Available online at https://jcupt.bupt.edu.cn

The Journal of China Universities of Posts and Telecommunications

April 2025, 32(2): 82 – 100

Research on network traffic anomaly detection method based on multimodel hybrid feature extraction

Guo Hongwei[1] (✉), Zhao Zhongnan[2]

1. College of Science, Heilongjiang Institute of Technology, Harbin 150050, China
2. School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China

Abstract

Hybrid feature extraction model that combines residual neural networks (ResNet) and bidirectional gated recurrent units (ResNet-BGRU) was proposed to address the low accuracy of single feature extraction in deep neural networks. ResNet captures spatial features of network traffic, while bidirectional gated recurrent units (BGRU) extracts temporal features. These features are integrated and refined using a multi-head attention mechanism to handle spatiotemporal multi-dimensional data and adapt to diverse detection scenarios. By enhancing feature representation and capturing complex patterns, the model improves detection performance. Experiments on multiple datasets demonstrate that ResNet-BGRU outperforms traditional machine learning and comparable deep learning methods in accuracy and computational efficiency, offering a more robust solution for network traffic anomaly detection.

Keywords anomaly detection, deep learning, feature extraction, multi-model

1 Introduction

Network traffic anomalies[1] refer to situations where the behavior of network traffic deviates from its normal behavior. With the increasing scale and complexity of the network, the impact of network traffic anomalies on network performance is becoming more and more significant. Accurate and fast detection of network traffic anomalies and a reasonable response are the prerequisites to ensuring normal network operation, so network traffic anomaly detection has become a research topic of great interest[2-3].

However, the greater the dependence of human society on the Internet, the more difficult it is to estimate the damage that a major cybersecurity incident can cause to human society. In today's society, both the personal network services used by residents and the vital national government's network systems are subject to serious cybersecurity threats every day. Once a network security problem occurs, it may lead to the interruption of application services and the leakage of users' personal information, or it may cause serious economic losses or even national security problems. As various network technologies are updated and iterated, and the complexity of the system structure increases, it becomes increasingly difficult to monitor the traffic

Special Issue: The 27th Annual Meeting of The China Association for Science and Technology

Corresponding author: Guo Hongwei, E-mail: guohongwei@hljit.edu.cn

DOI: 10.19682/j.cnki.1005-8885.2025.0020

data.

In response to the complex and serious network security issues, all fields conduct research on network security issues, and various security protection technologies are launched and developed, such as firewall technology, user security authentication mechanisms, access rights control, and other measures. Even so, due to the continuous innovation of Internet technology, the network environment is becoming more complex, and the attack methods are becoming more sophisticated. Relying only on these traditional defense methods cannot completely guarantee the security of the network.

From another perspective, as the use of networks becomes more and more extensive and complex, the application scenarios for networks also become more and more complex[4], requiring higher and higher accuracy for network anomaly detection[5]. In this context, traditional network traffic anomaly detection techniques suffer from low accuracy and high false alarm rates due to their slow response, limited generalization capability, and inability to effectively detect unknown attacks. Therefore, it is important for network security to study more efficient network traffic anomaly detection methods to improve accuracy and real-time detection.

Network traffic anomaly detection[6] is a very effective defense measure and has become an increasingly important research hotspot because it can quickly detect known or unknown attacks occurring in the network. Network traffic anomalies, i. e., data traffic that has an impact on the normal operation as well as the stable operation of the network system, are quite different from regular traffic data and will generally lead to reduced network performance or, in more serious cases, may cause the entire system to be paralyzed. There are two main causes of network traffic anomalies[7]: firstly, performance reasons due to imperfect network architecture design or improper user operation resulting in abnormal traffic, such as network link connection problems, network equipment problems, and so on. Secondly, security reasons, that is, the network experienced attack behavior resulting in abnormal traffic, such as distributed denial of service (DoS) attacks, worms, and so on. In this paper, we study mainly network traffic generation anomalies due to security factors. Network traffic anomaly detection, that is, using all kinds of network anomaly detection methods to analyze the traffic and quickly find out the traffic data with attack behavior.

With the impressive achievements of deep learning techniques in fields such as speech recognition, network traffic anomaly detection using deep learning methods has become a hot research direction in recent years[8-10]. Deep learning can automatically extract high-level features from a large amount of data through multilayer neural networks to identify network traffic anomalies. Therefore, in the face of the massive data in the current network environment, designing a reasonable network traffic anomaly detection model according to the network traffic and making full use of the advantages of deep learning are expected to further improve the detection accuracy of network traffic anomalies.

2 Related work

Network traffic anomaly detection technology has evolved to the present day with a wealth of accumulated results. Researchers have studied the topic of network traffic anomaly detection since the early nineteenth century, developing a large number of algorithms and models in the hope of providing better solutions and methods for the progressively more complex network security threats. According to the research results of Moustafa et al.[11], network traffic anomaly detection methods can be classified into six categories: classification-based, clustering-based, deep learning-based, knowledge-based, hybrid-based, and statistical-based[12-15]. This paper focuses on deep learning-based methods.

Most of the existing mainstream network traffic anomaly detection methods apply traditional machine learning methods to network traffic anomaly detection[16-17], and although they achieve good results, the traditional machine learning methods first require humans to design some effective features, which leads to higher labor costs in the face of massive

data and is a shallow learning method. However, with the development of the Internet, the increasing amount of massive data in the network poses a great challenge to the traditional approach.

Deep learning, a subset of machine learning, employs the construction of multi-layer neural networks to make the model reach a relatively "deep" layer and can automatically extract deep-level features from massive data without complicated manual feature extraction, so it can save a lot of labor costs. Deep learning gains widespread adoption in network traffic anomaly detection due to its robust automatic feature extraction capabilities.

Convolutional neural networks (CNN), a deep learning model based on image information, are widely used in the field of network traffic anomaly detection because of their excellent spatial feature extraction capability. To exploit the excellent spatial feature extraction capability of CNN, Jia et al.[18] applied a CNN to an intrusion detection system, and its binary classification accuracy on Knowledge Discovery and Data Mining Cup 1999 (KDD-Cup99) reached 95.07%. To address the problem of multiple participants training a global model without sharing privacy data to improve the accuracy of network anomaly detection, false alarm rate, and lack of data labeling, Zhao et al.[19] proposed a network anomaly detection model incorporating federation learning and CNN and conducted binary classification and multi-classification experiments on the Network Security Laboratory-Knowedge Discovery and Data Mining (NSL-KDD) dataset, respectively, both of which achieved high classification accuracy. Refs. [20 – 21] take advantage of the good classification ability of CNN for images, first converting the traffic data into images, and then using CNN to extract the features of the images, thus identifying the attacks. In Ref. [22], a CNN-based character-level network anomaly detection system was proposed, where network traffic records are considered to be sequences of characters, and the character vectors in the records are aggregated into a matrix as the input to the CNN, followed by traffic classification. Li et al.[23] proposed a single model that can detect collective and point anomalies using stacked temporal convolutional networks, and their experimental results show that CNN outperforms the recurrent neural network (RNN) model and overcomes the drawback that the traditional model cannot distinguish point anomalies from collective anomalies.

RNN is a type of neural network that processes sequential data as input and iteratively handles information through a recurrent structure along the progression of the sequence. It has excellent temporal feature extraction capabilities because it is well suited for extracting temporal features from network traffic data. Li et al.[24] noted that traditional machine learning algorithms require manual design before building model features, feature selection can only be done for specific datasets, and most anomaly detection methods ignore the temporal information between packets. To address the above problems, they developed a hybrid model based on RNN and restricted Boltzmann machines, and achieved high detection results on both anomaly detection datasets. Manickam et al.[25] combined likelihood fuzzy C-mean clustering with a RNN and applied it successfully to an intrusion detection system in a cloud environment. Fu et al.[26] combined long short-term memory neural networks (LSTN) for network attack detection, and their experimental results show that the method can significantly improve detection efficiency. Shang et al.[27] used a stacked autoencoder fused with long short-term memory (LSTM) to solve the anomaly detection problem based on time series. Kasongo et al.[28] used the gated recurrent unit (GRU) instead of LSTM for wireless intrusion detection, and their experimental results show that the training time is lower than that of LSTM and there is no degradation in accuracy. Xu et al.[29] applied GRU to network intrusion detection and similarly demonstrated that, compared to LSTM, GRU has a lower training cost and does not degrade the performance of the model. Moreover, the performance of a BGRU is better than that of a GRU. Zhang et al.[30] fused the attention mechanism with the capsule network to make the model focus more on features that are more important to traffic anomaly detection, reduce the influence of noise, and improve the robustness of the model. Yin et al.[31]

improved random attention capsule network based on variable fusion[30] by introducing the idea of "residual" into the capsule network, which further improved the overall performance of the model. It can be seen that a reasonable mixture of multiple models can improve the anomaly detection rate.

The above analysis demonstrates the feasibility of applying deep learning models to network traffic anomaly detection, with promising results achieved. However, scholars need to conduct in-depth research to solve these problems in order to more effectively use the deep learning model to extract features and more reasonably use the raw network traffic data for research to improve the accuracy of network traffic anomaly detection.

In this paper, the network traffic anomaly detection method based on the combination of multiple models was studied, the network traffic anomaly detection from the algorithmic and data levels was analyzed, and the detection performance of the model was improved by combining the application of multiple deep learning models. The main contents and innovations of this paper are as follows.

1) From the perspective of spatiotemporal information fusion, a model for integrated detection of multidimensional information is designed using a combination of multi-depth models. It enriches the means of recognition of target information and improves the detection capability of abnormal information in complex environmental systems.

2) To address the problem of low performance of traffic anomaly detection due to the limitation of features extracted by a single deep learning model, a hybrid feature extraction model based on ResNet and BGRU was proposed in this paper to extract the spatial features of network traffic data and the temporal features of network traffic data, and the two features are integrated to conduct research of network traffic anomaly detection, so as to achieve a high classification accuracy.

3) For the characteristics of spatiotemporal multidimensional information structure, the application of an anomaly detection multi-head attention mechanism is designed to highlight the recognition ability of complex pattern information and improve the computational performance of the model.

3 ResNet-BGRU

Deep learning-based network traffic anomaly detection is practical through a large number of studies. In contrast deep learning models with a single structural metric can often only extract some of the features implied by network traffic. Therefore, when the model is applied, it may lead to the problem of low accuracy due to the limitations of the extracted features. To address this problem, ResNet-BGRU was proposed in this paper, which is a comprehensive model. Specifically, the model uses ResNet to extract the spatial features implied by network traffic and also uses BGRU to extract the temporal features implied by network traffic. After mixing the two features, a classification study of the model is conducted. The advantage of this approach is that it mixes both spatial and temporal features of network traffic to perform traffic clustering. In addition, after combining the two features, an attention mechanism is introduced in the model to highlight the importance of certain features to further enhance the model's adaptability.

3.1 ResNet-BGRU architecture

ResNet-BGRG learns both spatial and temporal features of network traffic data by using ResNets and BGRUs, and gives more weight to some important features after fusion by an attention mechanism, so as to strengthen the classification ability of the model. The overall architecture of the ResNet-BGRU model is shown in Fig. 1. The ResNet-BGRU model consists of three main components. The preprocessing module first processes the data sources used in this paper to enable them to be input into the neural network model. The feature extraction module inputs the preprocessed data into both the ResNet module and the BGRU module to extract spatial and temporal features, and then inputs the extracted spatial and temporal features into the attention module, giving more weight to the features

that contribute more to the task. In the abnormal traffic classification module, the integrated features of the traffic extracted by the second part of the traffic feature extraction module are input into the fully connected neural network and the softmax activation function is used to identify the traffic according to the task.

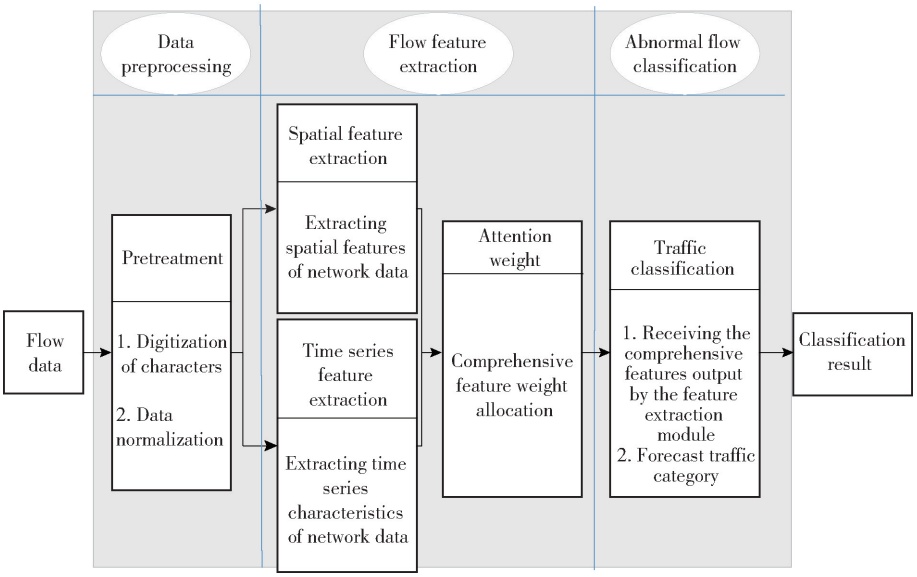


Fig. 1 Architecture of ResNet-BGRU

3.2 Preprocessing module

The data is preprocessed because the data input to the neural network model needs to be in the specified format, so the data contains character types that need to be turned into numbers, and then all the data needs to be normalized to reduce the computational costs of the model. This process does not change the dimensionality of the data, and the processed data can be input into the neural network model for training.

3.3 Feature extraction module

1) Spatial feature extraction

Spatial features of network traffic data are extracted by ResNet. When there are too many model layers, the issues of gradient disappearing, gradient explosion, and network deterioration can be efficiently avoided using the ResNet. Two-dimensional (2D) convolution is usually better at dealing with images and other related problems, while one-dimensional (1D) convolution is often used to deal with 1D sequence-related data. The traffic data in this paper is similar to text data, so this paper decides to use 1D convolution to extract the spatial features of the traffic. The structure diagram of the ResNet and the structure of the residual block (ResBlack) in this paper are shown in Figs. 2 and 3.

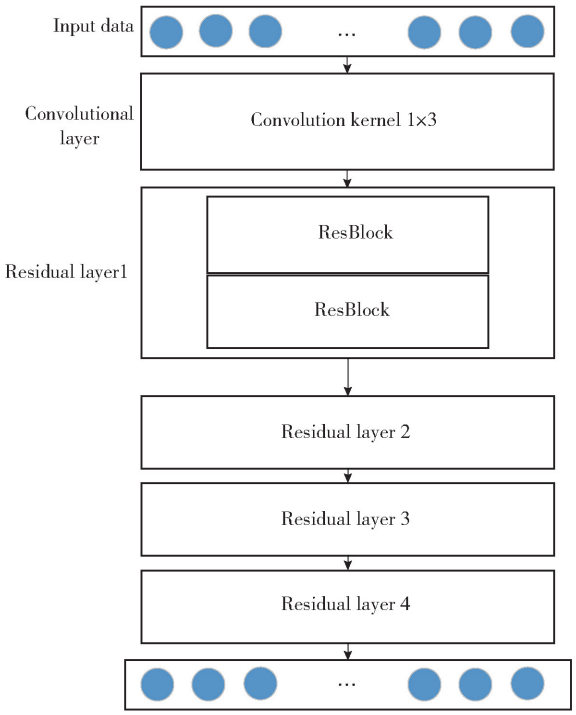


Fig. 2 Structure diagram of ResNet

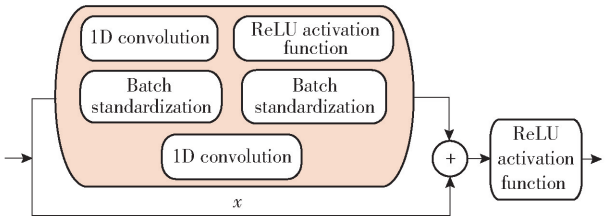


Fig. 3 Structure of ResBlock

The algorithm flow for feature extraction by a ResNet is as follows.

First, the raw flow data undergoes numerical processing and normalization to obtain the standardized input data $\boldsymbol{V}$, with dimensions $1 \times N$, where N represents the length of the time series. This data is then fed into a convolutional layer for feature extraction using 1D convolution. The convolution kernel size is set to $1 \times h$ (specifically 1×3 in this study), with a sliding stride of L. The stride L determines the step size of the kernel's movement, performing a convolution operation at each step until the entire sequence is traversed. The padding value is set as 1 in this paper. The padding value prevents information loss. The convolution is calculated as

$$s_i = f(wv_{i:i+h-1} + b) \tag{1}$$

where w represents the convolution kernel, which is the parameter that the model will eventually learn through continuous training, $v_{i:i+h-1}$ represents the local feature segment of input $\boldsymbol{V}$ for each convolutional scan, where the kernel window slides across the sequence, and b represents the bias value, $f(\cdot)$ represents the rectified linear unit (ReLU) activation function, which is

$$f(x) = \max\{0, x\} = \begin{cases} 0; & x < 0 \\ x; & x \geqslant 0 \end{cases} \tag{2}$$

Following the convolution layer, the output feature vector $\boldsymbol{s} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{n-h+1})$ are output to ResBlock. The element of feature is calculated as

$$\boldsymbol{s}_{l+1} = h(\boldsymbol{s}_l) + F(\boldsymbol{s}_l, w_l); \quad l = 1, 2, \ldots, n-h+1, \quad n = 1, 2, \ldots, N \tag{3}$$

In Eq. (3), $h(\boldsymbol{s}_l)$ is a 1×1 convolution operation that has no effect on model performance and is only used for dimensionality reduction or enhancement, and $F(\boldsymbol{s}_l, w_l)$ represents the output result of the feature vector $\boldsymbol{s}_l$ after the convolution operation of the ResBlock. The specific structure of the ResBlock is shown in Fig. 3. The ResBlock structure in this paper is a two-layer convolutional structure. The first layer is a convolutional layer with a 1×3 convolutional kernel, which is then input into the batch normalization layer to prevent overfitting of the model, and the second layer is also a convolutional layer with a 1×3 convolutional kernel. In this paper, four residual layers are chosen, and each residual layer contains two ResBlock structures. The features extracted from the four residual layers are fed into the pooling layer, which aims to reduce the computational time by reducing the neural network model parameters. In this paper, average pooling $A_{vg}(\cdot)$ is chosen to perform the pooling operation. The specific calculation is

$$S = A_{vg}(\boldsymbol{s}) \tag{4}$$

The output of the pooled feature vector is 128, this is used to fuse the temporal features extracted by the temporal feature extraction module. The main parameters of the ResNet are shown in Tabel 1.

Table 1 Main parameter of ResNet

Network layer	Operation	Input	Convolution kernel	Output
Convolutional layer 1	1D convolution × 1	1 × 121	1 × 3	16 × 121
Convolutional layer 1	1D convolution × 4	16 × 121	1 × 3	16 × 121
Convolutional layer 2	1D convolution × 4	16 × 121	1 × 3	32 × 61
Convolutional layer 3	1D convolution × 4	32 × 61	1 × 3	64 × 31
Convolutional layer 4	1D convolution × 4	64 × 31	1 × 3	128 × 16
Average pooling	Average pooling	128 × 16	–	128 × 1
Flatten	Flatten	128 × 1	–	128

2) Time sequence feature extraction

The traditional GRU can only use the information before the current position, but not the information after it, while in practice, the information at the current position is not only related to the previous information, but also may be closely related to the information after it. In this situation, bidirectional gated RNN that can effectively use not only the information before but also the information after is proposed for learning. BGRU is a combination of

forward GRU and backward GRU, which can ensure the information of the previous sequence and also pay attention to the information of the later sequence, so the BGRU was finally choosed to replace the traditional GRU model. Subsequent experiments also show that the performance of BGRU is better than that of the GRU model. The structure of the BGRU model in this paper is shown in Fig. 4.

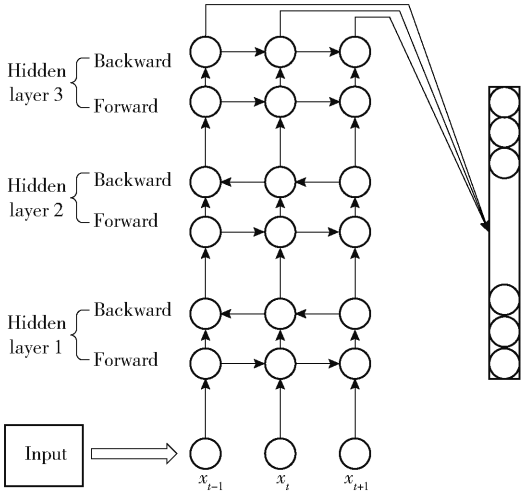


Fig. 4 Structure of the BGRU model

The number of neurons in the BGRU is 128, it contains three layers of BGRU. In Fig. 4, x_t represents the input value at the moment t, and the output value h_{t-1} at the previous moment, and the input value x_t at the current moment are output after the calculation of reset gate and update gate of GRU. In the forward computation, the reset gate determines the retention of historical information through sigmoid activation.

$$r_t = \sigma(w_r[h_{t-1}, x_t] + b_r) \tag{5}$$

where w_r represents the weight for the reset gate, h_{t-1} denotes the hidden state from the previous timestep, x_t is the input at current timestep t, $\sigma(\cdot)$ denotes the sigmoid activation function, and b_r is the corresponding bias term. The update gate, which controls the state updates, is computed as

$$z_t = \sigma(w_z[h_{t-1}, x_t] + b_z) \tag{6}$$

with w_z being the update gate weight and b_z is its bias term. The candidate hidden state is then generated through hyperbolic tangent transformation.

$$\tilde{h}_t = \tanh(w_h[r_t h_{t-1}, x_t] + b_h) \tag{7}$$

where w_h is the candidate state weight, b_h is its bias term. The final forward output combines the previous state and candidate state through the update gate.

$$\overrightarrow{h}_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t \tag{8}$$

where $\overrightarrow{h}_t$ indicates the forward GRU's output result at this moment.

The backward computation follows a similar structure but processes the sequence in reverse temporal order. The backward reset gate is calculated as

$$r'_t = \sigma(w'_r[h_{t+1}, x_t] + b'_r) \tag{9}$$

where w'_r and b'_r are the backward reset gate parameters, and h_{t+1} represents the hidden state from the next timestep. Similarly, the backward update gate is computed by

$$z'_t = \sigma(w'_z[h_{t+1}, x_t] + b'_z) \tag{10}$$

The backward candidate state is generated through

$$\tilde{h}'_t = \tanh(w'_h[r'_t h_{t+1}, x_t] + b'_h) \tag{11}$$

where w'_h and b'_h are the backward candidate state parameters, $\tanh(\cdot)$ represents the hyperbolic tangent function. The final backward output is obtained through

$$\overleftarrow{h}_t = (1 - z'_t) h_{t+1} + z'_t \tilde{h}'_t \tag{12}$$

where $\overleftarrow{h}_t$ denotes the output result of the backward GRU at this moment. In Eqs. (9) – (12), all weight matrices (w_r, w'_z, w_h, w'_r, w'_z, w'_h) and bias terms (b_r, b_z, b_h, b'_r, b'_z, b'_h) are learned parameters during training.

The output of this moment is obtained by fusing these two outputs and calculating them, specifically, is shown as

$$h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t] \tag{13}$$

After three layers of BGRU, the obtained temporal features are input to the fully connected layer, and the output of the fully connected layer is 128, which is used to fuse with the features output from the spatial feature extraction module.

3) Attention mechanism

The attention mechanism can weight certain important features so that the model can focus on those more important features. Therefore, the spatial features extracted by the ResNet is fused with the temporal features extracted by the BGRU, and then the mixed features are input into the multi-headed attention module to further extract more accurate and

comprehensive features of the network traffic. In this paper, the transformer's multi-headed attention mechanism was used, which consists of multiple self-attention modules. As illustrated in Fig. 5, its structure involves matrix multiplication (MatMul) to compute attention weights and an optional mask to prevent accessing future information during decoding.

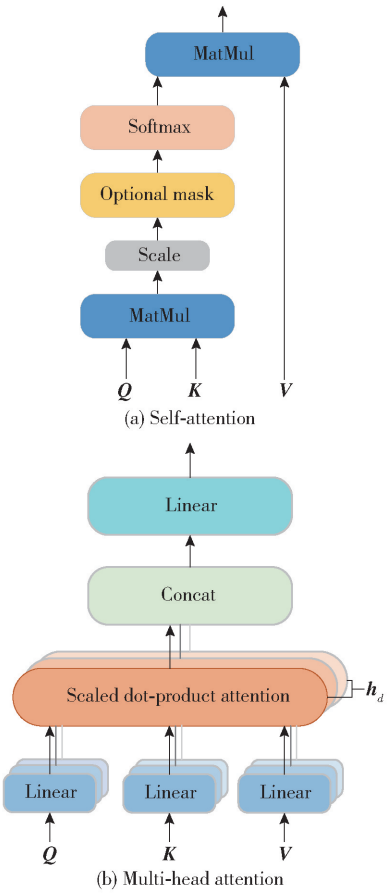


Fig. 5 Self-attention and multi-head attention

In Fig. 5, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ denote "query", "key", and "value", respectively. The scaled dot-product attention in the multi-head attention mechanism is a self-attentive calculation. The spatial and temporal features extracted from the traffic data are fused into the feature vector $\boldsymbol{s}_l$ and input into the multi-head attention, which is calculated as

$$\boldsymbol{Q} = \boldsymbol{w}_{\mathrm{q}} \boldsymbol{s}_l \tag{14}$$

$$\boldsymbol{K} = \boldsymbol{w}_{\mathrm{k}} \boldsymbol{s}_l \tag{15}$$

$$\boldsymbol{V} = \boldsymbol{w}_{\mathrm{v}} \boldsymbol{s}_l \tag{16}$$

$$\boldsymbol{h}_d = f_{\mathrm{s}}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d}}\right)\boldsymbol{V} \tag{17}$$

$$M_d(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = O_{\mathrm{C}}(\boldsymbol{h}_d,\ldots,\boldsymbol{h}_D)\boldsymbol{w}^o \tag{18}$$

where $O_{\mathrm{C}}(\cdot)$ denotes the concatenation of all attention heads' output vectors followed by linear transformation to generate the final features. $\boldsymbol{w}^o$ is the learnable parameter matrix in the multi-head attention mechanism used to integrate the outputs of each attention head.

In Eqs. (14) – (18), $\boldsymbol{w}_{\mathrm{q}}$, $\boldsymbol{w}_{\mathrm{k}}$, $\boldsymbol{w}_{\mathrm{v}}$, $\boldsymbol{w}^o$ are all parameters that can be learned and $\boldsymbol{h}_d$ denotes the output vector sequence of the dth self-attention, D denotes the number of self-attentive modules in the multi-headed $M_d(\cdot)$ attention module, which is set to 8 in this paper. The values calculated by multiple self-attention modules are connected to obtain the final output. $f_{\mathrm{s}}(\cdot)$ denotes the softmax activation function.

3.4 Classification module

After weighting the fused features using the attention mechanism, the output vectors are fed into the fully connected layer and classified using the activation function. The softmax activation function may generate overflow and underflow problems. In order to solve this problem and speed up the operation, the log_softmax activation function is chosen. Log_softmax is based on the softmax function and calculates its corresponding logarithmic values in the range of $(-\infty, 0)$, which is used to calculate the cross-entropy loss function. Log_softmax speeds up the operation and improves data stability. The log_softmax activation function is

$$\ln f(y_j) = \ln \frac{\mathrm{e}^{y_j}}{\sum_1^C \mathrm{e}^{y_j}} \tag{19}$$

where y_j is the output value of the jth node, and C is the total number of categories. The probability of the node belonging to each category is calculated using the log_softmax function, and the category with the highest probability is the one to which the sample belongs.

3.5 ResNet-BGRU

The best model parameters saved during model training are extracted using ResNet-BGRU hybrid features, and the data to be measured is fed into the trained neural network model. The flow chart of the model algorithm is shown in Fig. 6.

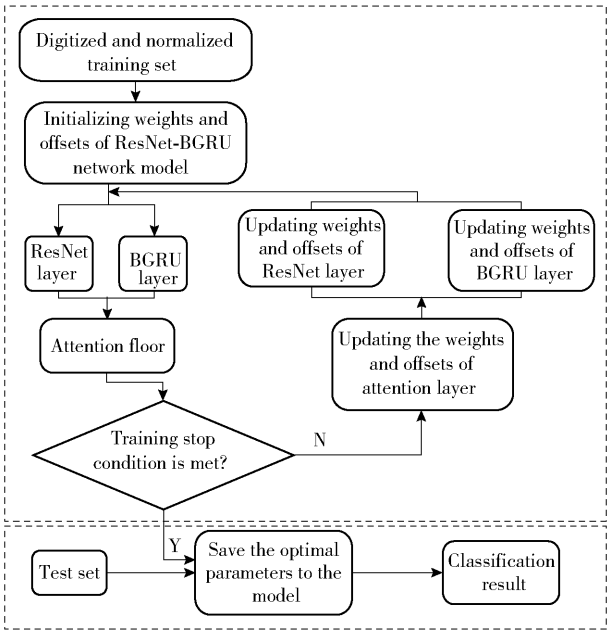


Fig. 6 ResNet-BGRU flow chart

Firstly, the original data is processed by character numerical and data normalization. Secondly, the weight parameters of the model are initialized, the processed data is input to the model for training, the spatial features are extracted by inputting the ResNet layer through Eqs. (1) – (4), and the temporal features are extracted by inputting the BGRU layer through Eqs. (5) – (12), and the extracted two features are fused. The fused features are input into the attention module. The attention weight is calculated through the attention layer using Eqs. (14) – (18), so as to assign more weight to the more important features, after which the model parameters are continuously adjusted by back propagation until the end condition is satisfied. The best parameters are saved, and the best parameter model is used to test the data for classification to determine the category to which the test data belongs.

4 Experimental results and analysis

4.1 Experimental environment

In this paper, the experimental study is conducted in a Windows operating system environment, and PyTorch is used as the deep learning framework, the configuration is shown in Table 2.

Table 2 Experimental environment

Name of experimental environment	Configuration information
Operating system	Windows 10
Memory/GB	16
Programming language	Python3.6
Deep learning framework	PyTorch

4.2 Performance evaluation metrics

The evaluation metrics for network traffic anomaly detection in this paper are precision, accuracy, recall, and F1-score. Additionally, the confusion matrix is introduced, which is shown in Table 3.

Table 3 Confusion matrix definition

Predicted value	Real value	
	0	1
0	TP	FP
1	FN	TN

In Table 3, TP indicates the count where both the predicted type and the true type are 0, TN indicates the count where the true type is 1 and the predicted type also happens to be 1, FP indicates the count where the true type is 1, but the model predicts the type to be 0, and FN indicates the count where the true type is 0, but the model predicts the type to be 1.

Based on the confusion matrix, the details of how these four metrics are calculated.

Precision P_r is the ratio of the number of traffic data correctly identified to a specified type to all network traffic data classified to that type, with a precision ranging from 0 to 1, and is calculated as

$$P_r = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{20}$$

where N_{TP} denotes TP, N_{FP} denotes FP.

Accuracy A_c is the proportion of correctly identified network traffic data to the overall test traffic data, and the value of accuracy ranges from 0 to 1, and is calculated as

$$A_c = \frac{N_{TN} + N_{TP}}{N_{TP} + N_{TN} + N_{FP} + N_{FN}} \tag{21}$$

where N_{TN} denotes TN, N_{FN} denotes FN.

Recall R_c is the ratio of traffic data correctly classified to a specified type to all traffic data belonging to that category, and the value of recall ranges from 0 to 1, and is calculated as

$$R_c = \frac{N_{TP}}{N_{FN} + N_{TP}} \tag{22}$$

F1-score F_1 is a composite value calculated by using the model accuracy and recall, the higher the score, the better the performance of the model, and the value of F1-score ranges from 0 to 1, is calculated as

$$F_1 = \frac{2P_r R_c}{R_c + P_r} \tag{23}$$

4.3 Experimental data description

Currently, most network traffic anomaly detection data is not publicly available in many research datasets because of privacy issues. In this paper, the NSL-KDD and UNSW-NB15 datasets were chosen which are currently publicly available and most commonly used in the field of network traffic anomaly detection, to conduct the experimental study.

The original version of the NSL-KDD dataset, KDD-Cup99, is a dataset with nearly 5 million entries, is a relatively large amount of data. And there are a large number of redundant features in the KDD-Cup99 dataset, and it contains a large amount of duplicate data. The model performance could not be evaluated well, so all duplicates present in the original data were removed from the NSL-KDD dataset, and only one copy of each duplicate record was kept, thus keeping the balance of each type of data. In the NSL-KDD dataset, there are 41 statistical features in total, containing 34 continuous features and 7 discrete features. The dataset includes 39 malicious attack types, 17 of which only appear in the NSL-KDD testing set, so as to better match the realistic real-world environment. These 39 attack types are grouped into four major categories, DoS, surveillance or probe (Probe), remote to local (R2L), and user to boot (U2R). The specific category labels and the distribution of the number of categories in the training and testing sets are shown in Tables 4 and 5.

Table 4 NSL-KDD dataset attack category

Attack category	Training set label	Testing set label
DoS	Back, neptune, smurf, teardrop, land, pod	Apache2, mailbomb, processtable
Probe	Satan, portsweep, ipsweep, nmap.	Mscan, saint.
R2L	Warezmaster, warezclient, ftpwrite, guesspassword, imap, multihop, phf, spy	Sendmail, named, snmpgetattack, snmpguess, xlock, xsnoop, worm
U2R	Rootkit, bufferoverflow, loadmodule, perl	Httptunnel, ps, sqlattack, xterm

Table 5 Distribution of the number of labels in the NSL-KDD dataset

Attack category	Number of the training set's label	Number of the testing set's label
Normal traffic	67 345	9 711
DoS	45 926	7 460
Probe	11 655	2 421
U2R	52	67
R2L	995	2 885

The UWSW-NB15 dataset is generated by a tool simulation and contains normal activity as well as synthetic attack behavior data. It contains more attack types compared to the NSL-KDD dataset, published by the Australian Centre for Cyber Security (ACCS), and contains comma-separated values files for both the testing and training sets. It contains a total of 175 341 training data and 82 332 testing data. The UWSW-NB15 dataset contains 9 attack types and one normal type, totaling 10 categories. Table 6 shows the detailed categories of this dataset and the distribution of the numbers in the training and testing sets.

Table 6 UNSW-NB15 dataset label description and data distribution

Attack category	Training set	Testing set
Normal attack	56 000	37 000
Analysis attack	2 000	677
Backdoor attack	1 746	583
DoS	12 264	4 089
Exploits attack	33 393	11 132
Fuzzers attack	18 184	6 062
Generic attack	40 000	18 871
Reconnaissance attack	10 491	3 496
Shellcode attack	1 133	378
Worm attack	130	44

4.4 Data preprocessing

The neural network model requires data in a specific format, so data preprocessing is required for the NSL-KDD as well as the UNSW-NB15 datasets.

1) NSL-KDD data preprocessing

a) Character digitization

For the NSL-KDD dataset, each sample initially contains 42 raw features. Taking a representative sample [0,UDP,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,0.00,0.00,0.00,0.00,0.00,0.00,normal] as an example, the features can be categorized into three groups. Firstly, the basic connection features include duration the numerical value 0 in the first column, protocol type (the categorical value UDP in the second column), service type (private in the third column), and connection flag (SF in the fourth column). The protocol type contains three variants, transmission control protocol (TCP), user datagram protocol (UDP), and Internet control message protocol (ICMP), the service type covers 70 different network services, while the connection flag has 11 possible values. These categorical features are converted into numerical representations through one-hot encoding. Secondly, the traffic statistics features, spanning from the 5th to the 41st columns, consist of direct numerical features such as source-to-destination byte counts, login attempts, and error fragment numbers, along with derived features that reflect behavioral patterns within time windows, including connection counts to the same target host (values 1,1 in the 23rd – 24th and traffic rate metrics (e.g., value 1.00 in the 29th column). These features are preserved in their original form or undergo standardization. Thirdly, the label column (the 42nd column) indicates sample categories, where "normal" represents regular connections while other values correspond to various attack types. The labels are mapped to numerical codes for model processing. During analysis, the feature "num_outbound_cmds" was found to contain exclusively zero values and was consequently removed due to its lack of discriminative value. The core preprocessing steps involve one-hot encoding of three categorical features (expanding into 84 binary dimensions), retention of 38 original numerical features, and ultimately generating a 121-dimensional feature vector combined with a 1D label. The attack category labels are shown in Table 7.

Table 7 NSL-KDD dataset category label coding table

Attack category	Coding
Normal	0
DoS	1
Probe	2
R2L	3
U2R	4

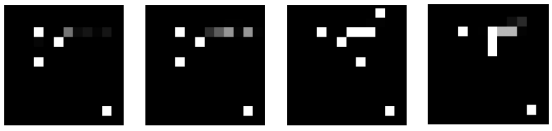
b) Normalization

Different evaluation metrics usually have different units of measurement, which can lead to a large difference in the magnitude of the values, thus affecting the analysis of the subsequent model results. In order to mitigate the impact of different units of measurement and reduce the calculated quantities of the model, the flow data are standardized to place each indicator at the same order of magnitude. In this paper, min-max normalization is used to convert all values of the sample data to the range of [0, 1]. The min-max normalization is

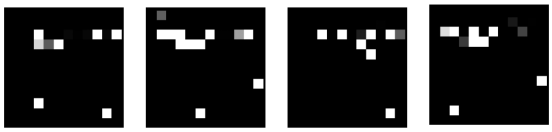
$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{24}$$

In Eq. 24, $x_{\max}$ is the maximum value of the feature, and $x_{\min}$ is the minimum value of the feature. x represents the data of a feature in the current sample to be transformed and x^* represents the transformed data.

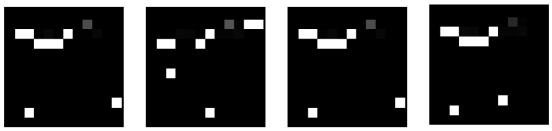
The data is processed into 121-dimensional data with a value domain of [0,1]. For visualization, the 121-dimensional data of feature length is transformed into an 11×11 matrix. In Fig. 7, the sample images of the five labels of NSL-KDD are shown. Fig. 7 shows that among the same kind of samples, there are similar feature distributions, such as the feature distribution of R2L. There are also clear distinctions between different samples, such as the distinct block distributions between R2L and U2R.



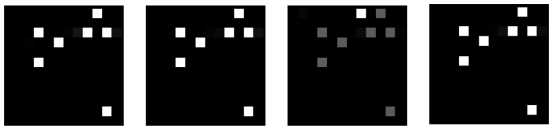
(a) Visualization of normal feature



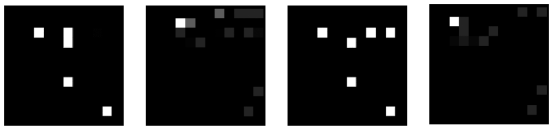
(b) Visualization of Probe feature



(c) Visualization of DoS feature



(d) Visualization of R2L feature



(e) Visualization of U2R feature

Fig. 7 NSL-KDD data visualization

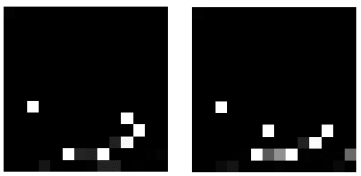
2) UNSW-NB15 data preprocessing

A sample of data from the dataset UNSW-NB15 is shown as [0.121 478, TCP, -, FIN, 6, 4, 258, 172, 74.087 49, 252, 254, 14 158.942 38, 8 495.365 234, 0, 0, 24.295 6, 8.375, 30.177 547, 11.830 604, 255, 621 772 692, 2 202 533 631, 255, 0, 0, 0, 43, 43, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, normal, 0]. From the data sample, it can be seen that the data sample includes three character type features: protocol, service, and state (i.e., the 2nd, 3rd, and 4th columns). Among them, protocol contains 133 types of representation, service contains 13 types, and state contains 9 types of representation. We treat them all in the same way as the NSL-KDD dataset. The last bit is the label divided into good label by the original data, and it will be removed. Finally, the 196-dimensional features and 1D tagging bit, totaling 197 dimensions, are obtained. The classification labels are shown in Table 8.

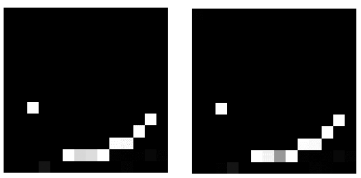
Table 8 UNSW-NB15 dataset category label coding table

Attack category	Coding
Normal attack	0
Worm attack	1
Analysis attack	2
Backdoor attack	3
DoS	4
Exploit attack	5
Fuzzers attack	6
Generic attack	7
Reconnaissance attack	8
Shellcode attack	9

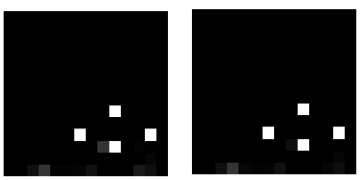
The preprocessed UNSW-NB15 dataset is visualized by transforming it into a 14 × 14 grayscale matrix with 196 feature dimensions. The visualized results are shown in Fig. 8, which are grayscale maps of 10 labels from the UNSW-NB15 dataset. Two random samples from each label category were taken for presentation in this paper. As can be seen from Fig. 8, the difference between the different samples can be visualized by transforming the samples into grayscale plots. For example, the four distinct blocks of analysis have a diamond-shaped distribution structure, while the bottom row of generic attack has a distinct row of blocks. Thus the data has practical classification feasibility.



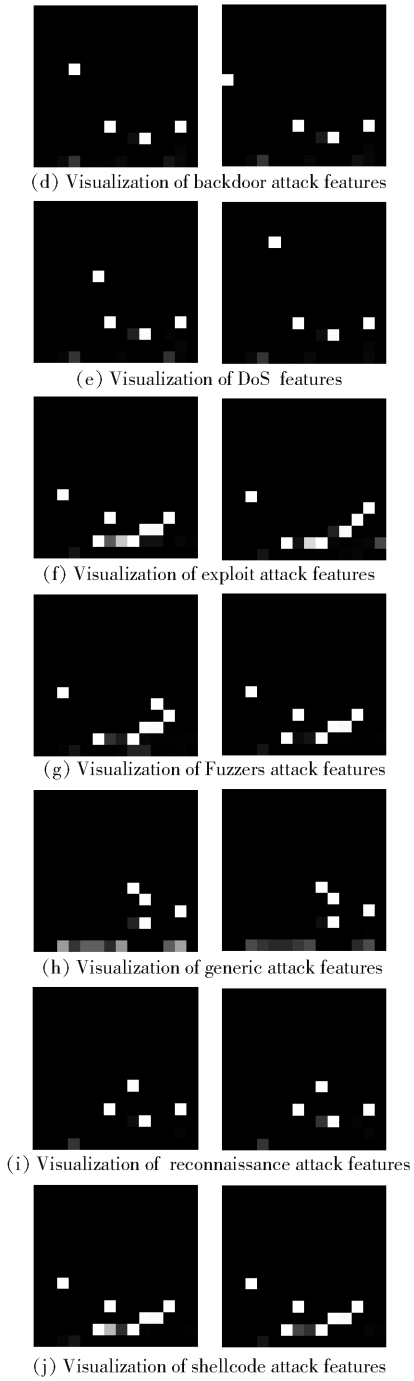
(a) Visualization of normal attack features



(b) Visualization of worm attack features



(c) Visualization of analysis attack features


(d) Visualization of backdoor attack features

(e) Visualization of DoS features

(f) Visualization of exploit attack features

(g) Visualization of Fuzzers attack features

(h) Visualization of generic attack features

(i) Visualization of reconnaissance attack features

(j) Visualization of shellcode attack features

Fig. 8 UNSW-NB15 dataset visualization

4.5 Experimental results and analysis

1) Determination of model parameter

In this paper, some parameters were tested in 10-classification experiments on the UNSW-NB15 dataset to investigate the effects of different parameters on the model's performance. Firstly, the number of different BGRU layers has an impact on the experimental results. In order to find the optimal number of layers, one, two, three, and four layers are chosen to compare the classification accuracy while keeping other parameters constant, and the performance difference between the traditional GRU model and the BGRU model are compared. The experimental results are shown in Fig. 9. As can be seen from Fig. 9, the overall performance of BGRU is better than the traditional GRU model, which is expected because BGRU can learn more feature information not only from the previous information but also from the next moment. In addition, the accuracy of the BGRU starts to decrease at four layers. Therefore, three-layer BGRU is chosen as the model structure.

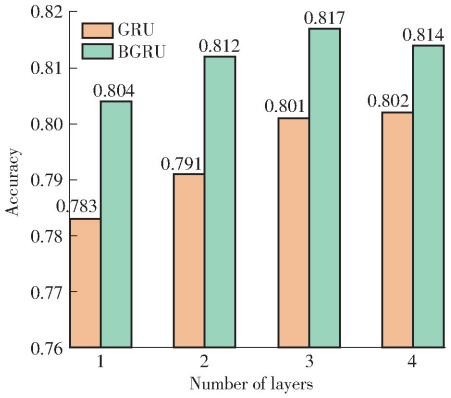


Fig. 9 Effect of different layers on accuracy of ResNet-BGRU

Secondly, the number of ResBlock also affects the accuracy of the model, so the effect of different numbers of BGRU layers and different numbers of residual layer combinations on the classification accuracy of the ResNet-BGRU model is tested, as shown in Table 9. From Table 9, it can be found that the highest accuracy is the combination of 3 layers of BGRU and 4 layers of ResNet. Therefore, the final model chosen in this paper is the combined model of a 4-layer ResNet and a 3-layer BGRU.

Table 9 Accuracy of ResNet-BGRU under different structures

Number of residual layers	Accuracy		
	1-layer BGRU	2-layer BGRU	3-layer BGRU
2	0.780	0.785	0.792
3	0.801	0.810	0.812
4	0.804	0.812	0.817

Finally, the performance of the basic model (without attention mechanism) is tested as a baseline, and the results show a classification accuracy of 0.809 (see Table 10). Subsequently, the impact of the attention mechanism on the performance of the ResNet-BGRU is explored. It can be seen that ResNet-BGRU performance is best when 8 heads are used, so the number of multi-head attention heads is chosen as 8. In addition, it can be found that ResNet-BGRU performance decreases when the self-attention mechanism is used, which may be because self-attention only performs one calculation, resulting in too much attention being paid to some unimportant features, while multi-head attention can be calculated by multiple heads to finally. The results are integrated to prevent individual heads from focusing too much on certain features.

Table 10 Effect of different attention on accuracy of ResNet-BGRU

Attentions type	Accuracy
Without attention	0.809
Self-attention	0.798
Multi-head attention-8	0.817
Multi-head attention-16	0.815

In order to improve efficiency, the loss function adopts the cross-entropy loss function, the batch_size is chosen to be 128, the Adam optimizer is selected to update the network parameters, the learning rate is chosen to be 0.01, and the learning rate decay coefficient is changed to 90% of the original learning rate every 2 epochs. Table 11 shows the parameter setting of ResNet-BGRU.

Table 11 Parameter setting of ResNet-BGRU

Parameter	Value
Epoch	200
Batch size	128
Learning rate	0.01
Decay coefficient	0.9
Optimizer	Adam
Activation function	ReLU
Loss function	Cross-entropy

2) Experimental results on NSL-KDD dataset

a) Binary classification experiment

In order to compare with models such as support vector machine (SVM), *K*-nearest neighbor (KNN), decision tree (DT), random forest (RF), and AlertNet[32], the same experiments are conducted on the dataset using two branches of deep learning models without changing any parameter, and tandem experiments are performed with both models, several classical machine learning models are implemented as comparison experiments. In this paper, the normal and abnormal binary classification experiments on the NSL-KDD dataset were investigated. The experimental results, as shown in Table 12, demonstrate that ResNet-BGRU outperforms other models across all performance metrics. The hybrid deep learning model AlertNet experiences a decrease in overall accuracy due to the integration of different model settings. The model with a tandem structure outperforms the single deep learning model but is lower than ResNet-BGRU parallel model, because the spatial features of the original traffic data are extracted first, and the extracted features are passed to the next model, which may destroy the temporal features implicit in the original traffic data. This leads to a lower model accuracy than the ResNet-BGRU. While the machine learning models of DT, RF, and KNN have almost the same performance on NSL-KDD dataset for binary classification, the SVM has the worst performance. Compared with machine learning models and deep learning models of both branches, the performance of ResNet-BGRU is the best in binary classification, reaching 85.5% in accuracy and 85.2% in F1-Score.

Table 12 Comparison of the binary classification results of different models on the NSL-KDD dataset

Model	Accuracy	Precision	Precision recall	F1-score
ResNet	0.841	0.855	0.841	0.848
BGRU	0.823	0.831	0.823	0.830
DT	0.775	0.810	0.775	0.775
RF	0.775	0.836	0.775	0.772
SVM	0.747	0.797	0.747	0.744
KNN	0.778	0.818	0.778	0.777
R-BGRU	0.846	0.860	0.846	0.849
AlertNet	0.801	0.692	0.769	0.807
ResNet-BGRU	0.855	0.875	0.855	0.852

b) Multi-classification experimental results

Only recognizing normal and abnormal traffic often does not meet the practical needs, and different treatments should be available for different types of abnormal traffic. And the binary classification task often does not fully express the performance of the model, therefore, an experimental study of 5-classification is conducted in this paper on the NSL-KDD dataset. The experimental results are shown in Table 13. It can be found that the performance of ResNet-BGRU with integrated feature extraction after combining the advantages of the two deep learning models is significantly better than that of the single deep learning model, and ResNet-BGRU with the parallel approach outperforms the model with the tandem hybrid approach and the hybrid deep learning model AlertNet. In the comparison with the traditional machine learning models, it can be seen that the RF performs the worst on the NSL-KDD dataset, the SVM performs better on the multiclassification of the NSL-KDD dataset, and ResNet-BGRU still has the best results among all models for the multiclassification task.

Table 13 Comparison results 5-classification of different models on the NSL-KDD dataset

Model	Accuracy	Precision	Precision recall	F1-score
ResNet	0.823	0.854	0.823	0.797
BGRU	0.819	0.851	0.819	0.810
DT	0.763	0.767	0.763	0.728
RF	0.746	0.813	0.746	0.710
SVM	0.768	0.814	0.768	0.732
KNN	0.759	0.800	0.759	0.713
R-BGRU	0.831	0.850	0.831	0.822
AlertNet	0.785	0.810	0.785	0.765
ResNet-BGRU	0.840	0.860	0.840	0.831

In addition, in order to visualize the prediction of ResNet-BGRU on 5-classification of samples, the multi-category confusion matrix is shown in Fig. 10. In Fig. 10, the color intensity on the diagonal of the confusion matrix indicates the number of correctly classified samples (darker colors represent higher counts), while the adjacent color scale visually maps the numerical values to colors, aiding in interpreting the meaning of each color in the matrix. From Fig. 10, it can be seen that for normal, Probe, and DoS, ResNet-BGRU has a strong classification ability and most of the samples can be correctly predicted. While for U2R, with only 52 training samples, ResNet-BGRU is less capable of correctly classifying because it does not learn enough features. The F1-score can show the overall performance of the models in terms of precision and recall. Therefore, the F1-score of each model in each category is shown in Fig. 11. It is obvious from Fig. 11 that ResNet-BGRU outperforms ResNet, BGRU, tandem models, and machine learning models on Normal, Probe, DoS, and R2L, but all models perform worse on the type of U2R, which is because U2R has only 52 samples on the training set and U2R does not have enough to learn its features. In contrast, SVM performs slightly better on U2R with only a small number of samples.

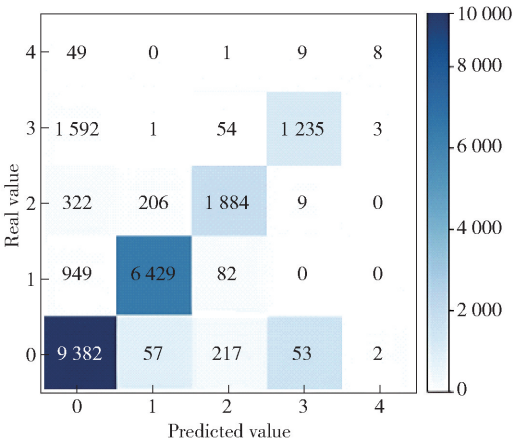


Fig. 10 Confusion matrix of ResNet-BGRU on the NSL-KDD dataset

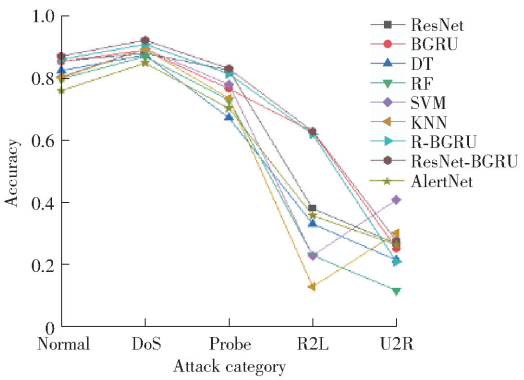


Fig. 11 F1-score of different models in each category

3) Experimental results on UNSW-NB15 dataset

a) Binary classification experiment

Similarly, an experimental study of binary classification is first conducted on the UNSW-NB15 dataset to observe the performance of each model on binary classification. The experimental results are shown in Table 14. From Table 14, it can be seen that the overall binary classification accuracy of the deep learning models on the UNSW-NB15 dataset is greater than that of the traditional machine learning models, and the gap between the machine learning models is not large, with SVM performing the worst in binary classification. The performance of the combined model is significantly superior to that of the individual ResNet and BGRU models. Moreover, ResNet-BGRU surpasses both the tandem model and the hybrid deep learning model, achieving the highest scores across all four performance metrics.

Table 14 Experiment results of binary classification of each model on the UNSW-NB15 dataset

Model	Accuracy	Precision	Precision recall	F1-score
ResNet	0.851	0.880	0.851	0.848
BGRU	0.846	0.877	0.846	0.841
DT	0.847	0.861	0.847	0.844
RF	0.846	0.874	0.846	0.840
SVM	0.810	0.858	0.810	0.799
KNN	0.843	0.862	0.843	0.839
R-BGRU	0.858	0.882	0.858	0.852
AlertNet	0.784	0.844	0.725	0.820
ResNet-BGRU	0.862	0.886	0.862	0.857

b) Multi-category experiment

The results of 10-classification for each model on UNSW-NB15 dataset are shown in Table 15. From Table 15, it can be seen that the accuracy of ResNet-BGRU is higher than that of the single deep learning model using only ResNet and BGRU, the tandem model, and the traditional machine learning model in terms of 10-classification and 0.1% lower than that of the BGRU model in terms of accuracy. However, the accuracy often cannot express the comprehensive performance of the model, and the F1-score, can better express the comprehensive performance of the model in terms of accuracy and recall. The precision and F1-score of the hybrid deep learning model AlertNet is relatively close to those of ResNet-BGRU. F1-score of ResNet-BGRU is 0.805, which is the highest among all models. Therefore, it can be demonstrated that ResNet-BGRU shows good performance on the 10-classification tasks on the UNSW-NB15 dataset.

Table 15 Experiment results of 10-classification of each model on the UNSW-NB15 dataset

Model	Accuracy	Precision	Precision recall	F1-score
ResNet	0.789	0.810	0.789	0.788
BGRU	0.781	0.815	0.781	0.784
DT	0.683	0.768	0.683	0.716
RF	0.722	0.813	0.722	0.747
SVM	0.690	0.804	0.690	0.701
KNN	0.713	0.781	0.713	0.739
R-BGRU	0.799	0.797	0.799	0.788
AlertNet	0.660	0.623	0.660	0.596
ResNet-BGRU	0.817	0.814	0.817	0.805

To clearly show the prediction of ResNet-BGRU for each class of samples, the confusion matrix is shown in Fig. 12.

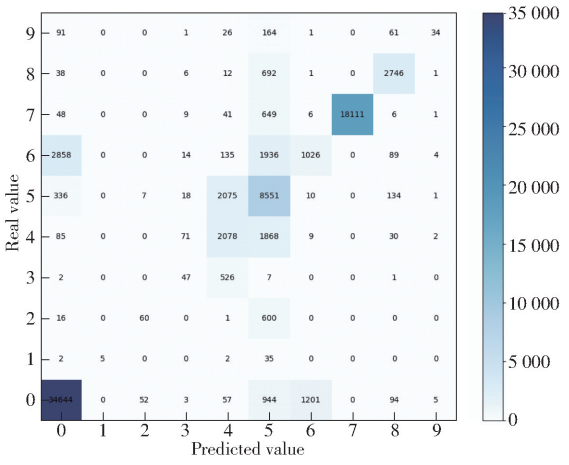


Fig. 12 Multi-classconfusion matrix on UNSW-NB15 dataset

From Fig. 12, it can be seen that for normal attack, exploit attack, and generic attack, which have a large number of samples, ResNet-BGRU has a strong classification ability, while for worm attack, which

have a small number of samples, these samples do not learn their features well, and most of their samples are predicted to have exploits attack. The grayscale plot shows that the grayscale plot of the worm attack samples has many similarities with the grayscale plot of the exploit attack samples, which may lead to their features being more similar, and thus most of the worm attacks are predicted to be exploit attacks. Also, it can be seen from the confusion matrix that 100% accuracy for generic attack can be achieved.

In Fig. 13, the F1-scores of each model in each category are shown. From Fig. 13, it can be seen that the classification performance of each model on generic attack is comparable, and they all achieve relatively high F1-scores. It can also be seen from the grayscale plot that the last row of pixels in generic attack has a great deal of differentiation compared to other categories, so all models can find the generic attack type well. On Fuzzers attack and shellcode attack, ResNet-BGRU has poor F1-score compared to the other models. However, for the other 7 types, ResNet-BGRU maintains the highest F1-scores.

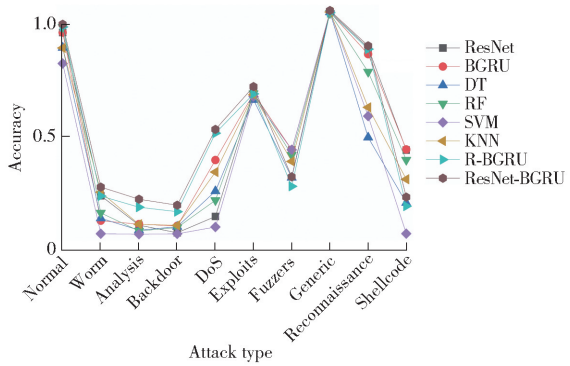


Fig. 13 F1-scores of differend attack type of each model on the UNSW-NB dataset

In the realm of deep learning, time-series and spatial features are often regarded as distinct aspects of input data, collectively forming the input vectors for deep neural networks. However, the approach to handling these and the architectural design significantly impact a model's capacity to learn features and make predictions. The hybrid feature extraction model ResNet-BGRU, leveraging ResNet and BGRU, adopts an effective methodology in dealing with these dual characteristics, with its advantages further elaborated as follows.

Firstly, specialized feature extraction, ResNet and BGRU are optimized to target spatial and temporal features, respectively. ResNet, through its residual learning framework, focuses on extracting static and spatially-related features from network traffic, whereas BGRU employs its gating mechanisms to efficiently capture dynamic changes within time series. This specialized approach enhances the exploitation of complex patterns within the data.

Secondly, enhanced feature fusion, by merging the spatial features extracted by ResNet with the temporal features captured by BGRU, ResNet-BGRU retains information from each type while boosting its composite representation of network traffic. This fusion strategy contributes to improved precision in identifying anomalous behavior.

Finally, ResNet-BGRU facilitates the consolidation of multi-dimensional information within network traffic data by integrating the strengths of two deep learning paradigms. This goes beyond what a single-model approach can achieve, enabling the model to concurrently analyze data from spatial and temporal perspectives, thereby capturing anomalies more comprehensively.

In summary, ResNet-BGRU hybrid feature extraction model, through its specialized extraction, enhanced fusion of features, and integration of multi-dimensional information, demonstrates its superiority over conventional models, offering a robust framework for detecting anomalies in network traffic.

5 Conclusions

In this paper, a hybrid feature extraction model ResNet-BGRU was explored and proposed. The model utilizes ResNet to extract spatial features contained in network traffic, while simultaneously employing BGRU to extract temporal features. By fusing these two types of features, the model performs network traffic anomaly detection. The performance of ResNet-BGRU on UNSW-NB15 and NSL-KDD datasets was evaluated. Experimental results show that ResNet-BGRU model achieves an accuracy of 84% for 5-classification on the

NSL-KDD dataset and 81.7% for 10-classification on the UNSW-NB15 dataset, outperforming various compared models. Despite the promising results achieved by ResNet-BGRU in anomaly detection tasks, there remain several limitations and areas for improvement. On one hand, the model's ability to generalize and be applied to more complex attack types and network environments needs further validation. Future research should focus on utilizing more diverse datasets for training and exploring regularization techniques or ensemble learning methods to improve the model's generalization capability. On the other hand, regarding computational efficiency optimization, future research can concentrate on optimizing the model structure and reducing computational complexity to improve the model's feasibility in practical deployments.

Acknowledgements

This work was supported by the the China Postdoctoral Science Foundation (2019M651263).

References

[1] AHMETOGLU H, RESUL D. A comprehensive review on detection of cyber-attacks: data sets, methods, challenges, and future research directions. Internet of Things, 2022, 20: Article 100615.

[2] FU Y, WANG K, DUAN X Y, et al. Survey of research on abnormal traffic detection for software defined networks. Journal on Communications, 2024, 45(3): 208 – 226 (in Chinese).

[3] SUN H L, LONG X, HAN L S, et al. Overview of anomaly detection techniques for industrial Internet of things. Journal on Communications, 2022, 43(3): 196 – 210 (in Chinese).

[4] DUAN X Y, FU Y, WANG K. Network traffic anomaly detection method based on multi-scale characteristic. Journal on Communications, 2022, 43(10): 65 – 76 (in Chinese).

[5] GU W, XING H Y, HOU T H. Abnormal traffic detection method based on traffic spatial-temporal features and adaptive weighting coefficients. Journal of Electronics & Information Technology, 2024, 46(6): 2647 – 2654 (in Chinese).

[6] HEIDARI A, JABRAEIL JAMALI M A. Internet of things intrusion detection systems: a comprehensive review and future directions. Cluster Computing, 2022, 26: 3753 – 3780.

[7] BHATTACHARYYA D K, KALITA J K. Network anomaly detection: a machine learning perspective. Boca Raton, FL, USA: CRC Press, 2019.

[8] LENG Y L, ZOU X Y. Network abnormal traffic detection model based on CNN-BiBASRU-AT. Microelectronics & Computer, 2024, 41(1): 93 – 99 (in Chinese).

[9] XIAO B, GAN Y, WANG M, et al. Network abnormal traffic detection based on port attention and convolutional block attention module. Journal of Computer Applications, 2024, 44(4): 1027 – 1034 (in Chinese).

[10] SONG Y, HOU B N, CAI Z P. Network intrusion detection method based on deep learning feature extraction. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2021, 49(2): 115 – 120 (in Chinese).

[11] MOUSTAFA N, HU J K, SLAY J. A holistic review of network anomaly detection systems: a comprehensive survey. Journal of Network and Computer Applications, 2019, 128: 33 – 55.

[12] IGLESIAS F, FERREIRA D C, VORMAYR G, et al. NTARC: a data model for the systematic review of network traffic analysis research. Applied Sciences, 2020, 10(12): Article 4307.

[13] SMRITHY G S, BALAKRISHNAN R. A statistical-based light-weight anomaly detection framework for wireless body area networks. The Computer Journal, 2022, 65(7): 1752 – 1759.

[14] RESENDE P A A, DRUMMOND A C. A survey of random forest based methods for intrusion detection systems. ACM Computing Surveys, 2018, 51(3): Article 48.

[15] JAW E, WANG X. A novel hybrid-based approach of snort automatic rule generator and security event correlation (SARG – SEC). PeerJ Computer Science, 2022, 8: Article e900.

[16] ZHANG S N, SUN B. Research on network anomaly detection method based on machine learning. Journal of Jilin University (Information Science Edition), 2021, 39(6): 732 – 742 (in Chinese).

[17] HUANG Q W, LI L Y, SHEN F K, et al. Network anomaly traffic detection based on ensemble feature selection. Journal of East China Normal University (Natural Science), 2021, (6): 100 – 111 (in Chinese).

[18] JIA F, KONG L Z. Intrusion detection algorithm based on convolutional neural network. Transaction of Beijing Institute of Technology, 2017, 37(12): 1271 – 1275. (in Chinese)

[19] ZHAO Y, WANG L B, CHEN J J, et al. Network anomaly detection based on federated learning. Journal of Beijing University of Chemical Technology (Natural Science), 2021, 48(2): 92 – 99 (in Chinese).

[20] XIAO Y H, XING C, ZHANG T N, et al. An intrusion detection model based on feature reduction and convolutional neural networks. IEEE Access, 2019, 7: 42210 – 42219.

[21] ZHANG S C, XIE X Y, XU Y. Intrusion detection method based on a deep convolutional neural network. Journal of Tsinghua University (Science and Technology), 2019, 59(1): 44 – 52 (in Chinese).

[22] LIN S Z, SHI Y, XUE Z. Character-level intrusion detection based on convolutional neural networks. Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN'18), 2018, Jul 8 – 13, Rio de Janeiro, Brazil. Piscataway, NJ, USA: IEEE, 2018: 8p.

[23] LI Z H, XIANG Z J, GONG W J, et al. Unified model for collective and point anomaly detection using stacked temporal convolution networks. Applied Intelligence, 2022, 52(3): 3118 – 3131.

[24] LI C P, WANG J L, YE X Z. Using a recurrent neural network and restricted Boltzmann machines for malicious traffic detection. NeuroQuantology, 2018, 16(5): 823 – 831.

[25] MANICKAM M, RAMARAJ N, CHELLAPPAN C. A combined

PFCM and recurrent neural network-based intrusion detection system for cloud environment. International Journal of Business Intelligence and Data Mining, 2019, 14(4): 504 – 527.

[26] FU Y S, LOU F, MENG F Z, et al. An intelligent network attack detection method nased on RNN. Proceedings of the IEEE 3rd International Conference on Data Science in Cyberspace (DSC'18), 2018, Jun 18 – 21, Guangzhou, China. Piscataway, NJ, USA: IEEE, 2018: 483 – 489.

[27] SHANG W L, SHI H, ZHAO J M, et al. An anomaly detection method of process data based on SAE-LSTM. Acta Electronica Sinica, 2021, 49(8): 1561 – 1568 (in Chinese).

[28] KASONGO S M, SUN Y X. A deep gated recurrent unit based model for wireless intrusion detection system. ICT Express, 2021, 7(1): 81 – 87.

[29] XU C Y, SHEN J H, DU X, et al. An intrusion detection system using a deep neural network with gated recurrent units. IEEE Access, 2018, 6: 48697 – 48707.

[30] ZHANG X L, YIN S L. Intrusion detection model of random attention capsule network based on variable fusion. Journal on Communications, 2020, 41(11): 160 – 168 (in Chinese).

[31] YIN S L, ZHANG X L, ZUO L Y. Intrusion detection system for dual route deep capsule network. Journal of Computer Research and Development, 2022, 59(2): 418 – 429 (in Chinese).

[32] ZHANG X L, YIN S L. Intrusion detection model of random attention capsule network based on variable fusion. Journal on Communications, 2020, 41(11): 160 – 168 (in Chinese).

(Editor: Wang Xuying)

From p. 69

[32] USMAN S M, KHALID S, ASLAM M H. Epileptic seizures prediction using deep learning techniques. IEEE Access, 2020, 8: 39998 – 40007.

[33] YANG X W, ZHAO J Q, SUN Q, et al. An effective dual self attention residual network for seizure prediction. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2021, 29: 1604 – 1613.

[34] TURK Ö, ÖZERDEM M S. Epilepsy detection by using scalogram based convolutional neural network from EEG signals. Brain Sciences, 2019, 9(5): Article 115.

[35] RASHED-AL-MAHFUZ M, MONIM A, UDDIN S, et al. A deep convolutional neural network method to detect seizures and characteristic frequencies using epileptic electroencephalogram (EEG) data. IEEE Journal of Translational Engineering in Health and Medicine, 2021, 9: Article 2000112.

[36] SUKRITI, CHAKRABORTY M, MITRA D. A computationally efficient automated seizure detection method based on the novel idea of multiscale spectral features. Biomedical Signal Processing and Control, 2021, 70: Article 102990.

[37] ZHAO W, ZHAO W B, WANG W F, et al. A novel deep neural network for robust detection of seizures using EEG signals. Computational and Mathematical Methods in Medicine, 2020, DOI: 10.1155/2020/9689821.

(Editor: Wang Xuying)