Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

CRGT-SA: an interlaced and spatiotemporal deep learning model for network intrusion detection[*]

Jue CHEN, Wanxiao LIU, Xihe QIU[†‡], Wenjing LV, Yujie XIONG

School of Electronic and Electrical Engineering,
Shanghai University of Engineering Science, Shanghai 310027, China
[†]E-mail: qiuxihe@sues.edu.cn

Received May 30, 2024; Revision accepted Sept. 18, 2024; Crosschecked June 5, 2025

Abstract: To address the challenge of cyberattacks, intrusion detection systems (IDSs) are introduced to recognize intrusions and protect computer networks. Among all these IDSs, conventional machine learning methods rely on shallow learning and have unsatisfactory performance. Unlike machine learning methods, deep learning methods are the mainstream methods because of their capability to handle mass data without prior knowledge of specific domain expertise. Concerning deep learning, long short-term memory (LSTM) and temporal convolutional networks (TCNs) can be used to extract temporal features from different angles, while convolutional neural networks (CNNs) are valuable for learning spatial properties. Based on the above, this paper proposes a novel interlaced and spatiotemporal deep learning model called CRGT-SA, which combines CNN with gated TCN and recurrent neural network (RNN) modules to learn spatiotemporal properties, and imports the self-attention mechanism to select significant features. More specifically, our proposed model splits the feature extraction into multiple steps with a gradually increasing granularity, and executes each step with a combined CNN, LSTM, and gated TCN module. Our proposed CRGT-SA model is validated using the UNSW-NB15 dataset and is compared with other compelling techniques, including traditional machine learning and deep learning models as well as state-of-the-art deep learning models. According to the simulation results, our proposed model exhibits the highest accuracy and F1-score among all the compared methods. More specifically, our proposed model achieves 91.5% and 90.5% accuracy for binary and multi-class classifications respectively, and demonstrates its ability to protect the Internet from complicated cyberattacks. Moreover, we conduct another series of simulations on the NSL-KDD dataset; the simulation results of comparison with other models further prove the generalization ability of our proposed model.

Key words: Intrusion detection; Deep learning; Convolutional neural network; Long short-term memory; Temporal convolutional network
https://doi.org/10.1631/FITEE.2400459 **CLC number:** TP391.4

1 Introduction

The latest report from Juniper Research states that there will be 84 billion network connections in 2024 (Oseni et al., 2023). Meanwhile, cyberattacks are constantly evolving and pose a significant threat to a wide variety of cutting-edge technologies, such as smart hospitals, power, medical, and campuses (Lv et al., 2021). For example, attacks against critical infrastructures used for power generation can lead to loss of property or even personal safety (Cook et al., 2020).

Intrusion detection systems (IDSs) are responsible for identifying intrusions that evade security technique, and providing a vital second level of resistance to protect computer networks (Fang et al., 2021). Multiple recent IDS studies adopt

[‡] Corresponding author

[*] Project supported by the Young Scientists Fund of the National Natural Science Foundation of China (No. 62102241)

ORCID: Jue CHEN, https://orcid.org/0000-0001-7508-2635; Xihe QIU, https://orcid.org/0000-0003-4024-925X

machine learning and deep learning methods to solve security issues (Al-Garadi et al., 2020; Chen et al., 2022). Conventional machine learning methods include logistic regression (LR), Gaussian naive Bayes (GNB), K-nearest neighbors (KNN), adaptive boosting (AdaB), and random forest (RF), and almost all the methods mentioned above use shallow learning which relies on manual feature engineering to extract features (Vasilomanolakis et al., 2015). Because of the huge amount of data, shallow learning is unable to solve real-time problems (Laghrissi et al., 2021). As a result, these methods achieve unsatisfactory performance in identifying different types of cyberattacks (Wang K et al., 2023).

Unlike machine learning methods, deep learning methods have become the most dominant roles in the field of intrusion detection, because of their ability to deal with mass data without prior knowledge of specific domain expertise. For instance, convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and temporal convolutional networks (TCNs), as standard deep learning technologies, can deal with network intrusions with different degrees of difficulty, complexity, and distributivity (Wang XF et al., 2020). To achieve high accuracy in detecting and classifying different types of cyberattacks, this study proposes a new network intrusion detection approach with a hybrid deep learning model.

Among deep learning technologies, recurrent neural network (RNN) is a kind of dynamic and feed-forward neural network, and is capable of learning sequential data over timesteps (Aldweesh et al., 2020). As an enhanced version of RNN, LSTM can use a gating mechanism to learn long-term dependencies. Unlike RNNs, TCN processes inputs in parallel and extracts high-dimensional abstract features from raw data, making it a viable alternative (Fenghour et al., 2021). As a result, we combine LSTM with a gated TCN to make full use of time-series prediction of LSTM combined with the feature extraction and fusion of TCN, which can finally improve the performance of processing time-series data.

On the other hand, CNN comprises convolutional layers, pooling layers, and (optional) fully connected layers. Among these layers, convolutional layers contain filters to extract features. Pooling layer then selects features from the convolutional layers through sub-sampling (Aldweesh et al., 2020). In general, CNN fits multi-dimensional data well when extracting spatial features. Moreover, when the input of the neural network is multiple vectors of different sizes that are related to each other, the self-attention mechanism can be used to make the IDS notice relationships between different parts of the input, and then select more important features.

In summary, this paper proposes a novel and hybrid deep learning model called CRGT-SA to achieve network intrusion detection. The proposed model uses a CNN to learn spatial features, combines gated TCN with RNN to extract temporal characteristics, interlaces them to generate the integrated features, and imports self-attention to select more important features. The main contributions of this paper are summarized as follows:

1. A novel IDS model that combines CNN and LSTM with gated TCN is proposed to improve the capability of learning spatiotemporal characteristics from network traffic in a hierarchical manner. More specifically, our proposed model splits the feature extraction into multiple steps with a gradually increasing granularity, and executes each step through a combined CNN, LSTM, and gated TCN module.

2. A self-attention mechanism is imported for calculating the weight which reflects the importance of each feature, and makes the neural network focus on the most important features, which finally improves the IDS performance.

3. A series of simulations is conducted on the UNSW-NB15 dataset. The simulation results verify that our proposed model achieves superior performance on intrusion detection compared with other traditional machine learning and state-of-the-art deep learning models. More specifically, our proposed model achieves accuracy of 91.5% and 90.5% in the binary and multi-class classifications, respectively. To further prove the generalization ability of our proposed model, we conduct another series of simulations on the NSL-KDD dataset. The simulation results show that our proposed model still achieves the best performance among all the compared models, in both binary and multi-class classifications.

2 Related works

Artificial intelligence approaches have continuously been applied to develop reliable IDSs.

For example, Marteau (2021) proposed a semi-supervised ensemble approach based on random partitioning binary trees for intrusion detection. Moreover, detection when facing collective anomalies was improved through taking into account the relative frequency of visits to the leaves of the trees (Marteau, 2021). To enhance the performance of single learners, Abdelmoumin et al. (2022) built an ensemble learning model which is composed of principal component analysis (PCA), a support vector machine, and a neural network to detect attacks on the Internet of Things (IoT). Khan (2021) presented an IDS based on soft voting to select optimal supervised classifiers to maximize accuracy and minimize false alarm rates. Furthermore, Khan (2021) adopted different sampling methods to solve the data imbalance problem (Khan et al., 2023). Qi et al. (2022) merged local sensitive hash (LSH), isolated forest, and PCA together to efficiently detect attacks in Industry 4.0; these components operate on multi-aspect data, catch group anomalies, and reduce dimensionality for correlations between different attributes. However, these classical machine learning methods can only learn shallow features, which limits the learning ability and detection accuracy.

Unlike traditional machine learning methods, deep learning has received extensive attention in the domain of intrusion detection because of its powerful learning ability and independence from feature engineering, which can further improve the accuracy. For instance, Abeshu and Chilamkurti (2018) proposed a distributed deep learning-based attack detection architecture for fog computing in IoT, which exchanges parameters and models between worker and master nodes. In terms of the deep learning models, the stacked autoencoder and softmax regression were used for feature engineering and attack classification, respectively (Abeshu and Chilamkurti, 2018). Kumar et al. (2022) integrated blockchain with a deep learning technique to realize privacy-preserving intrusion detection in the Internet of Vehicles (IoV). More specifically, the blockchain and LSTM modules were used to transmit data securely and identify cyberattacks. Oseni et al. (2023) designed a CNN-based IDS for IoV and employed the SHapley Additive exPlanations (SHAP) mechanism to interpret how a feature value increases or decreases a model's prediction. Nie et al. (2022) constructed an intrusion detection model based on a generative adversarial network (GAN) to detect a single attack and multiple attacks for secure social IoT. Taking the data and concept drifts into consideration, Wahab (2022) presented drift detection and outlier detection methods to study the change in the variances of the features over time and identify the outliers that diverge both from historical and temporally close data points, respectively. To counter drifts, Wahab (2022) discussed an online deep neural network (DNN)-based model to adjust the size of hidden layers and to realize intrusion detection dynamically. Laghrissi et al. (2021) first used PCA and mutual information (MI) for dimensionality reduction and feature selection, respectively. On that basis, they implemented an LSTM-based model to realize intrusion detection. Because of the small number of labeled IoT traffic records, Abdel-Basset et al. (2021) introduced a hierarchical semi-supervised training model to realize intrusion detection. Moreover, they imported a multi-scale residual temporal convolutional model and an improved attention mechanism to fine-tune the network capability in learning spatiotemporal representations and estimate the importance score of different features, respectively. Vinayakumar et al. (2019) established a hybrid network IDS using a DNN with five hidden layers that can handle mass data in real time. To prove the universality of the DNN-based model, contrast experiments were conducted on multiple public datasets, including KDDCup 99, NSL-KDD, UNSW-NB15, Kyoto, WSN-DS, and CICIDS 2017.

On the basis of machine learning and deep learning methods, multiple hybrid learning models have been proposed for intrusion detection that combines multiple machine learning or deep learning algorithms to make full use of each one's advantage. Hassan et al. (2020) combined deep CNN with LSTM to create an effective network intrusion detection approach for big-data scenarios, in which CNN and weight-dropped LSTM were used for extracting meaningful features and retaining long-term dependencies, respectively. Khan (2021) built a convolutional RNN-based IDS to predict and classify malicious cyberattacks. In this system, CNN and RNN modules were responsible for capturing local features and temporal features, respectively (Khan, 2021). Cao et al. (2022) designed a network intrusion detection approach merging a CNN with gated recurrent units (GRUs) to solve the issue of low classification accuracy. After solving data imbalance and realizing feature selection, CNN and GRU modules were used to extract spatial features and long-distance dependent information features, respectively. Kasongo (2023) adopted multiple different RNN types, including LSTM, GRU, and RNN to compose a network intrusion detection framework and compared the performance of these RNN models. Moreover, an XGBoost-based algorithm was implemented for feature selection to reduce the feature space of datasets. Jian et al. (2019) presented a multi-path IDS composed of two models: coupled data embedding (CDE) and coupled outlier scoring of high-dimensional data (COSH) for clustering and outlier detection, respectively.

Even though existing works have combined CNN with RNN modules for intrusion detection, most of them simply structure these networks in tandem, leading to the loss of temporal or spatial information. Unlike the methods mentioned above, our proposed model divides the feature extraction into multiple steps with gradually increasing granularity, and performs each step using a combined CNN, gated TCN, and LSTM block, which can maintain the spatiotemporal characteristics of network traffic effectively. Moreover, this paper introduces the self-attention mechanism to select the most significant features.

3 Proposed model

3.1 Overview

As mentioned before, most existing related works that combine CNN with RNN modules for intrusion detection simply structure them in tandem. For instance, in the hierarchical spatialtemporal features-based IDS (HAST-IDS) architecture (Wang W et al., 2018) (as shown in Fig. 1), only after CNN block has learned low-level spatial features from network traffic, will it be handed over to the RNN module for subsequent processing. It can be inferred that a loss of temporal information may exist due to CNN module, which degrades the performance of the following RNN module and the whole system.

In consideration of the above issues, this paper proposes a novel and hybrid deep learning model called CRGT-SA, which integrates CNN, RNN, gated TCN, and self-attention into an entirety, as shown in Fig. 2. Instead of allowing CNN to achieve full learning, our proposed model intertwines CNN with gated TCN and RNN (LSTM) modules; that is, feature extraction is divided into multiple steps with gradually increasing granularity, and each step is executed through a combined CNN, gated TCN, and LSTM block. Moreover, self-attention mechanism is introduced to measure the importance of the input features, so the network will emphasize the most significant features.

As can be seen from the dashed line labeled "increase granularity," learning begins with coarse-grained learning, and hence the output of CNN module will still reserve the temporal information for the following gated TCN and LSTM modules. The learning granularity becomes more fine-grained as the learning continues. However, at each level, CNN, gated TCN, and LSTM learn spatiotemporal characteristics at the same granularity. Consequently, these

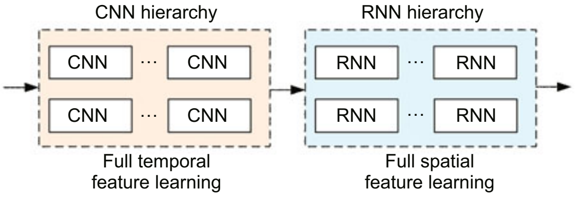

Fig. 1 Diagram of the HAST-IDS model

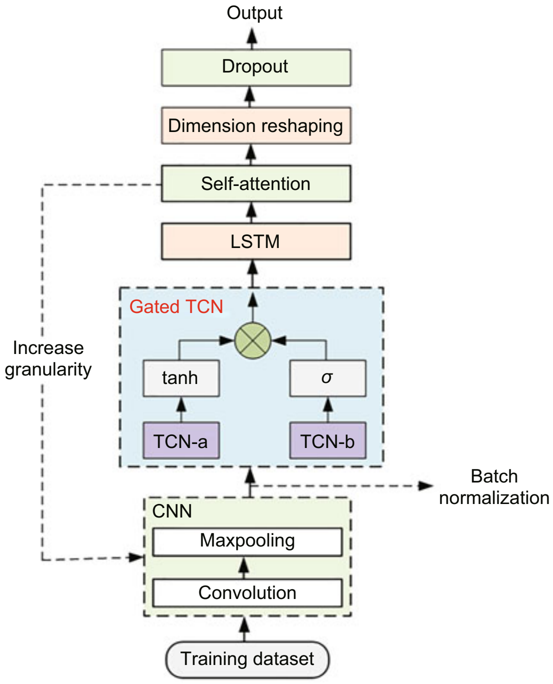

Fig. 2 Diagram of the CRGT-SA model

modules can learn independently to their full extent. Moreover, batch normalization, dimension reshaping, and dropout are useful for handling covariance shift, data reshaping, and over-fitting, respectively.

3.2 Batch normalization

When the DNN is adopted, the range of input value varies dynamically from layer to layer in the training stage, which makes learning efficiency be highly dependent and leads to unstable learning outcomes. To make the sample data more stable and accelerate the convergence of the DNN, batch normalization is adopted to modulate the output of CNN to satisfy the requirement of gated TCN in our proposed model, and the corresponding formula is shown as follows:

$$\hat{x} = \frac{x - \mu_\mathrm{B}}{\sqrt{\delta_\mathrm{B}^2 + \varepsilon}}, \tag{1}$$

where x is the single eigenvalue currently to be normalized in the input tensor, and μ_B and δ_B correspond to the batch mean and batch standard deviation, respectively. Note that ε is used to ensure a nonzero denominator, which can be ignored. Based on $\hat{x}$, the normalization generates the output $\hat{y}$ through the formula:

$$\hat{y} = \gamma\hat{x} + \beta, \tag{2}$$

where both γ and β are trained for a better learning effect.

3.3 Gated TCN

TCN can be used to solve a time-series prediction problem, which is composed of a 1D convolutional layer with the same input and output lengths. Fig. 3 shows a diagram of the dilated casual convolution with the kernel size to be 2. The inputs are selected every d steps and a standard 1D convolution is applied to the selected inputs. Given a 1D sequence of inputs $\boldsymbol{x} \in \mathbb{R}^T$ and filter $\boldsymbol{f} \in \mathbb{R}^K$, the representation of the dilated casual convolution operation of $\boldsymbol{x}$ with $\boldsymbol{f}$ at step t is shown in the following formula:

$$\boldsymbol{x} * \boldsymbol{f}\,(t) = \sum_{s=0}^{K-1} \boldsymbol{f}\,(s)\boldsymbol{x}\,(t - d \cdot s)\,, \tag{3}$$

where d is the dilation factor. The symbol “*” is a convolution operation; specifically, it is a 1D dilated causal convolution. As the name suggests, the dilated causal convolution is a combination of dilated convolution and causal convolution. In dilated convolution, interval sampling is allowed, whereas in causal convolution, data at time t of layer i only depend on the effect of time t of layer $i-1$ and the previous values, which is a strict time-constrained model that discards the influence of future data during training. Moreover, the residual network is adopted to transmit data across layers and ensure the consistency of the input and output.

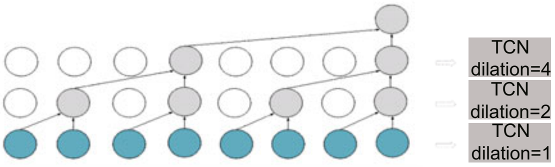

Fig. 3 Dilated causal convolution

On that basis, we import gated TCN in this paper to extract complicated temporal features, where only an output gate is involved in our proposed model. Given the input $\boldsymbol{X} \in \mathbb{R}^{N\times D\times S}$, it takes the following formula:

$$\boldsymbol{h} = g\left(\boldsymbol{\theta}_1 * \boldsymbol{X} + \boldsymbol{b}_1\right) \odot \sigma\left(\boldsymbol{\theta}_2 * \boldsymbol{X} + \boldsymbol{c}\right), \tag{4}$$

where $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, $\boldsymbol{b}_1$, and $\boldsymbol{c}$ are model parameters, $\odot$ is the element-wise product, $g(\cdot)$ is an activation function of the outputs, and $\sigma(\cdot)$ is the sigmoid function.

3.4 CNN

CNN can handle dense connections between layers of deep neural networks, and consists of convolutional layers, pooling layers, and optional fully connected layers. Convolutional layers directly receive multi-dimensional inputs, and convolve between the inputs to generate feature maps. Pooling layers perform sub-sampling for feature maps to reduce the dimensionality (Gan et al., 2022). Because a packet is stored in 1D format, we use the following formula to process the input vector $\boldsymbol{g}$ with filter $\boldsymbol{f}$ of size m:

$$\left(\boldsymbol{f} * \boldsymbol{g}\right)(i) = \sum_{j=1}^{m} \boldsymbol{g}\,(i)\,\boldsymbol{f}\,(i - j + m/2). \tag{5}$$

Moreover, rectified linear unit (ReLU) is chosen as the activation function in consideration of its high speed convergence.

$$\boldsymbol{f}(\boldsymbol{z}) = \max(\boldsymbol{0}, \boldsymbol{z}). \tag{6}$$

3.5 LSTM

LSTM belongs to gated RNN, which controls the feedback with multiple gate functions to reserve long-term dependencies instead of short-term ones. Fig. 4 shows a diagram of LSTM, which is composed of four connected subnetworks (represented as $\boldsymbol{p}$-net, $\boldsymbol{g}$-net, $\boldsymbol{f}$-net, and $\boldsymbol{q}$-net), multiple control gates, and a memory component.

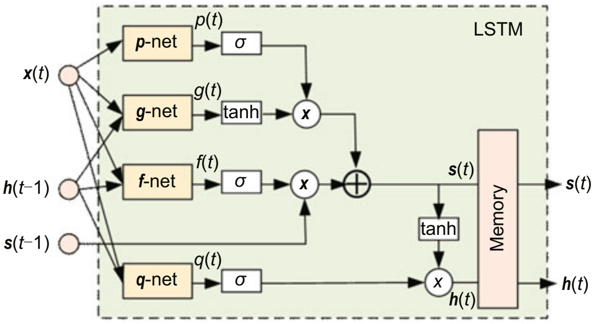

Fig. 4 Diagram of the LSTM

All the subnetworks have a unified structure, as shown in the following expression:

$$\boldsymbol{b}_2 + \boldsymbol{U}\boldsymbol{x}\left(t\right) + \boldsymbol{W}\boldsymbol{h}\left(t-1\right), \tag{7}$$

where $\boldsymbol{x}(t)$, $\boldsymbol{h}(t-1)$, and $\boldsymbol{b}_2$ are the current input, previous output, and bias, respectively. $\boldsymbol{U}$ and $\boldsymbol{W}$ are the weight matrix for the current input and recurrent weight matrix for the previous output, respectively. Note that the four subnetworks are different in $\boldsymbol{b}_2$, $\boldsymbol{U}$, and $\boldsymbol{W}$. The outputs of the four subnetworks are executed by two types of controlling gates, i.e., σ and tanh, to calculate the feedback $\boldsymbol{s}(t)$ from the previous learning and the current output $\boldsymbol{h}(t)$:

$$\boldsymbol{s}\left(t\right) = \sigma\left(\boldsymbol{f}\left(t\right)\right)*\boldsymbol{s}\left(t-1\right)+\sigma\left(\boldsymbol{p}\left(t\right)\right)*\tanh\boldsymbol{g}\left(t\right), \tag{8}$$

$$\boldsymbol{h}\left(t\right) = \tanh\boldsymbol{s}\left(t\right) * \sigma\left(\boldsymbol{q}\left(t\right)\right). \tag{9}$$

3.6 Self-attention

The function of self-attention mechanism is to enable the model to learn to assign weights for input signals on its own; i.e., different dimensions of the input signal are scored and features are weighted according to scores, which can emphasize the influence of significant features. As shown in Fig. 5, the self-attention mechanism can generate weights for different connections dynamically, which can be used as a layer in a neural network, as specified in the following formula:

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \text{soft max}\left(\frac{\boldsymbol{K}^{\mathrm{T}}\boldsymbol{Q}}{\sqrt{F_k}}\boldsymbol{V}\right), \tag{10}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ represent the matrices of query vectors, key vectors, and value vectors respectively, and $\sqrt{F_k}$ is the vector length with k being the dimension.

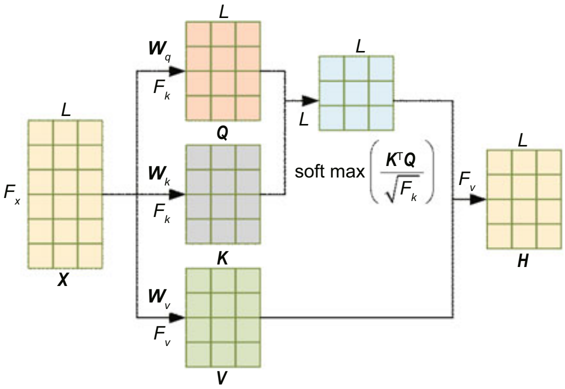

Fig. 5 Diagram of the self-attention mechanism. $\boldsymbol{F_x}$ maps input $\boldsymbol{X}$ to $\boldsymbol{Q}$; $\boldsymbol{F_k}$ maps input $\boldsymbol{X}$ to $\boldsymbol{K}$; $\boldsymbol{F_v}$ maps input $\boldsymbol{X}$ to $\boldsymbol{V}$. $\boldsymbol{H}$: output matrix; $\boldsymbol{L}$: sequence length; $\boldsymbol{W_q}$, $\boldsymbol{W_k}$, and $\boldsymbol{W_v}$: linear weights for $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$, respectively

3.7 Output

On one hand, due to the variation in learning granularity among different steps, there exists a possibility that the output size of one processing step (performed by a combined CNN, gated TCN, and LSTM block) may not match the required input size of the subsequent step. Therefore, the dimension reshaping layer is added to adjust the data for the following module. On the other hand, a typical problem of learning mass data with a DNN is overfitting, where the network learns too well from the training data, which limits the capability of recognizing variables from new data samples. In this situation, the dropout layer is added to randomly remove multiple connections from the DNN to reduce the possibility of overfitting problem.

4 Simulation setup and evaluations

4.1 Dataset

The evaluation of an intrusion detection model is strongly associated with the chosen dataset.

Multiple datasets for intrusion detection contain a large amount of superfluous data, making simulation results untrustworthy (Zhuo et al., 2017). To ensure the effectiveness of simulations in our study, we choose the UNSW-NB15 dataset (Moustafa and Slay, 2016) without any redundancy, which was produced by the Australian Center for Cyber Security (ACCS) in 2015. The data samples were first collected from the Internet, and then simulated in a lab to generate the dataset. There are nine UNSW-NB15 attack types: Denial of Service (DoS), Exploits, Generic, Shellcode, Reconnaissance, Backdoor, Worms, Analysis, and Fuzzers, and the corresponding proportions are 6.35%, 17.28%, 22.85%, 0.59%, 5.43%, 0.90%, 0.07%, 1.14%, and 9.41%, respectively. Normal traffic accounts for the remaining 35.98% of the dataset. Table 1 illustrates the specific description of each attack type.

4.2 Data preprocessing

Before inputting data to our proposed model, the raw data should be cleaned, labeled, and annotated first. More specifically, the data preprocessing stage contains three steps: conversion, standardization, and cross validation.

1. Conversion. To make the simulations more effective, data need to agree with the input format required by the neural network. The original network traffic data contain classification features in the form of text information, which cannot be directly processed by our proposed model. As a result, the text information needs to be transformed to numerical values, which can be implemented through the use of “get dummies” in Pandas.

2. Standardization. The means and standard derivations of the input data may differ, which can lead to inefficient learning. As a result, we zoom the input data to ensure that the means and standard derivations are 0 and 1, respectively.

3. Cross validation. UNSW-NB15 contains 2 540 044 samples, and we employ the stratified k-fold cross validation strategy to split all these samples into k groups, with $k-1$ groups and one group for training and validation, respectively.

4.3 Simulation setup

For the sake of proving the effectiveness of our proposed CRGT-SA model, we implement this model with PyTorch, Keras, and Scikit-Learn packages, and run the CPU @3.20 GHz and 16.0 GB RAM on the HP EliteDesk 800 G2 SFF desktop equipped with Intel® Core™ i5-6500. Table 2 displays a summary of our proposed model. Moreover, RMSprop is applied to optimize the weight and bias when training our proposed model, with the learning and dropout rates as 0.001 and 0.5, respectively. The pseudocode of the training procedure is shown in Algorithm 1.

Algorithm 1 Training procedure of CRGT-SA

```
Require: training dataset, learning rate, number of training
    epochs, batch size, and test dataset
Ensure: classification results of the test dataset
Preprocess training dataset, including the completion of
    missing values, the label coding of discrete features, and
    the matrix of reshaping the input vector;
for $i$ from 1 to the number of training epochs do
    for $j$ from 1 to $n$ do
        Start: $k = N/\mathrm{Batch}$;
        Split the training dataset into $k$ groups;
        Load the proposed model;
        Fit model with $k-1$ groups;
        Validate the model with the remaining $k^{\mathrm{th}}$ group;
    end for
end for
Test model on the test dataset;
```

Table 1 Description of attack types in the UNSW-NB15 dataset

Attack type	Description
Denial of Service (DoS)	Disguise as real hosts to access resources
Exploits	Control resources through vulnerabilities
Generic	Hash collision based on block cipher
Shellcode	Exploit vulnerabilities in software to execute code
Reconnaissance	Collect information illegally
Backdoor	Bypass security mechanisms to access data or control resources illegally
Worms	Spread through media
Analysis	Infiltrate web programs through scripts
Fuzzers	Input unexpected data and observe the output to find the vulnerabilities

Table 2 Summary of the CRGT-SA model

Layer	Name	Type	Number of parameters
1	CNN	CRGT-SA-Block	2.546×10^5
	Batch normalization		
	TCN		
	LSTM		
	Self-attention		
2	CNN	CRGT-Block	$1.3{\times}10^6$
	Batch normalization		
	TCN		
	LSTM		
	Self-attention		
3	CNN	CRGT-Block	$9.0{\times}10^6$
	Batch normalization		
	TCN		
	LSTM		
	Self-attention		
4	conv	Sequential	2.620×10^5
5	avg_pool	AvgPool1D	0
6	drop_out	Dropout	0
7	out	Sequential	6.50×10^4

The number of trainable parameters is $1.088\ 16 \times 10^7$, the number of total parameters is $1.088\ 16 \times 10^7$, and the number of the total estimated model parameters is $4.1249{\times}10^7$

4.4 Evaluation metrics

To measure the performance of the CRGT-SA model and compare it with those of other machine learning and deep learning models, we need to compute the accuracy, precision, recall, and F1-score for each model, which are explained and derived below.

1. Accuracy. This corresponds to the ratio of the number of correctly identified traffic records to the total number of traffic records.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{11}$$

2. Precision. This corresponds to the ratio of the number of correctly detected cyberattacks to the total number of traffic records identified as cyberattacks.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{12}$$

3. Recall. This corresponds to the ratio of the number of correctly detected cyberattacks to the total number of cyberattacks.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{13}$$

4. F1-score. This corresponds to the harmonic mean of the precision and recall.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{14}$$

In these formulas, TP is the number of cyberattacks correctly detected as intrusions, TN is the number of normal traffic records correctly classified, FP is the number of normal traffic records wrongly classified as cyberattacks, and FN is the number of cyberattacks wrongly classified as normal traffic.

4.5 Baseline methods

To demonstrate the validity of the CRGT-SA model, we implement a set of the most advanced machine learning and deep learning models for comparison. Machine learning models include LR, GNB, KNN, AdaB, and RF, which are introduced as follows in brief:

1. LR. This model estimates the probability of an event according to a given dataset with independent variables. Considering that the result is a probability, the dependent variables are in the range of 0 to 1.

2. GNB. This model assumes that the conditional probability of each feature dimension follows a Gaussian distribution, calculates the posterior probability for the new sample under a certain feature distribution according to the Bayes formula, and finally determines the category of the sample by maximizing the posterior probability.

3. KNN. To find the category for a new input instance with a given training dataset, the model firstly finds K instances closest to the target instance, and then classifies the target instance into the class, to which most of these K instances belong.

4. AdaB. This model is an iterative algorithm that trains different classifiers on a unified training dataset, and then combines these classifiers into a stronger classifier to generate the final results.

5. RF. This model is a supervised learning method that can summarize rules from a series of characterized and labeled data and represent these rules through the structure by a tree diagram. In the tree, each internal node, branch, and leaf node correspond to a judgment on an attribute, an output of a judgment result, and a classification result, respectively.

5 Simulation results and analysis

In this section, we describe our implementation of our proposed CRGT-SA model, and measure the corresponding loss trend and confusion matrix. Moreover, we conduct a comparative simulation and an ablation simulation against the recent standard and state-of-the-art methods to prove the effectiveness of our proposed model.

5.1 Loss trend

The loss curves over the number of steps for our proposed CRGT-SA model is depicted in Fig. 6, and the relationship between the numbers of steps and epochs is shown in Fig. 7. Fig. 6 shows the loss curves of binary and multi-class classifications. When the number of steps reaches 5×10^5, the loss tends to converge. As shown in Fig. 7, the number of steps is in linear relation to the number of epochs, and 5.5×10^5 steps corresponds to 100 epochs. Therefore, we set the number of epochs to be 100 in the following simulations. More specifically, in terms of the binary classification, the difference between the model's predicted value and the true value stabilizes at 0.1, while in terms of the multi-class classification, the difference stabilizes below 0.4. It can be explained because the latter is more difficult than the former; specifically, the sample of several cyberattacks is too small to detect accurately in the multi-class classification.

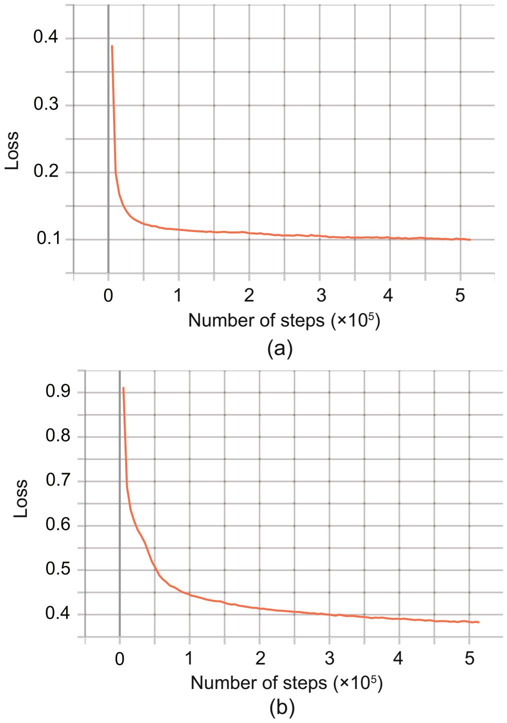

Fig. 6 Loss curves of binary classification (a) and multi-class classification (b)

5.2 Comparative analysis with traditional machine learning models

In this subsection, we compare the performance of our proposed CRGT-SA model against those of traditional machine learning models, and the comparative results of the binary and multi-class classifications are shown in Tables 3 and 4, respectively. As shown in Table 3, the accuracy of traditional machine learning models is between 71.6% and 87.7%, and the F1-score is between 79.2% and 91.2%. Among these models, LR and GNB have poor effects, the accuracy

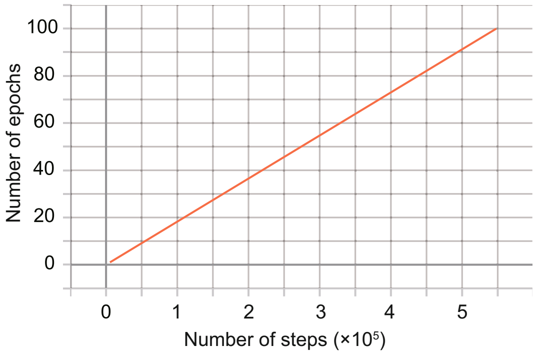

Fig. 7 Relationship between the numbers of epochs and steps

Table 3 Comparative analysis with traditional machine learning models of binary classification

Model	Accuracy (%)	F1-score (%)
LR	75.3	79.2
GNB	71.6	81.8
KNN	82.9	86.9
AdaB	83.9	88.4
RF	87.7	91.2
CRGT-SA	**91.5**	**91.6**

The best results are in bold

Table 4 Comparative analysis with traditional machine learning models of multi-class classification

Model	Accuracy (%)	F1-score (%)
LR	56.1	42.8
GNB	8.5	13.0
KNN	65.2	63.8
AdaB	63.1	55.7
RF	73.6	69.5
CRGT-SA	**90.5**	**74.3**

The best results are in bold

and F1-socre of KNN and AdaB are acceptable, and RF has the best performance. It can be inferred that the ensemble learning involved in RF makes it more tolerant to noise and outliers. In contrast, the accuracy and F1-score of the CRGT-SA model are 91.5% and 91.6%, respectively. In other words, the accuracy and F1-score of our proposed model are better by at least 4.3% and 0.4%, respectively, which indicates that this model achieves the best performance on binary classification among all the models.

As shown in Table 4, the accuracy of traditional machine learning models is between 8.5% and 73.6%, and the F1-score is between 13.0% and 69.5%. When compared to the binary classification, the accuracy and F1-score of multi-class classification obviously decrease. Among these models, GNB has the poorest performance, which may be attributable to the fact that the GNB assumes that features are independent of each other. When the number of features is large or features are in correlation with each other, GNB may not perform well. Because this study does not adopt the feature dimension reduction, and features may be relative to each other in the UNSW-NB15 dataset, then GNB obviously lags behind other models. In contrast, the accuracy and F1-score of the CRGT-SA model are 90.5% and 74.3%, respectively. In other words, the accuracy and F1-score of our proposed model are better by at least 23.0% and 6.9%, respectively, indicating that this model achieves the best performance on multi-class classification among all the models.

In addition, it can be observed that the difference between the accuracy and F1-score of our proposed CRGT-SA model in the multi-class classification is 16.2%, which cannot be ignored. The F1-score is a harmonic average of precision and recall, which indicates that this metric is more sensitive to class imbalance. In the chosen UNSW-NB15 dataset, as the Backdoor, Analysis, and Worms attacks make up only 0.90%, 1.14%, and 0.07% of all the traffic, it is difficult for models to detect these three types of cyberattacks. As a result, the poor performance on minority classes results in a reduction of F1-score, which should be improved in our future works.

5.3 Comparative analysis with state-of-the-art deep learning models

To further prove the superiority of our proposed CRGT-SA model, we compare it with state-of-the-art deep learning models, including HAST (Wang W et al., 2018), LuNet (Wu and Guo, 2019), and MSCNN-LSTM (Zhang et al., 2020), because all these models (including our proposed model) use CNN and RNN to learn spatial and temporal features, respectively. More specifically, both the HAST and MSCNN-LSTM combine CNN with RNN in tandem. By contrast, LuNet, which integrates CNN and RNN modules, serves as the baseline architecture for our proposed model. The simulation results are shown in Table 5, where it can be observed that the HAST model has the poorest performance, the performances of LuNet and MSCNN-LSTM are acceptable, and CRGT-SA has the best performance. The analysis is summarized in the following subsections.

Table 5 Comparative analysis with state-of-the-art deep learning models

Model	Accuracy (%)	F1-score (%)
HAST	81.4	53.0
LuNet	80.2	**76.8**
MSCNN-LSTM	89.8	66.8
CRGT-SA	**90.5**	74.3

The best results are in bold

5.3.1 Comparison of HAST with MSCNN-LSTM

Although both models stack all LSTM layers after CNN layers, they differ in the convolutional kernels of each layer. The HAST model always adopts the same scale, while MSCNN-LSTM uses the multi-scale convolutions effectively. As a result, MSCNN-LSTM is better than the HAST model in terms of accuracy and F1-score.

5.3.2 Comparison of HAST and MSCNN-LSTM with LuNet and CRGT-SA

The architectures of HAST and MSCNN-LSTM models may cause CNN layers to drop out the temporal information, leading to ineffective learning for LSTM layers. However, LuNet and CRGT-SA intertwine CNN with LSTM layers to capture both spatial and temporal features sufficiently. As a result, LuNet and CRGT-SA are better than HAST and MSCNN-LSTM due to dividing the feature extraction into multiple steps and the combined CNN+RNN block in each step.

5.3.3 Comparison of LuNet with CRGT-SA

On the basis of LuNet, our proposed CRGT-SA integrates CNN, RNN, gated TCN, and self-attention into a single model. Similarly, our proposed model can retain spatiotemporal properties at each step to ensure that all layers can fully exert their learning capacities. To better capture the temporal information, our proposed model combines gated TCN with LSTM modules to extract features from different angles. Moreover, our proposed model imports the self-attention mechanism to select significant features from network traffic data to help recognize cyberattacks. As a result, our proposed model improves the accuracy by 12.8% when compared with LuNet. However, our proposed model decreases the F1-score by 3.2% when compared with LuNet. Due to its gated TCN and self-attention mechanism, CRGT-SA exhibits a higher risk of overfitting compared to LuNet, particularly for minor classes in UNSW-NB15. This may contribute to its marginally lower F1-score on test data, though the difference is statistically acceptable.

5.4 Ablation studies

In this subsection, ablation simulations are executed to analyze the contributions of different components of our proposed CRGT-SA model for recognizing different types of cyberattacks. To investigate the function of the components involved in our proposed CRGT-SA model, which contains CNN, TCN, LSTM blocks, and a self-attention mechanism, we compare it with CNN, CNN-TCN, CNN-LSTM, and CNN-TCN-LSTM models. We evaluate these models and calculate the corresponding accuracies and F1-scores for binary and multi-class classifications as shown in Tables 6 and 7, respectively.

Among all the models, our proposed CRGT-SA model achieves the best performance on accuracy in both binary and multi-class classifications. This can be explained because our proposed model is composed of multiple interlaced modules, with each module learning one type of feature. In general, a more complicated model typically has stronger representation and can learn more complex patterns and functions, which is useful for capturing subtle dataset differences, leading to improvement in accuracy. In terms of the F1-score, our proposed model is 3.2% lower than the CNN-LSTM model of multi-class classification. The reason was explained in the previous subsection; i.e., minor classes of the dataset result in overfitting. However, the F1-score difference between the two models is still acceptable.

To explore the function of the self-attention mechanism, we compare the performance of our proposed CRGT-SA model with and without the self-attention module. The performance improvement, 10.0% in binary classification accuracy, 21.0% in multi-class accuracy, and 6.6% in multi-class F1-score, can be directly attributed to the inclusion of the self-attention mechanism, as this represents the key architectural difference between CNN-TCN-LSTM and our proposed CRGT-SA model. This explains the importance of the self-attention mechanism in selecting significant features.

Table 6 Ablation studies of binary classification

Model	Accuracy (%)	F1-score (%)
CNN	84.1	88.5
CNN-TCN	82.1	89.5
CNN-LSTM	84.8	90.3
CNN-TCN-LSTM	83.2	**91.6**
CRGT-SA	**91.5**	**91.6**

The best results are in bold

Table 7 Ablation studies of multi-class classification

Model	Accuracy (%)	F1-score (%)
CNN	77.2	73.2
CNN-TCN	76.1	69.4
CNN-LSTM	80.2	**76.8**
CNN-TCN-LSTM	74.8	69.7
CRGT-SA	**90.5**	74.3

The best results are in bold

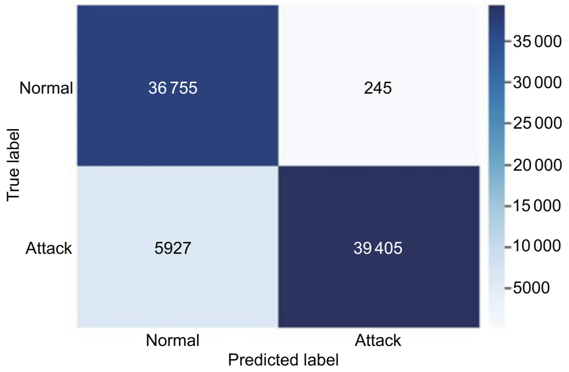

Fig. 8 Confusion matrix on binary classification

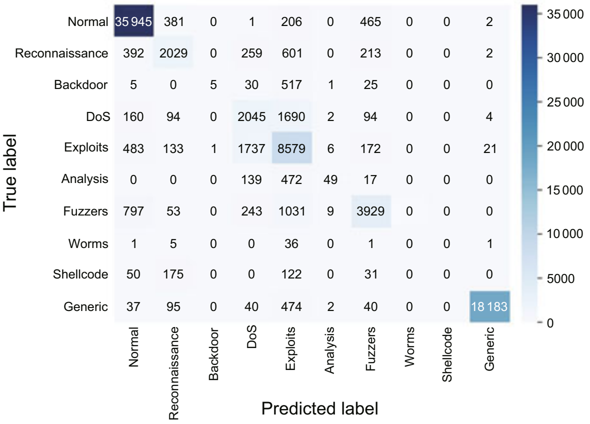

Fig. 9 Confusion matrix on multi-class classification

5.5 Confusion matrix

Confusion matrices of using our proposed CRGT-SA model on the UNSW-NB15 dataset for binary and multi-class classifications are shown in Figs. 8 and 9, respectively. As shown in Fig. 8, our proposed model can correctly detect most normal and abnormal traffic; specifically, 99.34% normal traffic is correctly classified. As shown in Fig. 9, our proposed model can correctly detect most normal traffic, as well as Reconnaissance, DoS, Exploits, Fuzzers, and Generic attacks. However, it is difficult for our proposed model to discover Backdoor, Analysis, Worms, and Shellcode attacks, and most of these attacks are classified as Exploits attacks. This can be explained because these five types of attacks mentioned above all rely on exploiting security vulnerabilities in the target system or software, and hence Backdoor, Analysis, Worms, and Shellcode attacks are similar to Exploits attacks in signature. Moreover, the Backdoor, Analysis, Worms, and Shellcode attacks make up only 0.90%, 1.14%, 0.07%, and 0.59% of all the traffic in the dataset, respectively, and provide insufficient data for the training stage. Therefore, the Backdoor, Analysis, Worms, and Shellcode attacks are wrongly recognized as Exploits attacks by the CRGT-SA model.

As one of our future directions, we plan to improve the accuracy of our proposed CRGT-SA model in detecting the four minor classes of cyberattacks, which will require us to master their differences more precisely at first. In terms of Backdoor and Exploits attacks, the former focuses on embedding backdoors in the system for long-term access and control, whereas the latter focuses on exploiting the vulnerability itself to perform malicious actions. In terms of Analysis and Exploits attacks, the former aims to improve the security of the system and protect the privacy of users, whereas the latter aims to perform malicious actions. In terms of Worms and Exploits attacks, the former type primarily infects the system by replicating and spreading itself and forming a worm network, whereas the latter primarily exploits specific vulnerabilities to perform malicious operations. In terms of Shellcode and Exploits attacks, the former can be considered as a subset or special case of the latter. On that basis, we plan to import the GAN module into our proposed model to generate realistic data and expand the diversity of samples for these minor classes.

5.6 Simulation results and analysis on the NSL-KDD dataset

To demonstrate the generalization ability of our proposed CRGT-SA model, we conduct simulations on the NSL-KDD dataset to compare the performance of our proposed model with those of traditional and state-of-the-art machine learning and deep learning models.

5.6.1 Comparative analysis with traditional machine learning models on the NSL-KDD dataset

Similar to Section 5.2, we conduct simulations to compare the performance of our proposed CRGT-SA model with those of traditional machine learning models including LR, GNB, KNN, AdaB, and RF on the NSL-KDD dataset, and the comparative results of binary and multi-class classifications are shown in Tables 8 and 9, respectively. It can be observed that our proposed model achieves the best performance among all the compared models in both binary and multi-class classifications. On the contrary, the GNB model has the poorest performance on both scenarios among all the models. This can be explained because the NSL-KDD dataset contains a large amount of network traffic data that are highly correlated and redundant. However, the GNB model assumes that all features are independent, which violates the characteristics of this dataset, leading to a low accuracy and F1-score. Moreover, the AdaB model performs poorly on multi-class classification, because it is a method based on weak classifier integration, which relies on the predictive performance of a weak classifier. In the multi-class classification scenario, weak classifiers have limited ability to distinguish complex patterns and multiple categories, which finally results in low accuracy and F1-score.

Table 8 Comparative analysis with traditional machine learning models of binary classification on the NSL-KDD dataset

Model	Accuracy (%)	F1-score (%)
LR	75.3	74.2
GNB	56.5	38.7
KNN	77.5	76.9
AdaB	77.6	77.3
RF	78.0	76.7
CRGT-SA	**90.7**	**91.8**

The best results are in bold

Table 9 Comparative analysis with traditional machine learning models of multi-class classification on the NSL-KDD dataset

Model	Accuracy (%)	F1-score (%)
LR	75.0	70.6
GNB	44.2	46.1
KNN	75.4	72.0
AdaB	46.8	40.5
RF	75.8	71.6
CRGT-SA	**86.7**	**86.6**

The best results are in bold

5.6.2 Comparative analysis with state-of-the-art deep learning models on the NSL-KDD dataset

Similar to Section 5.3, we conduct simulations to compare the performance of our proposed CRGT-SA model with those of state-of-the-art deep learning models including HAST, LuNet, and MSCNN-LSTM on the NSL-KDD dataset, and the comparative results are shown in Table 10. It can be observed that our proposed model achieves the best performance among all the compared models. When compared with the UNSW-NB15 dataset, our proposed model obtains higher accuracy and F1-score on the NSL-KDD dataset, because this dataset contains only four categories of cyberattacks and three of them belong to major classes and are easier to classify.

Table 10 Comparative analysis with state-of-the-art deep learning models on the NSL-KDD dataset

Model	Accuracy (%)	F1-score (%)
HAST	79.9	77.1
LuNet	81.3	77.4
MSCNN-LSTM	84.9	78.6
CRGT-SA	**86.7**	**86.6**

The best results are in bold

5.6.3 Ablation studies on the NSL-KDD dataset

Similar to Section 5.4, we conduct ablation simulations to compare the performance of our proposed CRGT-SA model with those of CNN, CNN-TCN, CNN-LSTM, and CNN-TCN-LSTM models on the NSL-KDD dataset, and the comparative results of the binary and multi-class classifications are shown in Tables 11 and 12, respectively. It can be observed that our proposed model achieves the best performance among all the compared models. Moreover, CNN-LSTM has a higher accuracy and F1-score when compared with CNN-TCN in multi-class classification. Even though both LSTM and TCN models are used for capturing temporal characteristics, they are different in learning long-term dependencies. The LSTM model designs a special gating mechanism to selectively remember or forget the past information, which is effective in capturing long-term dependencies. However, the TCN model relies on the extended convolution operations to learn temporal characteristics, because a simple convolution cannot effectively capture the long-term dependencies.

Table 11 Ablation studies of binary classification on the NSL-KDD dataset

Model	Accuracy (%)	F1-score (%)
CNN	81.3	81.6
CNN-TCN	84.3	74.3
CNN-LSTM	82.4	82.8
CNN-TCN-LSTM	84.0	84.7
CRGT-SA	**90.7**	**91.8**

The best results are in bold

Table 12 Ablation studies of multi-class classification on the NSL-KDD dataset

Model	Accuracy (%)	F1-score (%)
CNN	77.9	80.7
CNN-TCN	78.1	78.7
CNN-LSTM	83.6	85.1
CNN-TCN-LSTM	79.4	82.2
CRGT-SA	**86.7**	**86.6**

The best results are in bold

5.6.4 Confusion matrix on the NSL-KDD dataset

The confusion matrices of using our proposed CRGT-SA model on the NSL-KDD dataset for binary and multi-class classifications are shown in Figs. 10 and 11, respectively. It can be observed that our proposed model can correctly classify normal traffic as well as three abnormal traffic types including DoS, Probe, and R2L attacks, because all of them belong to the majority classes of the dataset. However, most U2R traffic is recognized as normal traffic by our proposed model. This is because in the NSL-KDD dataset, only 119 of 148 517 pieces of data are labeled as U2R attacks, a proportion of 0.08%. In terms of the category with fewer data, our proposed model cannot be trained enough, resulting in poor classification.

6 Conclusions and future work

This study designs a novel and hybrid deep learning model called CRGT-SA to achieve network intrusion detection. To extract sufficient spatiotemporal properties from network traffic data, our proposed model intertwines CNN with gated TCN and LSTM module. More specifically, we divide the feature extraction into multiple steps with gradually increasing granularity, and execute each step using a combined CNN, LSTM, and gated TCN module. On that basis, the self-attention mechanism is introduced to select significant features from network traffic data. In the simulations, we evaluate and compare our proposed CRGT-SA model with traditional machine learning models and state-of-the-art deep learning models. According to the simulation results, our proposed model achieves the highest accuracy and F1-score among all the models, achieving accuracy of 91.5% and 90.5% for binary and multi-class classifications, respectively. Moreover, the generalization ability of our proposed model is further demonstrated through another series of simulations on the NSL-KDD dataset in comparison with other models.

However, as can be seen from the structure of our proposed CRGT-SA model, the computational complexity cannot be ignored. To enhance the computational efficiency, we plan to import the process

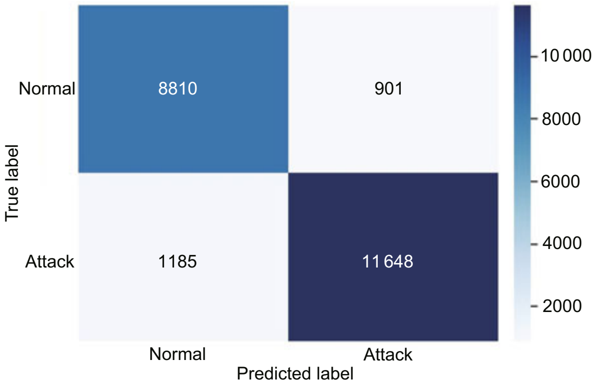

Fig. 10 Confusion matrix of binary classification on the NSL-KDD dataset

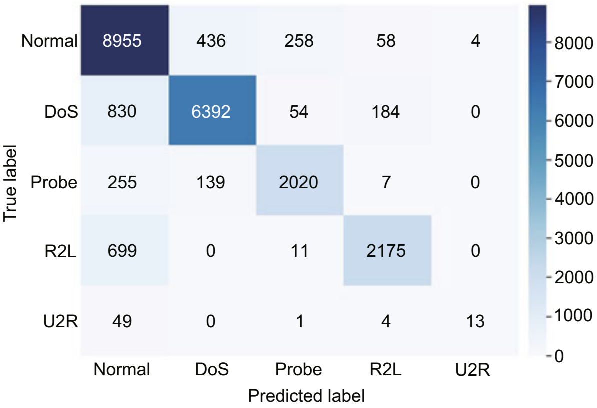

Fig. 11 Confusion matrix of multi-class classification on the NSL-KDD dataset

of feature selection into the data preprocessing stage in future work, such as adopting the genetic algorithm to reduce dimensionality. Moreover, as can be observed from the confusion matrix, our proposed CRGT-SA model cannot detect Backdoor, Analysis, Worms, or Shellcode attacks efficiently because of insufficient data in the training stage. To cope with this problem, we plan to import the GAN module into our proposed model to generate samples of rare attack categories to expand the dataset between the data preprocessing and intrusion detection stages.

Contributors

Jue CHEN and Wanxiao LIU designed the research. Wanxiao LIU processed the data. Jue CHEN and Wanxiao LIU drafted the paper. Xihe QIU, Wenjing LV, and Yujie Xiong helped organize the paper. Jue CHEN revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Abdel-Basset M, Hawash H, Chakrabortty RK, et al., 2021. Semi-supervised spatiotemporal deep learning for intrusions detection in IoT networks. *IEEE Int Things J*, 8(15):12251-12265. https://doi.org/10.1109/JIOT.2021.3060878

Abdelmoumin G, Rawat DB, Rahman A, 2022. On the performance of machine learning models for anomaly-based intelligent intrusion detection systems for the Internet of Things. *IEEE Int Things J*, 9(6):4280-4290. https://doi.org/10.1109/JIOT.2021.3103829

Abeshu A, Chilamkurti N, 2018. Deep learning: the frontier for distributed attack detection in fog-to-things computing. *IEEE Commun Mag*, 56(2):169-175. https://doi.org/10.1109/MCOM.2018.1700332

Aldweesh A, Derhab A, Emam AZ, 2020. Deep learning approaches for anomaly-based intrusion detection systems: a survey, taxonomy, and open issues. *Knowl Based Syst*, 189:105124. https://doi.org/10.1016/j.knosys.2019.105124

Al-Garadi MA, Mohamed A, Al-Ali AK, et al., 2020. A survey of machine and deep learning methods for Internet of Things (IoT) security. *IEEE Commun Surv Tutor*, 22(3):1646-1685. https://doi.org/10.1109/COMST.2020.2988293

Cao B, Li CH, Song YF, et al., 2022. Network intrusion detection model based on CNN and GRU. *Appl Sci*, 12(9):4184. https://doi.org/10.3390/app12094184

Chen J, Xiong YJ, Qiu XH, et al., 2022. A cross entropy based approach to minimum propagation latency for controller placement in software defined network. *Comput Commun*, 191:133-144. https://doi.org/10.1016/j.comcom.2022.04.030

Cook AA, Misirli G, Fan Z, 2020. Anomaly detection for IoT time-series data: a survey. *IEEE Int Things J*, 7(7):6481-6494. https://doi.org/10.1109/JIOT.2019.2958185

Fang LM, Li Y, Liu Z, et al., 2021. A practical model based on anomaly detection for protecting medical IoT control services against external attacks. *IEEE Trans Ind Inform*, 17(6):4260-4269. https://doi.org/10.1109/TII.2020.3011444

Fenghour S, Chen DQ, Guo K, et al., 2021. Deep learning-based automated lip-reading: a survey. *IEEE Access*, 9:121184-121205. https://doi.org/10.1109/ACCESS.2021.3107946

Gan BQ, Chen YQ, Dong QP, et al., 2022. A convolutional neural network intrusion detection method based on data imbalance. *J Supercomput*, 78(18):19401-19434. https://doi.org/10.1007/s11227-022-04633-x

Hassan MM, Gumaei A, Alsanad A, et al., 2020. A hybrid deep learning model for efficient intrusion detection in big data environment. *Inform Sci*, 513:386-396. https://doi.org/10.1016/j.ins.2019.10.069

Jian SL, Pang GS, Cao LB, et al., 2019. CURE: flexible categorical data representation by hierarchical coupling learning. *IEEE Trans Knowl Data Eng*, 31(5):853-866. https://doi.org/10.1109/TKDE.2018.2848902

Kasongo SM, 2023. A deep learning technique for intrusion detection system using a recurrent neural networks based framework. *Comput Commun*, 199:113-125. https://doi.org/10.1016/j.comcom.2022.12.010

Khan MA, 2021. HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion detection system. *Processes*, 9(5):834. https://doi.org/10.3390/pr9050834

Khan MA, Iqbal N, Imran N, et al., 2023. An optimized ensemble prediction model using AutoML based on soft voting classifier for network intrusion detection. *J Netw Comput Appl*, 212:103560. https://doi.org/10.1016/j.jnca.2022.103560

Kumar R, Kumar P, Tripathi R, et al., 2022. P2SF-IoV: a privacy-preservation-based secured framework for Internet of Vehicles. *IEEE Trans Intell Transp Syst*, 23(11):22571-22582. https://doi.org/10.1109/TITS.2021.3102581

Laghrissi F, Douzi S, Douzi K, et al., 2021. Intrusion detection systems using long short-term memory (LSTM). *J Big Data*, 8(1):65. https://doi.org/10.1186/s40537-021-00448-4

Lv ZH, Qiao L, Li JH, et al., 2021. Deep-learning-enabled security issues in the Internet of Things. *IEEE Int Things J*, 8(12):9531-9538. https://doi.org/10.1109/JIOT.2020.3007130

Marteau PF, 2021. Random partitioning forest for point-wise and collective anomaly detection—application to network intrusion detection. *IEEE Trans Inform Forens*

Secur, 16:2157-2172.
https://doi.org/10.1109/TIFS.2021.3050605

Moustafa N, Slay J, 2016. The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset. *Inform Secur J Glob Perspect*, 25(1-3):18-31.
https://doi.org/10.1080/19393555.2015.1125974

Nie LS, Wu YX, Wang XJ, et al., 2022. Intrusion detection for secure social Internet of Things based on collaborative edge computing: a generative adversarial network-based approach. *IEEE Trans Comput Soc Syst*, 9(1):134-145.
https://doi.org/10.1109/TCSS.2021.3063538

Oseni A, Moustafa N, Creech G, et al., 2023. An explainable deep learning framework for resilient intrusion detection in IoT-enabled transportation networks. *IEEE Trans Intell Transp Syst*, 24(1):1000-1014.
https://doi.org/10.1109/TITS.2022.3188671

Qi LY, Yang YH, Zhou XK, et al., 2022. Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure Industry 4.0. *IEEE Trans Ind Inform*, 18(9):6503-6511.
https://doi.org/10.1109/TII.2021.3139363

Vasilomanolakis E, Karuppayah S, Mühlhäuser M, et al., 2015. Taxonomy and survey of collaborative intrusion detection. *ACM Comput Surv*, 47(4):55.
https://doi.org/10.1145/2716260

Vinayakumar R, Alazab M, Soman KP, et al., 2019. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7:41525-41550.
https://doi.org/10.1109/ACCESS.2019.2895334

Wahab OA, 2022. Intrusion detection in the IoT under data and concept drifts: online deep learning approach. *IEEE Int Things J*, 9(20):19706-19716.
https://doi.org/10.1109/JIOT.2022.3167005

Wang K, Zhang AH, Sun HR, et al., 2023. Analysis of recent deep-learning-based intrusion detection methods for in-vehicle network. *IEEE Trans Intell Transp Syst*, 24(2):1843-1854.
https://doi.org/10.1109/TITS.2022.3222486

Wang W, Sheng YQ, Wang JL, et al., 2018. HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access*, 6:1792-1806.
https://doi.org/10.1109/ACCESS.2017.2780250

Wang XF, Han YW, Leung VCM, et al., 2020. Convergence of edge computing and deep learning: a comprehensive survey. *IEEE Commun Surv Tutor*, 22(2):869-904.
https://doi.org/10.1109/COMST.2020.2970550

Wu PL, Guo H, 2019. LuNet: a deep neural network for network intrusion detection. IEEE Symp Series on Computational Intelligence, p.617-624.
https://doi.org/10.1109/SSCI44817.2019.9003126

Zhang JW, Ling Y, Fu XB, et al., 2020. Model of the intrusion detection system based on the integration of spatial-temporal features. *Comput Secur*, 89:101681.
https://doi.org/10.1016/j.cose.2019.101681

Zhuo XY, Zhang JL, Son SW, 2017. Network intrusion detection using word embeddings. IEEE Int Conf on Big Data, p.4686-4695.
https://doi.org/10.1109/BigData.2017.8258516