them within the network. Through experiments on a real cloud platform, we investigate the efficacy and accuracy of the proposed method, and compare it with existing network anomaly detection algorithms to verify its superiority. In the future, we plan to improve cloud service quality by combining knowledge to explain the root causes of network anomalies and automating the intelligent classification of cloud data center network anomalies.

Authors' contributions

Yunfeng Duan proposed the main idea and principles of the research and sketched the manuscript. Chenxu Li designed and implemented the algorithms and experiment schemes and drafted the technical part. Fanqin Zhou guided the design of the algorithms and experiment, prepared the final manuscript for submission. Hao Sun helped with setting up experiment environment, including the illustrative figures. Jiaxing Chen helped with the implementation of the experiments and prepared the analytical figures. Guo Chen prepared the background and related work parts of the manuscript. Chun Zhang drafted the background and related research part of the manuscript. Zehua Gao refined the whole text of the manuscript, and help with preparing the final manuscript for submission. All the authors reviewed the manuscript.

Yunfeng Duan and Chenxu Li has equal and important contribution to the research, so we list them as first author with 'equal contribution'.

Funding

This work is supported by the Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2022) and CMCC and BUPT cooperative program (Grant No. A2022256).

Availability of data and materials

Not applicable.

Declarations

Ethics approval and consent to participate

Not applicable.

Competing interests

The authors declare no competing interests.

Received: 6 April 2023 Accepted: 23 July 2023
Published online: 28 August 2023

References

1. He Q, Dong Z, Chen F, Deng S et al (2022) Pyramid: Enabling hierarchical neural networks with edge computing. In: Proceedings of the ACM Web Conference 2022. Association for Computing Machinery. New York, NY, USA. pp 1860–1870
2. Zhou X, Xu X, Liang W et al (2021) Deep-learning-enhanced multitarget detection for end-edge-cloud surveillance in smart IoT. IEEE Internet Things J 8(16):12588–12596
3. Krishnan P, Jain K, Aldweesh A, Prabu P, Buyya R (2023) Openstackdp: a scalable network security framework for SDN-based OpenStack cloud infrastructure. J Cloud Comput 12(1):26–26
4. Yuan L, He Q, Chen F, Zhang J, Qi L, Xu X, Xiang Y, Yang Y (2021) CSEdge: Enabling collaborative edge storage for multi-access edge computing based on blockchain. IEEE Trans Parallel Distrib Syst 33(8):1873–1887
5. Xia X, Chen F, He Q, Grundy J, Abdelrazek M, Jin H (2020) Online collaborative data caching in edge computing. IEEE Trans Parallel Distrib Syst 32(2):281–294
6. Zhou X, Yang X et al (2021) Energy-efficient smart routing based on link correlation mining for wireless edge computing in Iot. IEEE Internet Things J 9(16):14988–14997
7. Dai H, Yu J, Li M, Wang W, Liu AX, et al (2022) Bloom filter with noisy coding framework for multi-set membership testing. IEEE Trans Knowl Data Eng 35(7):6710–6724
8. Wu S, Shen S, Xu X, et al (2022) Popularity-aware and diverse web APIs recommendation based on correlation graph. IEEE Trans Comput Soc Systems 10(2):771–782
9. Qi L, Lin W, Zhang X, et al (2022) A correlation graph based approach for personalized and compatible web APIs recommendation in mobile APP development. IEEE Trans Knowl Data Eng 35(6):5444–5457
10. Jia Y, Liu B, Dou W et al (2022) CroApp: a CNN-based resource optimization approach in edge computing environment. IEEE Trans Ind Inform 18(9):6300–6307
11. Sulaiman Alhaidari MA Ali Alharbi, et al (2019) Network traffic anomaly detection based on Viterbi algorithm using SNMP MIB data. In: Proceedings of the 2019 3rd International Conference on Information System and Data Mining. Association for Computing Machinery. New York, NY, USA. pp 92–97
12. Smieško J, Kontšek M, Hajtmanek R (2021) Anomaly recognition in bursty IP traffic models. In: 2021 19th International Conference on Emerging eLearning Technologies and Applications (ICETA). pp 351–358
13. Tang J, Chen M, Chen H, Zhao S, Huang Y (2023) A new dynamic security defense system based on TCP_REPAIR and deep learning. J Cloud Comput 12(1):21–21
14. Yang Y, Yang X, Heidari M, et al (2022) Astream: Data-stream-driven scalable anomaly detection with accuracy guarantee in IIoT environment. IEEE Transactions on Network Science and Engineering. pp 1–1
15. Zhou X, Liang W, Yan K, et al (2022) Edge enabled two-stage scheduling based on deep reinforcement learning for Internet of everything. IEEE Internet of Things Journal 10(4):3295–3304
16. Kong L, Wang L, Gong W, Yan C, Duan Y, Qi L (2021) LSH-aware multitype health data prediction with privacy preservation in edge environment. World Wide Web. Kluwer Academic Publishers. USA. 25(5):1793–1808.
17. Wang F, Li G, Wang Y, et al (2022) Privacy-aware traffic flow prediction based on multi-party sensor data with zero trust in smart city. ACM Trans Internet Technol (TOIT). Association for Computing Machinery. New York, NY, USA. online (just accepted):1533–5399
18. Veličković P, Cucurull, et al (2017) Graph attention networks. In: Proceedings of the 6th International Conference on Learning Representations({ICLR}). OpenReview.net. Vancouver, BC, Canada. pp 1–12
19. Li Z, Xu X, Hang T, et al (2022) A knowledge-driven anomaly detection framework for social production system. IEEE Trans Comput Soc Systems. early access (2022):1–14

20. Cho K, Van Merriënboer B, Gulcehre, et al (2014) Learning phrase representations using RNN Encoder-Decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing ({EMNLP}). Association for Computational Linguistics. Doha, Qatar. pp 1724–1734
21. Chen X, Feibish SL, et al (2018) Catching the microburst culprits with snappy. In: Proceedings of the Afternoon Workshop on Self-Driving Networks. Association for Computing Machinery. New York, NY, USA. pp 22–28
22. de Almeida LC, Pasquini R, Verdi FL (2021) Using machine learning and in-band network telemetry for service metrics estimation. In: 2021 IEEE 10th International Conference on Cloud Networking (CloudNet). IEEE. Cookeville, TN, USA. pp 33–39
23. Xu X, Gu J, Yan H, et al (2022) Reputation-aware supplier assessment for blockchain-enabled supply chain in Industry 4.0. IEEE Transactions on Industrial Informatics. 19(4):5485–5494
24. Zhang H, Wang D, Zhang W, Tan L, Kibalya G, Zhang P, Igorevich KK (2023) QoS prediction in intelligent edge computing based on feature learning. J Cloud Comput 12(1):1–16
25. Tan SC, Ting KM, Liu TF (2011) Fast anomaly detection for streaming data. In: Twenty-second international joint conference on artificial intelligence. AAAI Press. Barcelona, Catalonia, Spain. pp 1511–1516
26. Sathe S, Aggarwal CC (2016) Subspace outlier detection in linear time with randomized hashing. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE. Piscataway, NJ. pp 459–468
27. Said Elsayed M, Le-Khac NA, et al (2020) Network anomaly detection using LSTM based autoencoder. In: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks. Association for Computing Machinery. New York, NY, USA. pp 37–45
28. Liu J, Li X (2020) Anomaly detection algorithm for industrial control networks based on graph neural networks. Comput Syst Appl 29:234–238
29. Deng A, Hooi B (2021) Graph neural network-based anomaly detection in multivariate time series. In: Proceedings of the AAAI conference on artificial intelligence. AAAI Press, Palo Alto, California USA. pp 4027–4035
30. Joshi R, Qu T, Chan MC, Leong B, Loo BT (2018) Burstradar: Practical real-time microburst monitoring for data center networks. In: Proceedings of the 9th Asia-Pacific Workshop on Systems. Association for Computing Machinery. New York, NY, USA. pp 1–8
31. Teixeira R, Harrison R, Gupta A, Rexford J (2020) Packetscope: Monitoring the packet lifecycle inside a switch. In: Proceedings of the Symposium on SDN Research. Association for Computing Machinery. New York, NY, USA. pp 76–82
32. Jia C, Pan T, Bian Z, et al (2020) Rapid detection and localization of gray failures in data centers via in-band network telemetry. In: NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium. Budapest, Hungary. pp 1–9
33. Zhang Y, Pan T, Zheng Y et al (2021) Automating rapid network anomaly detection with in-band network telemetry. IEEE Netw Lett 4(1):39–42
34. Tan L, Su W, Zhang W et al (2021) A packet loss monitoring system for in-band network telemetry: detection, localization, diagnosis and recovery. IEEE Trans Netw Serv Manag 18(4):4151–4168
35. Putina A, Rossi D (2020) Online anomaly detection leveraging stream-based clustering and real-time telemetry. IEEE Trans Netw Serv Manag 18(1):839–854
36. Wang R, Nie K, Chang, et al (2020) Deep learning for anomaly detection. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery. New York, NY, USA. pp 3569–3570
37. Qi L, Yang Y, Zhou X, et al (2021) Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure Industry 4.0. IEEE Trans Ind Inf 18(9):6503–6511
38. Ramirez JM, Rojo P, et al (2022) Cleaning matters! preprocessing-enhanced anomaly detection and classification in mobile networks. In: 2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet). IEEE. Piscataway, NJ. pp 103–112
39. Ting KM, Zhou GT, Liu, et al (2010) Mass estimation and its applications. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. Association for Computing Machinery. New York, NY, USA. pp 989–998
40. Xu L, Xu Z (2020) One-class classification with deep adversarial learning. In: Proceedings of the 2019 3rd International Conference on Computer Science and Artificial Intelligence. Association for Computing Machinery. New York, NY, USA. pp 103–106
41. Burnaev E, Ishimtsev V (2016) Conformalized density- and distance-based anomaly detection in time-series data. arXiv preprint arXiv:1608.04585
42. Ester M, Kriegel HP, Sander J, Xu X, et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd. AAAI Press. Palo Alto, California USA. pp 226–231
43. Breunig MM, Kriegel HP, et al (2000) LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. Association for Computing Machinery. New York, NY, USA. pp 93–104
44. Guha S, Mishra N, Roy G, Schrijvers O (2016) Robust random cut forest based anomaly detection on streams. In: International conference on machine learning. JMLR.org. New York, NY, USA. pp 2712–2721
45. Angiulli F, Fassetti F (2007) Detecting distance-based outliers in streams of data. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management. Association for Computing Machinery. New York, NY, USA. pp 811–820
46. Kontaki M, Gounaris A, et al (2011) Continuous monitoring of distance-based outliers over data streams. In: 2011 IEEE 27th International Conference on Data Engineering. IEEE. Piscataway, NJ. pp 135–146
47. Zhou X, Hu Y, Wu J et al (2022) Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial IoT. IEEE Trans Ind Inf 19(1):570–580

Publisher's Note