Network Traffic Anomaly Detection Based on DGBi-SA Model

Xin Wang, Hong Dai*, Zheng Huang, Yue Han

***Abstract*—Network traffic anomaly detection plays a crucial role in today's network security and performance management. In response to the challenges in current network traffic data processing, such as insufficient structuring and dynamics, the paper proposes a network traffic anomaly detection approach based on DGBi-SA model, which integrates Dynamic Graph Convolutional Networks (DyGCN), Bi-directional Long Short-Term Memory (Bi-LSTM), and the Self-Attention mechanism. To solve the problem of insufficient structuring, the model employs DyGCN to capture network traffic's dynamic graph structural characteristics proficiently. Simultaneously, combining Bi-LSTM and the Self-Attention mechanism enhances the model's ability to capture temporal and spatial dependencies. The main objective of the research in the paper is to improve the accuracy and detection efficiency of network traffic prediction, thereby achieving effective detection and timely response to abnormal behaviors in complex network environments. Through a sequence of experiments conducted on the CICIDS-2017 dataset, the paper verifies that the DGBi-SA model can effectively solve the above problems. The experimental results indicate that the model outperforms existing methods in key performance indicators such as AUC, precision, recall, and F1 score, thereby proving that the DGBi-SA model has important application value and scientific significance in network traffic anomaly detection.**

***Index Terms*—Network Traffic Anomaly Detection, Dynamic Graph Convolutional Network, Bi-directional Long Short-Term Memory, the Self-Attention Mechanism**

I. INTRODUCTION

In today's digital age, computer networks have become indispensable to people's daily lives, business activities, and scientific research. However, with the popularization of the Internet and the rapid development of information technology, network security is also facing increasingly complex and frequent threats. Network anomaly behaviors include network attacks, malicious traffic, and unauthorized access, which pose serious threats to the data security of the individuals, businesses, organizations, and may lead to service interruptions, system failures, and financial losses[1]. Therefore, effective network anomaly detection technologies ensure network security and performance.

Traditional network traffic anomaly detection methods are classified into three categories: supervised learning, unsupervised learning, and semi-supervised learning anomaly detection[2]. Among them, supervised learning is suitable for the case where there is a sufficiently large number of labeled samples in the dataset. In contrast, unsupervised learning is suitable for completely unlabeled datasets, and semi-supervised learning is for the case where only some of the samples in the dataset are labeled. In recent years, numerous researchers have utilized Support Vector Machines (SVMs) for network traffic anomaly detection. Eskin et al.[3] applied SVMs to anomaly detection. However, in practice, the training process often contains much noise, which can lead to poor generalization ability of the model. Subsequently, Hu et al. proposed the Robust Support Vector Machine (RSVM) algorithm that reduces noise interference, leading to improved results. In 2021, Ma et al.[4] employed Kernel Support Vector Machines (KSVM) for anomaly detection in network traffic. The study was innovative in choosing the appropriate kernel function and parameter settings, which allowed the model to handle nonlinear and high-dimensional data more efficiently. Then, Fosić et al.[5] presented a supervised learning method for Netflow network traffic anomaly detection. The method analyzes normal and anomalous network traffic data distribution and compares multiple algorithms to select the best anomaly detection solutions. However, although the method achieved some results in improving detection efficiency and accuracy, supervised learning still relied on a large amount of labeled data, and it is extremely difficult to obtain large-scale, high-quality labeled data in real network environments. Furthermore, when the distribution of network traffic data changes, the supervised learning model may need to be retrained to adapt to these changes, which greatly limits the model's adaptability and usefulness.

In order to further improve the performance of network traffic anomaly detection, researchers have integrated deep learning algorithms, including Convolutional Neural Networks (CNNs)[6], Recurrent Neural Networks (RNNs)[7], and Deep Auto-encoders[8]. These algorithms showed impressive results in various experiments. Compared with traditional machine learning algorithms, deep learning algorithms have stronger expressive and generalization abilities and can more accurately detect anomalous behaviors in network traffic. References[9-10] introduce the network traffic anomaly detection model that integrates the CNN with the spatio-temporal data features and the Bi-directional Long Short-Term Memory Network (Bi-LSTM), yielding

Manuscript received July 17, 2024; revised December 13, 2024. The research work was supported by the Fundamental Research Funds for the Liaoning Universities (No. LJ212410146058) and the Graduate student science and technology innovation project of University of Science and Technology Liaoning. (No. LKDYC202423).

Xin Wang is a postgraduate student of University of Science and Technology Liaoning, Anshan, Liaoning, CO 114051 China (e-mail: 2185803177@qq.com).

Hong Dai* is a professor in the School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, CO 114051 China (corresponding author to provide phone: +086-186-4226-8599; fax: 0412-5929818; e-mail: dear_red9@163.com).

Zheng Huang is an associate professor of School of Computer Science and Software Engineering, University of Science and Technology Liaoning, Anshan, Liaoning, CO 114051 China (e-mail: huangzheng@ustl.edu.cn).

Yue Han is a postgraduate student of University of Science and Technology Liaoning, Anshan, Liaoning, CO 114051 China (e-mail: 296872286@qq.com).

favorable outcomes. References[11-12] proposed a model based on the Self-Attention Mechanism (SAM) for extracting the correlation of multiple features of network traffic data.

However, despite the progress made by existing research, the network traffic anomaly detection field still needs to overcome many challenges. One of the main challenges is the need for more structure and dynamics in processing network traffic data, making traditional network anomaly detection methods difficult to adapt. Based on these challenges, the paper proposes a network traffic anomaly detection method based on DGBi-SA model. The model leverages the capabilities of DyGCN[13] for handling graph-structured data, effectively extracting spatial features from network traffic while incorporating time-series dependencies through Bi-LSTM[14] to enhance the predictive performance for changing traffic. In addition, integrating the Self-Attention mechanism[15], the model integrates the screening of key information and focuses analysis into the overall detection process, realizing comprehensive and accurate network traffic anomaly detection.

II. Proposed Method

A. Flow Matrix Prediction Model

In the paper, we defined the network traffic matrix as:

$$M^t = \begin{bmatrix} m_{11}^t & m_{12}^t & \cdots & m_{1n}^t \\ m_{21}^t & m_{22}^t & \cdots & m_{2n}^t \\ \vdots & \vdots & \vdots & \vdots \\ m_{n1}^t & m_{n2}^t & \cdots & m_{nn}^t \end{bmatrix} \quad (1)$$

Here, M^t represents the traffic matrix at the t-th time interval, and m_{ab}^t denotes the traffic value of the Origin-Destination (OD) pair from node a to node b. The prediction model aims to estimate the traffic matrix for the next time interval, based on historical data:

$$\tilde{M}^{t+1} = f\left(M^1, M^2, M^3, \ldots, M^t\right) \quad (2)$$

Where, $f(\bullet)$ is a mapping function that predicts the future traffic matrix $\tilde{M}^{t+1}$ based on the historical flow matrix M^t.

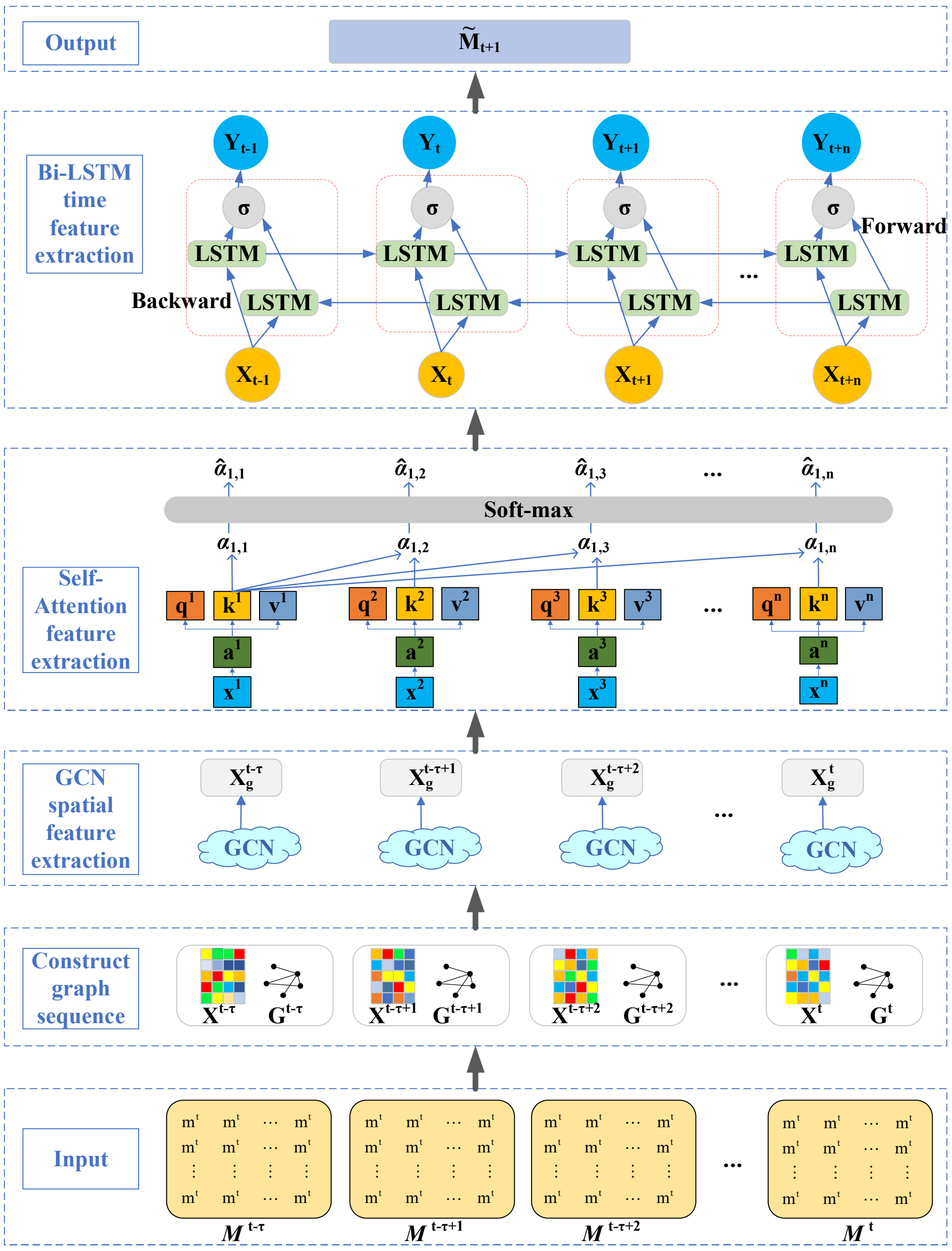

Fig. 1. The overall architecture of the flow matrix prediction model

The overall architecture of the flow matrix prediction model is shown in Fig.1. The purpose of the paper specifically refers to the DGBi-SA model, which integrates graph-structured data processing, the capture of dependencies in time series data, and the Self-Attention mechanism to achieve the comprehensive analysis and prediction of the traffic matrix.

For the historical traffic matrix time series $\{M^1, M^2, M^3, \ldots, M^t\}$, we first construct a sequence of graphs to derive the traffic matrix graph sequence $\{G^1, G^2, G^3, \ldots, G^t\}$ and the sequence of feature matrices $\{X^1, X^2, X^3, \ldots, X^t\}$. Then, the Graph Convolutional Network (GCN) is used for spatial feature extraction to capture the representation of the traffic matrix at each moment. Next, we use Bi-LSTM to extract temporal characteristics, outputting a sequence of temporal features $\{h^1, h^2, h^3, \ldots, h^t\}$.

Next, the Self-Attention calculates the correlation scores among all positions in the traffic matrix sequence. These scores are used as weights to sum up the representations of all positions, forming a new representation for each position. Finally, the output from the Self-Attention layer is passed through the Softmax activation function to generate the predicted matrix $\tilde{M}_{t+1}$.

The model continuously optimizes the model parameters by backpropagation and stochastic gradient descent method during each round of iteration, and finally, the loss between the prediction matrix $\tilde{M}_{t+1}$ and the true matrix M_{t+1} is minimized.

B. Constructing Graph Sequence

The flow matrix modeling is transformed into the graph structure in the paper. Fig.2 illustrates the traffic matrix for different time slices to build the graph sequence. In the structure, the network nodes are used as the nodes in the graph, and the OD flows between two nodes are used as edges between two nodes in the graph. Each edge's weight represents the OD flow's size between the two nodes.

The left matrix M^t represents the network flow matrix at the one-time slice, with each node in the right graph representing a backbone network node. It can be seen from the figure that the traffic matrix M^t is an $N \times N$ matrix, where N nodes are in the network, and N nodes are also in the graph structure. The element m_{ab}^t in the matrix represents the traffic size from the source node a to the destination node b. Therefore, an edge between node a and node b exists in the graph data, with a weight m_{ab}^t directed from node a to b.

The relationship expressed by the traffic matrix is mathematically represented as:

$$G^t = \left(v^t, \varepsilon^t\right), t \epsilon \{1, 2, 3, \ldots, T\} \tag{3}$$

Here, v represents the set of all nodes in the graph and ε denotes the set of all edges. The count of network nodes remains constant across different time slices within the same network. Therefore, traffic matrix prediction can be further abstracted as the link prediction problem, where the objective is to predict the set of edges in the graph for the next time slice based on the historical edge set. This involves learning structure over time to accurately forecast future links.

It can also be regarded as the problem of predicting the edge weights in the graph, that is, predicting the weights of all edges in the graph G^{t+1} at the forthcoming moment based on the historical graph sequence of the graph $\{G^1, G^2, G^3, \ldots, G^t\}$.

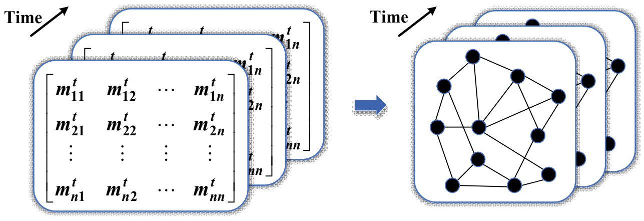
Time
$\begin{bmatrix} m_{11}^t & m_{12}^t & \cdots & m_{1n}^t \\ m_{21}^t & m_{22}^t & \cdots & m_{2n}^t \\ \vdots & \vdots & \vdots & \vdots \\ m_{n1}^t & m_{n2}^t & \cdots & m_{nn}^t \end{bmatrix}$
Time

Fig. 2. Flow matrix pre-time series converted to graph time series schematic

After extracting the node features for each time slice, the generated node features form the feature matrix $\mathrm{X}^t \in R^{N \times d}$ of the node-set, where d denotes the number of node features. Each time slice extracts the network node features similarly, forming a graph time series $\{G^1, G^2, G^3, \ldots, G^t\}$ for each graph in the sequence of node feature matrices $\{X^1, X^2, X^3, \ldots, X^t\}$.

C. Spatial Feature Modeling Based on GCN

GCN extracts features through neighborhood aggregation and the utilization of graph topological structures, enabling it to adapt to graphs of varying sizes and complexities. In the paper, we take advantage of GCN's graph structure extraction to learn the structure information under each time slice. GCN extends the idea of convolution to graphs by aggregating neighbor information through the defined spectral graph convolution.

Given a directed graph $G=(v, \varepsilon, X)$, where v, ε and X are the set of nodes, the set of edges, and all the node features that form the node feature matrix in the graph, respectively. Approximation of convolution operations using Chebyshev polynomials is an effective optimization method for GCN. The approach allows GCN to approximate the spectral convolution of a graph without directly computing the feature decomposition of the graph, thus enabling more efficient operation on large graphs and reducing computational complexity. The Chebyshev polynomial is calculated as follows:

$$f_\theta * G(X) = \sum_{k=0}^{K} \theta_k T_k\left(\hat{L}\right) X \tag{4}$$

Here, f_θ represents the convolution kernel; $\theta \epsilon R^K$ is a polynomial vector of degree K; $T_k(\hat{L})$ is the K-th order polynomial of the Chebyshev polynomials in the graphical Laplacian in the normalized form $\hat{L}=2L/\lambda_{max}-I_N$, where $\lambda_{\max}$ is the largest eigenvalue of L, and I_N is the N-dimensional identity matrix. Define $T_0(X)=1$, $T_1(X)=X$, $T_{k+1}(X)=2XT_k(X)-T_{k-1}(X)$.

Solve the above equations by using an approximate expansion, from which the information of the neighboring nodes up to the K-th order can be extracted.

The process allows for the aggregation of information from a node's neighbors at various levels of proximity, capturing both direct and indirect relationships within the network. This enhances the model's ability to capture complex dependencies.

Additionally, the GCN operates through an activation function to introduce non-linearity. In the paper, the ReLU (Rectified Linear Unit) activation function is employed, which is commonly used due to its effectiveness in preventing vanishing gradients and promoting faster convergence during training. Fig.3 shows the GCN message passing process.

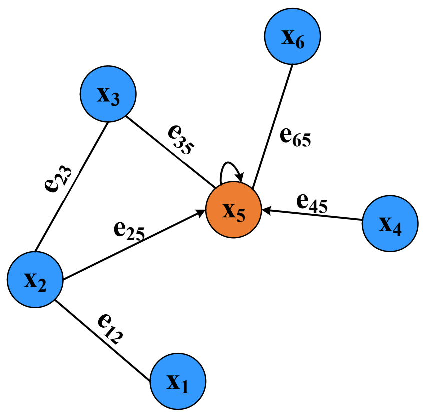

Fig. 3. GCN message passing process

According to the above formula, assuming $k = 1$ and $\lambda_{max} = 2$, then the formula for the first-order graph convolution operation can be expressed as:

$$X_G = f_\theta * G(X) = \sigma\left(D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}X^{(l)}W^{(l)}\right) \qquad (5)$$

Here, $\hat{A} = I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where D and A represent the degree and adjacency matrices of the graph, respectively; W are the model parameters; and σ is the activation function.

Fig.3 illustrates the process of the first-order message passing in GCN when k=1. Node X_5 represents the central node, with the other nodes representing its neighboring nodes. The first-order GCN aggregates neighbor features through the relationships between the central node and its neighboring nodes, thereby obtaining the spatial features of the central node.

In summary, first-order graph convolution can effectively obtain the first-order neighbor spatial features. Therefore, graph convolution is used to process the traffic matrix graph time series in the paper so that each node's representation contains the spatial information of its first-order neighbors. The method aggregates OD flows with the same source and destination nodes to extract spatial correlation features between them.

Finally, spatial representation for each traffic matrix is generated using spatial relationships.

D. Temporal Feature Modeling Based on Bi-LSTM

This section will model and analyze the temporal dependence properties between time series, mainly by inputting the vector of high-dimensional flow matrix representations into a recurrent neural network to explore the temporal correlation between the flow matrices.

The Long-short-term memory (LSTM) networks[16]are a type of RNN that combines short-term and long-term memories through a gate control mechanism to solve the gradient vanishing problem and enable the modeling of long sequence dependencies.

The propagation equation is expressed as follows[17]:

$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right) \qquad (6)$$

$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right) \qquad (7)$$

$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right) \qquad (8)$$

$$g_t = tanh(W_{gx}x_t + W_{gh}h_{t-1} + b_g) \qquad (9)$$

$$c_t = f_t * c_{t-1} + i_t * g_t \qquad (10)$$

$$h_t = o_t * \tanh(c_t) \qquad (11)$$

In the above equation, x_t is the t-th element in the input sequence, h_t is the hidden state at time step t, and c_t is the cell state at the t-th time step. i_t, f_t and o_t are the activation values of the input gate, the forget gate, and the output gate, respectively.

And g_t is the current candidate memory value. W and b are trainable parameters in the LSTM model, which are tuned to minimize the prediction error during the training process. W determines the impact of different input features on the model output, while b provides additional flexibility to the model.

σ and $tanh(\bullet)$ are the activation functions used to introduce nonlinear transformations into the computation of the model. And σ usually refers to the Sigmoid function, which compresses the input between 0 and 1, and it is used to control the degree of door opening.

Whereas, $tanh(\bullet)$ is the hyperbolic tangent function which compresses the input between -1 and 1, and it is used to generate updates of cell states and candidate memory values.

The structure of the LSTM unit is shown in Fig.4, which includes three gates: the forget gate, the input gate, and the output gate.

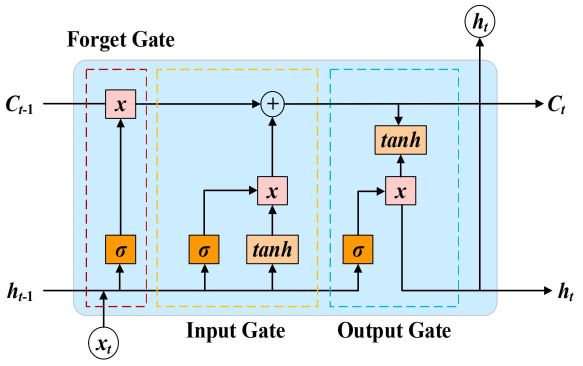

Fig. 4. An LSTM cell

Among them, the forget gate plays a crucial role in deciding which hidden information should be forgotten or discarded. It uses the Sigmoid activation function to determine which details need to be deleted from memory, ensuring that only relevant information is retained.

The input gate, on the other hand, can selectively add the input information of the current moment to the memory state, updating the cell state with new data. The gate regulates how much new information enters the memory by controlling the flow of input signals.

Finally, the output gate is responsible for generating a new hidden state and the output of the time step. It combines the inputs and the memory to determine the output, ensuring that the model retains important features while passing forward the necessary information for future time steps. The allows the model to focus adaptively on relevant data. Moreover, the output gate helps the model maintain a balance between short-term memory and long-term context, crucial for sequential data tasks.

Bi-LSTM[18] is an enhanced version of the traditional LSTM model. Its structure consists of LSTM units in two directions: one that processes the input sequence in chronological order, called the forward LSTM, and another that processes the input sequence in reverse chronological order, called the reverse LSTM[20]. The bidirectional structure allows Bi-LSTM to capture both past and future dependencies in the data, improving its ability to model complex temporal relationships.

The bi-directional feature extraction process of Bi-LSTM is shown in Fig.5.

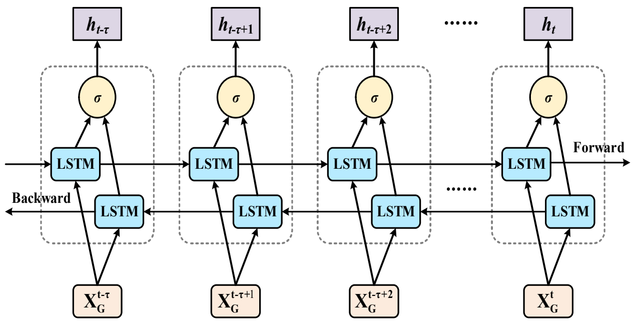

Fig. 5. Forward and backward feature extraction in Bi-LSTM

As shown in the figure, the outputs of these two directions are concatenated or merged to produce the final output. The advantage of this structure is that the forward LSTM captures pre-temporal features. In contrast, the reverse LSTM captures post-temporal features, and combining the two provides a more comprehensive feature-capturing capability for network anomaly detection.

For example, certain anomalous behavior may manifest as an abnormal packet increase in the first few moments of the flow and as a disconnection in the subsequent moments. Combining the timing information from these two directions allows Bi-LSTM to capture this back-and-forth dependency more effectively.

As shown in the above figure: Assuming that the input sequence is $X=\{X^1,X^2,X^3,\ldots,X^t\}$, the forward LSTM unit receives the input sequence from the forward direction and outputs the hidden state h_t step by step.

In contrast, the reverse LSTM unit processes the input sequence from the backward direction and outputs the hidden state h_t. At each time step t, the final output of Bi-LSTM is splicing the output of forward LSTM with the output of reverse LSTM, as shown in Eq.(12):

$$h_t=\left[\vec{h}_t,\overleftarrow{h}_t\right] \tag{12}$$

Here, h_t is the output of the spliced, $\vec{h}_t$ and $\overleftarrow{h}_t$ denote the output of the forward and reverse LSTM, respectively. This bi-directional splicing enables the model to capture both forward and backward information.

The calculation of $\vec{h}_t$ and $\overleftarrow{h}_t$ can be expressed by the following equation:

$$\vec{h}_t=\vec{o}_t * \tanh\left(\vec{c}_t\right) \tag{13}$$

$$\overleftarrow{h}_t=\overleftarrow{o}_t * \tanh\left(\overleftarrow{c}_t\right) \tag{14}$$

Here, $\vec{o}_t$ and $\overleftarrow{o}_t$ are the forward and reverse outputs of time t, respectively. $\vec{c}_t$ and $\overleftarrow{c}_t$ are the forward and reverse hidden unit cell states at the moment t.

E. Feature Correlation Detection

Self-Attention[20] is a mechanism for capturing the dependencies among different positions within the sequence.

First, the correlation scores of each position in the input sequence with all other positions are computed, which reflect the relevance between different elements in the sequence. Then, weighting and summing the representations of all positions using these scores as weights allows the model to focus on the most relevant parts of the sequence.

As shown below, Fig.6 illustrates the specific process of calculating the attention value.

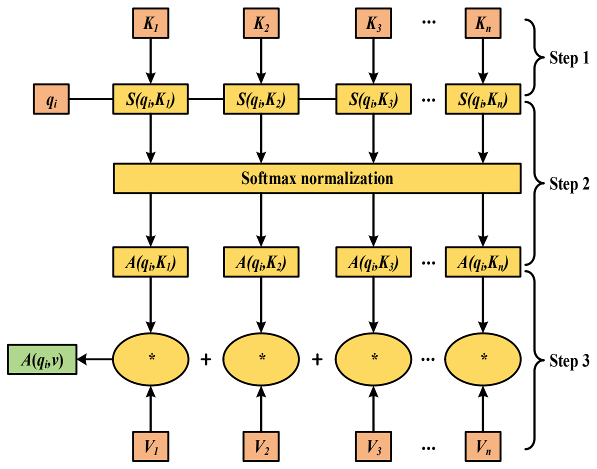

Fig. 6. Specific workflow for calculating attention value

Here, q_i represents the query vector associated with the i-th input. Meanwhile, k_j and v_j represent the key vector and value vector of the j-th input, respectively. Consequently, the attention value is determined by evaluating the relationship between the i-th input and each of the *1*st, *2*nd, ..., and the j-th inputs, thereby emphasizing the correlation among all inputs. The specific calculation is divided into three steps.

Firstly, the correlation score between each pair of elements is calculated by the Scaled Dot-Product score function, which is generally scaled by dividing by $\sqrt{d_k}$ (the dimension of k); the calculation is as follows:

$$S\left(q_i,k_j\right)=\frac{q_i\cdot k_j}{\sqrt{d_k}} \tag{15}$$

Then, the scores are normalized by using the Softmax function to obtain the attention weights, which are given by the following equation:

$$A\left(q_i,k_j\right)=\frac{\exp\left(S\left(q_i,k_j\right)\right)}{\sum_{j=1}^{n}\exp\left(S\left(q_i,k_j\right)\right)} \tag{16}$$

Finally, the value vectors are weighted and summed using the attention weights to obtain the output representation, which can be articulated as follows:

$$A\left(q_i, V\right) = \sum_{j=1}^{n} A\left(q_i, k_j\right) \cdot v_j \tag{17}$$

In this way, for each element xi, the Self-Attention mechanism computes the weighted representation, considering information from all other positions in the sequence, creating long-distance dependencies among different positions.

III. EXPERIMENTAL RESULTS AND ANALYSIS

A. Dataset

The CICIDS-2017 dataset was created by the Canadian Institute for Cybersecurity Research[21]. The dataset contains 283,0743 traffic records, with 70% allocated for the training set, 15% for the validation set, and 15% for the testing set. Each traffic record consisted of 78 features. The data collection period started on July 3 and ended on July 7, 2017, spanning 5 days. The first day's traffic contained only normal traffic. In contrast, over the next four days, a variety of malicious traffic was collected, including File Transfer Protocol (FTP), Secure Shell (SSH), Distributed Denial of Service (DDoS) attacks, Heartbleed vulnerabilities, Web attacks, infiltration, and botnets.

Compared with most other traffic datasets (such as the classic KDDCUP99, NSLKDD, etc.), the CICIDS-2017 dataset has more varied traffic types and a larger scale, which better reflects the real network environment and thus becomes one of the most used datasets in the field of anomaly traffic detection[22].The following table describes the main features of the CICIDS-2017 dataset, as shown in Table I.

TABLE I
INTRODUCTION TO THE CICIDS-2017 DATASET

Node Number /units	Edge Number / units	Graph Number/ units	Attack types /types	Flow Feature Dimensions/ dimensions
19211	2824000	2824	14	78

The following preprocessing steps are required before inputting the traffic data into the model:

1) Constructing Graph Data Structure

For each traffic record in a CSV file, an edge list is first constructed based on the source IP, destination IP, and labels. Next, the IP addresses are encoded using LabelEncoder to convert them into corresponding node ID, and based on these encodings the characteristics of the nodes are generated, such as the number of packets or bytes transmitted and other traffic statistics.

2) Adjacency Matrix Construction and Normalization

For each communication graph, the adjacency matrices of source IPs to flows, the adjacency matrices of destination IPs to flows, and the adjacency matrices between IP nodes are constructed, respectively. Subsequently, these adjacency matrices are normalized, and commonly used methods include random walk normalization and symmetric normalization.

3) Graph Segmentation and Labeling

Each communication graph's edge list, feature list, and label list are sorted for subsequent processing. In temporal order, every 1000 flow records form a communication graph G, creating a sequence of graphs that represent the network's behavior over time.

4) Communication graph labeling

Set labels for each communication graph. If malicious traffic is detected, set its label to 1; conversely, if it is normal traffic, set it to 0. The labeling process is crucial for supervised learning, as it provides the necessary ground truth for training the model.

B. Experimental Parameter Design

While performing the dynamic graph anomaly detection task, the model learns the normal pattern using normal data, and then identifies points on the new data that significantly depart from the learned normal pattern, which are judged as anomalies.

Therefore, the data in the training set should be normal and time-continuous. For the CICIDS-2017 dataset, the training set was constructed from data generated on Monday, and the test set consisted of data from Tuesday through Friday. The results of the division of the dataset are shown in Table II.

TABLE II
PARTITIONING OF THE CICIDS-2017 DATASET

Training set/ records	Validation Set/ records	Testing set/ records
1976	423	423

In order to evaluate the performance of the model, the paper draws ROC curves consisting of True Positive Rate (TPR) and False Positive Rate (FPR) at the different thresholds. TPR refers to the proportion of actual positive cases that the model correctly identifies as positive, meaning the true positives are divided by the total number of actual positives. In contrast, FPR refers to the proportion of actual negative cases the model incorrectly identifies as positive, meaning the false positives are divided by the total number of actual negatives. The area under the ROC curve (the AUC value) is a key measure of the model's predictive performance, where a higher AUC value indicates better model performance.

The shape of the ROC can help analyze the performance of the model under different thresholds, especially when dealing with unbalanced datasets. It can clearly demonstrate the effect of the model.

Additionally, the paper calculates the threshold on the ROC curve that maximizes the difference between TPR and FPR to determine the optimal cut point and evaluate the precision, recall, and F1 score corresponding to that point. Together, these metrics provide a comprehensive view of the model's performance evaluation.

Firstly, during the model training process, in order to obtain sufficient training samples while maintaining the order of the time series data, to better learn the characteristics of the data, and to enhance the generalization ability of the model, the paper adopts the sliding window with a size of 5 and a step size of 1 to segment the graphical dataset. Specifically, the sliding window is shifted to one unit on the right at a time to generate continuous time series samples. The approach ensures that each sample contains sufficient contextual information, enhancing the model's understanding of time series data.

Secondly, the learning rate is an important hyperparameter of the optimization algorithm, which directly affects the learning speed and stability of the model during the training process. A higher learning rate may cause the model to converge quickly during training, but it may likely skip the global optimal solution. In contrast, the lower learning rate can make the model more stable towards the global optimal solution but with a slower convergence speed. After many experiments and tuning, this paper determines that a learning rate of 0.0001 can well balance the stability of the model and the quality of convergence. In addition, the learning rate is also automatically adjusted using the Adam optimizer, which can effectively accelerate the training process and improve the model performance.

Next, Layer Normalization was implemented in the DGBi-SA model, which is designed to stabilize changes in the internal activation distribution during training.

In addition, the choice of dimension for node embedding is crucial; too small a dimension may lead to information loss, while too large a dimension may increase the computational complexity and risk of overfitting. Ultimately, based on the previous experiments' multiple attempts and evaluations, when the node embedding dimension is set to 64, it can achieve excellent information characterization under an acceptable computational load.

Finally, the paper chose 50 iterations of training to ensure that the algorithm can converge and avoid overfitting. The setting maximizes the model performance within a reasonable training time, while the training process is controlled by monitoring it through methods such as cross-validation to ensure the model's ability to generalize on the test data.

C. Anomaly Detection Performance

In order to validate the performance of the DGBi-SA model, the research conducted ablation experiments designed to clarify the extent to which the Self-Attention mechanism and the Bi-LSTM layer contribute to the overall performance of the model. By excluding these two key components one by one, we analyzed the performance of the model under different constructions. The ablation method can clearly demonstrate the impact of each component on the model performance and justify further optimization of the model. In the experiment, the paper set the anomaly score corresponding to the optimal cut point on the ROC curve as the anomaly detection threshold to ensure optimal sensitivity and specificity are considered when evaluating the model performance.

The results of the experimental data are shown in Table III.

TABLE III
ANOMALY DETECTION RESULTS

Model	AUC(%)	Precision(%)	Recall(%)	F1(%)
DGBi-SA	0.83	0.95	0.80	0.87
DyGCN+Bi-LSTM	0.76	0.86	0.66	0.77
DyGCN+Self-Attention	0.74	0.82	0.55	0.61

Table III shows that the DGBi-SA model performs well in terms of AUC, precision, recall, and F1 score, which proves its effectiveness and reliability in anomaly detection tasks. These results provide an important reference for further research and application. The superior performance of the DGBi-SA model indicates its ability to accurately detect anomalies, even in complex and dynamic network environments. By achieving high precision and recall, it demonstrates a balanced ability to correctly identify both normal and anomalous behaviors, reducing the likelihood of false positives and false negatives. Furthermore, the impressive F1 score highlights the model's robustness in handling imbalanced datasets, where the distribution of normal and anomalous instances may be skewed. These findings suggest that the DGBi-SA model is a promising candidate for real-world anomaly detection applications, offering a reliable solution for improving network security and operational efficiency.

The results of the ROC curves for each model in Fig.7 illustrate their performance and capabilities.

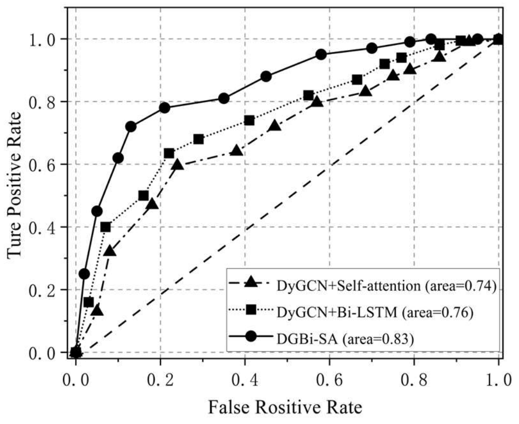

Fig. 7. The ROC curve of CICIDS-2017

The above experiments show that the DGBi-SA model performs significantly better than the rest of the configuration models. Specifically, it maintains the optimal balance between TPR and FPR and has better generalization ability and robustness.

The DGBi-SA model incorporates graph representation techniques and sequence analysis methods. The model employs DyGCN to capture data structure features and Bi-LSTM to deepen the understanding of time series dependencies. Meanwhile, the introduction of the Self-Attention mechanism further improves the efficiency of recognizing key data points related to anomaly detection. The combination enables the model to capture the traffic data's spatial correlation and temporal evolution. Experimental results show that the model performs well in AUC, precision, recall, and F1 scores, demonstrating a good balance of accuracy and recall, effectively reducing false alarms and improving the ability to capture real anomalies.

The DyGCN+Bi-LSTM model combines the ability of dynamic graph convolutional networks to aggregate information about neighboring nodes with the power of Bi-LSTM to capture time series dependencies. The combination improves the model's sensitivity to network mobility and temporal continuity, thus enhancing its overall performance. Nevertheless, there is still a gap in the overall performance of the model in terms of various metrics compared to the DGBi-SA model.

The DyGCN+Self-Attention model combines the DyGCN and the Self-Attention mechanisms, which are designed to enhance the ability to capture critical information. DyGCN effectively aggregates information from neighboring nodes to enhance the understanding of the network structure. At the same time, the Self-Attention mechanisms enable the model to be more flexible in focusing on important parts of the traffic data related to anomaly detection when processing sequence data. The combination enhances the model's sensitivity to dynamic network environments.

However, the experimental results indicate that relying solely on the combinations is insufficient for optimal anomaly detection. Additional methods or model refinements are needed to fully capture the complex patterns in the data.

IV. Conclusion

In the realm of contemporary cybersecurity and performance management, the effective detection of network traffic anomalies is of paramount importance. To overcome the current challenges in processing traffic data, such as the dynamic nature of data processing and the lack of structured information, the paper proposes a model based on DGBi-SA to cope with these complexities. The model effectively captures and analyzes the dynamic graph structural information and time-series network traffic features, significantly improving the monitoring and response speed of abnormal behavior in network environments.

The experimental results on the CICIDS-2017 dataset show that the DGBi-SA model demonstrates higher efficiency and precision in processing network traffic data and has obvious advantages in key performance metrics such as AUC value, precision rate, recall rate, and F1 score.

The achievements not only show the potential of the model in the current application of network traffic anomaly detection but also provide strong guidelines and a research foundation for future research directions. Through the research, we further deepen the theoretical architecture of network anomaly detection technology and provide practical and effective technical support for maintaining network security.

References

[1] M. Ahsan, K. E. Nygard, R. Gomes, M. M. Chowdhury, N. Rifat, and J. F. Connolly, "Cybersecurity Threats and Their Mitigation Approaches Using Machine Learning—A Review," Journal of Cybersecurity and Privacy, vol.2, no.3, pp527-555, 2022.

[2] Buecker, S. Arunkumar, B. Blackshaw, M. Borrett, P. Brittenham, J. Flegr, J. Jacobs, V. Jeremic, M. Johnston, and C. Mark. Using the IBM Security Framework and IBM Security Blueprint to Realize Business-Driven Security; IBM Redbooks, 2014.

[3] W. Hu, Y. Liao, and V. R. Vemuri, "Robust Anomaly Detection Using Support Vector Machines," Proceedings of the International Conference on Machine Learning, vol.6, 2003.

[4] Q. Ma, C. Sun, B. Cui, and X. Jin, "A Novel Model for Anomaly Detection in Network Traffic based on Kernel Support Vector Machine," Computers & Security, vol.104, pp102215-102228, 2021.

[5] Fosić, D. Žagar, K. Grgić, and V. Križanović, "Anomaly Detection in NetFlow Network Traffic Using Supervised Machine Learning Algorithms," Journal of Industrial Information Integration, vol.33, pp 100466, 2023.

[6] K. Mishra, S. Paliwal, and G. Srivastava, "Anomaly Setection Using Deep Convolutional Generative Adversarial Networks in the Internet of Things," ISA Transactions, vol.145, pp493-504, 2024.

[7] S. J. Kumaresan, C. Senthilkumar, D. Kongkham, B. Beenarani, and P. Nirmala, "Investigating the Effectiveness of Recurrent Neural Networks for Network Anomaly Detection," In 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Bangalore, pp1-5, 2024.

[8] V. Dutta, M. Pawlicki, R. Kozik, and M. Choraś, "Unsupervised Network Traffic Anomaly Detection with Deep Autoencoders," Logic Journal of the IGPL, vol.30, pp912-925, 2022.

[9] Y. X. Geng, L. Wang, Z. Y. Wang and Y. G. Wang, "Central Attention Mechanism for Convolutional Neural Networks," IAENG International Journal of Computer Science, vol.51, no.10, pp1642-1648, 2024.

[10] Y.C. Wang, Y.C. Houng, H.X. Chen, and S.M. Tseng, "Network Anomaly Intrusion Detection based on Deep Learning Approach," Sensors, vol.23, no.4, pp2171-2191, 2023.

[11] S. Cai, H. Xu, M. Liu, Z. Chen, and G. Zhang, "A Malicious Network Traffic Detection Model based on Bidirectional Temporal Convolutional Network with Multi-head Self-Attention Mechanism," Computers & Security, vol.136, pp103580, 2024.

[12] Y. Y. Ding, and L. Wang, "Research on the Application of Improved Attention Mechanism in Image Classification and Object Detection," IAENG International Journal of Computer Science, vol.50, no.4, pp1174-1182, 2023.

[13] Z. Cui, Z. Li, S. Wu, X. Zhang, Q. Liu, L. Wang, and M. Ai, "DyGCN: Efficient Dynamic Graph Embedding With Graph Convolutional Network," IEEE Transactions on Neural Networks and Learning Systems, vol.35, no.4, pp4635-4646, 2024.

[14] Lin Wang, Hudie Dong, and Gang Chen, "New Lyapunov-type Inequalities for Fractional Differential Equations with Bi-ordinal Psi-Hilfer Fractional Derivative Involving Multi-point Boundary Conditions," Engineering Letters, vol.31, no.2, pp648-655, 2023.

[15] W. Hu, L. Cao, Q. Ruan, and Q. Wu, "Research on Anomaly Network Detection based on Self-Attention Mechanism," Sensors, vol.23, no.11, pp5059, 2023.

[16] L. Li, Y. Yang, Z. Yuan, and Z. Chen, "A Spatial-Temporal Approach for Traffic Status Analysis and Prediction based on Bi-LSTM Structure," Modern Physics Letters B, vol.35, no.31, pp2150481, 2021.

[17] Bo Zhang, Xinfeng Yang, Yongqing Zhang, and Dongzhi Li, "Short-Term Inbound Passenger Flow Prediction of Urban Rail Transit Based on RF-BiLSTM," Engineering Letters, vol.31, no.2, pp665-673, 2023.

[18] H. Ye, B. Cao, Z. Peng, T. Chen, Y. Wen, and J. Liu, "Web Services Classification Based on Wide & Bi-LSTM Model," IEEE Access, vol.7, pp43697-43706, 2019.

[19] Li Dongping, Yang Yingchun, Shen Shikai, He Jun, Shen Haoru, Yue Qiang, Hong Sunyan, and Deng Fei, "Research on Deep Learning Model of Multimodal Heterogeneous Data Based on LSTM," IAENG International Journal of Computer Science, vol. 49, no.4, pp1016-1022, 2022.

[20] A. A. Liu, H. S. Tian, N. Xu, W. Z. Nie, Y. D. Zhang, and M. Kankanhalli, "Toward Region-Aware Attention Learning for Scene Graph Generation," IEEE Transactions on Neural Networks and Learning Systems, vol.33, no.12, pp7655-7666, 2022.

[21] R. Dube, "Faulty use of the CIC-IDS 2017 Dataset in Information Security Research," Journal of Computer Virology and Hacking Techniques, vol.20, no.1, pp203-211, 2024.

[22] Hamza KAMAL IDRISSI, and Ali KARTIT, "Network Intrusion Detection using Combined Deep Learning Models: Literature Survey and Future Research Directions," IAENG International Journal of Computer Science, vol.51, no.8, pp998-1010, 2024.