Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

A Transformer-based network intrusion detection approach for cloud security

Zhenyue Long[1], Huiru Yan[2], Guiquan Shen[1], Xiaolu Zhang[1], Haoyang He[2] and Long Cheng[1,2*]

Abstract

The distributed architecture of cloud computing necessitates robust defense mechanisms to secure network-accessible resources against a diverse and dynamic threat landscape. A Network Intrusion Detection System (NIDS) is pivotal in this context, with its efficacy in cloud environments hinging on its adaptability to evolving threat vectors while mitigating false positives. In this paper, we present a novel NIDS algorithm, anchored in the Transformer model and finely tailored for cloud environments. Our algorithm melds the fundamental aspects of network intrusion detection with the sophisticated attention mechanism inherent to the Transformer model, facilitating a more insightful examination of the relationships between input features and diverse intrusion types, thereby bolstering detection accuracy. We provide a detailed design of our approach and have conducted a thorough comparative evaluation. Our experimental results demonstrate that the accuracy of our model is over 93%, which is comparable to that of the CNN-LSTM model, underscoring the effectiveness and viability of our Transformer-based intrusion detection algorithm in bolstering cloud security.

Keywords Cloud computing, Network intrusion detection, Transformer model, Attention mechanism, Network security

Introduction

In recent years, cloud computing has grown rapidly because it provides on-demand, simplified network access and a shared pool of configurable computing resources that can be quickly provisioned and released with little management or service provider interaction [1]. Based on these advantages, cloud computing is being deployed in more and more areas and has attracted an increasing number of applications to migrate to cloud environments [2, 3].

As cloud services are delivered over the Internet, the security and privacy of the cloud resources and services being deployed are also receiving increased attention [4, 5].

*Correspondence:
Long Cheng
lcheng@ncepu.edu.cn
[1] Joint Laboratory on Cyberspace Security, China Southern Power Grid, Guangzhou, China
[2] School of Control and Computer Engineering, North China Electric Power University, Beijing, China

At the network layer, cloud suffers from traditional attacks such as IP spoofing, Address Resolution Protocol (ARP) spoofing, Routing Information Protocol (RIP) attack, DNS poisoning, man-in-the-middle attack, port scanning, insider attack, Denial of Service (DoS), Distributed Denial of Service (DDoS), etc [6]. To address such issues, major cloud providers (such as Amazon EC2, Microsoft Azure, Open Nebula, etc.) use the firewalls. Firewalls protect the front-end access points of a system and are considered the first line of defense. However, since firewalls only sniff network packets at the network perimeter, they cannot detect intrusion attacks. In addition, some DoS or DDoS attacks are too sophisticated for traditional firewalls to detect. Therefore, using only a traditional firewall to block all intrusions is not an effective solution, extra protective measures are required.

Besides, integrating a Network Intrusion Detection System (NIDS) in the cloud is a prevalent approach to safeguard against attacks. NIDS serves as an alert mechanism, enhancing the security posture by identifying and

flagging network breaches that successfully infiltrate the system. In recent years, the use of machine learning and deep learning algorithms to construct detection models for NIDS has become widespread [7]. Despite the significant performance improvements compared to traditional techniques, most models rely on ample data of attack instances. In real-world scenarios, an organization's or enterprise's network system generates lower quality attack sample data for training. As a result, deep intrusion detection models have limited detection capabilities based on this data [8]. Thus, an efficient network intrusion detection method must be developed.

Recently, Transformer [9] and its variants, which utilize self-attention mechanisms, have achieved significant success in performing Natural Language Processing (NLP) tasks such as text classification, dialogue recognition, and machine translation. The key idea of Transformer is to pre-train on a large text corpus and then apply the trained model to a smaller task-specific dataset for fine tuning. Moreover, enlightened by Transformer's ability to handle ordered sequences of data, some researchers have used Transformer to detect intrusions and anomalies, proving its robustness in many scenarios [10].

Since network intrusion is usually a continuous behavior in time, most of the existing models do not have the ability to learn time series features and lose time series features. Although some methods based on Recurrent Neural Network (RNN) can learn time series features, their serial-based training methods have the problems of long training time and low convergence efficiency. The attention mechanism of the Transformer can effectively learn the temporal correlation of network intrusion data, thereby advancing the accuracy of network intrusion detection. Therefore, in this paper, we propose a network intrusion detection method based on the Transformer model, explore the feasibility of using the Transformer model for network intrusion detection on the CIC-IDS 2018 dataset, and verify its prediction effect.

In general, the main contributions of our work are summarized as follows:

- We unveil a novel intrusion detection methodology grounded in Transformer technology, tailored for cloud ecosystems, showcasing adeptness in analyzing intrusion behavior characteristics, and offering protection against a broad spectrum of attacks.
- We provide a thorough discussion on the process of network intrusion detection. Specifically, we initially delineate the architecture of the Transformer model, followed by an in-depth elucidation of the process entailed in our devised intrusion detection model, structured into pivotal stages: data preprocessing, model training, and label prediction.
- We conduct a thorough evaluation of our methodology using well-established datasets and performance metrics, elaborating on the experimental setup, environment, and dataset. The diverse experimental outcomes highlight the robustness and effectiveness of our algorithm.

The remainder of this paper is organized as follows. Firstly, we traverse through pertinent research that underpins our work. Subsequently, we unveil the architecture of our designed model, delineate the environment, and expound on the algorithm employed. Thereafter, we present the results of our experiments, followed by a conclusion summarizing our work.

Related work

Network security and intrusion detection

In this day and age, network security is highly imperative given the considerable increase in the utilization of computer networks. Consequently, sustaining network security has become increasingly challenging [11]. Specifically, an intrusion is the act of attempting to violate security policies or bypass computer and network security mechanisms [12]. It refers to any actions that violate a computer system's security policy and jeopardize the accuracy, confidentiality, and accessibility of a network resource. There are a variety of measures that can be taken to address the intrusion challenge in the network, such as firewalls and intrusion detection systems. For example, the work [13] develops a centralized NIDS and firewall within a Software Defined Network (SDN [14]) to enhance SDN security against multiple types of attacks.

The concept of intrusion detection was first proposed by James Anderson in 1980 [15], and with the development of network technology, intrusion detection systems began to develop continuously. Network intrusion detection refers to the analysis of network behavior, security logs, and other related data to detect whether the user has the purpose of breaking in or breaking out of the system [16]. It usually needs to analyze the characteristics of IP address for routing and forwarding, port for application server access, transport protocol held, packet arrival time and payload, and based on this, the detection model is established [17, 18]. The model monitors the packets entering and leaving the network. When a suspicious packet enters the network, the model blocks or permits the packet, and reports the abnormal situation to the responsible person. After receiving the report, the responsible person can take further measures.

NIDS as a common solution for solving network intrusion problem, has gained much attention for decades [19]. NIDS can generally be categorized into two types: signature-based NIDS and anomaly-based NIDS.

Signature-based NIDS identify threats by comparing network activities with known Indicators of Compromise (IOC) [20], and anomaly-based NIDS analyze all network operations by measuring them against a pre-established and standardized baseline that depicts the system's normal behavior [21]. To detect and prevent cyber attacks on networks, researchers have devoted significant effort to proposing a variety of NIDS. For instance, In work [22], a self-supervised Graph Neural Network (GNN) is implemented, which integrates and utilizes edge features for the detection of network intrusions and anomalies. And the work [23] proposes a novel intrusion detection systemapproach to deal with unbalanced and high-dimensional traffic.

Network intrusion detection in clouds

As mentioned previously, network intrusions are being studied extensively, with intrusions into cloud environments becoming a trending topic as well. This also drives the development of related algorithms, numerous network security researchers have formulated various detection algorithms and presented solutions for intrusion detection in networks. For example, the work [24] proposes an enhanced genetic algorithm to improve intrusion detection models based on support vector machines. In addition, a fitness function is devised based on classification accuracy, false alarm rate, and data feature dimensions. And the work [25] proposes a weighted naive Bayes intrusion detection model based on particle swarms, which combines rough set theory and an improved particle swarm algorithm to improve the detection capabilities of NIDS.

The works [26, 27] use the features available in pcap file format to obtain tracking analysis metadata, and provides a general solution for network anomaly detection through k-means clustering algorithm on the basis of parallel hardware. Then Ji Saihua et al. [28] improve the k-means algorithm and applied it to intrusion detection. Furthermore, there are many studies that have proposed intrusion detection algorithms in networks using deep learning. For example, the work [29] proposes nonsymmetric deep autoencoder for unsupervised feature learning and presents a deep learning classification model, addressing concerns about NIDS. And the work [30] uses Self-Taught Learning (STL), a deep learning based technique, on the NSL-KDD network intrusion dataset to propose an approach for developing such an efficient and flexible NIDS.

Moreover, the performance of related algorithms has been greatly improved. Convolutional Neural Network (CNN) is prominent in the fields of image processing and NLP, among which the LeNet-5 Network [31] model proposed by Yann LeCun et al. achieves a low false positive rate on the MNIST dataset. Based on this model, Wang Yong et al. [32] propose six different CNN structures for network intrusion detection to improve the overall classification accuracy. RNN has a good effect on the processing of text, speech, and sequence data, but with the increase of text or sequence length, the forgetting characteristics of RNN will become more and more obvious. Jihvun Kim propose the Long Short-Term Memory (LSTM) structure of short-duration memory network [33], which effectively alleviates the disadvantage of gradient disappearance in RNN. These have driven NIDS utilization algorithms toward more efficient and practical.

As a result, many well-established network intrusion detection methods have been proposed for cloud environments. For example, the work [34] develops a useful intrusion detection system for the cloud environment by utilizing ensemble feature selection and classification methodologies. It is employed for selecting feature sets with reduced and valuable data from the intrusion datasets. And the work [35] designs an effective security system that specifically targets cloud-based DoS/DDoS attacks utilizing a Multithreaded Network Intrusion Detection System (PM-NIDS) tailored to various protocols. As mentioned above, Transformer has many advantages and has been used in many areas in recent years, including network intrusion detection. For example, the work [36] proposes a new intrusion detection model utilizing n-gram frequency and time-aware Transformer. This model can hierarchically learn traffic features from both session and packet levels while minimizing information. Transformer-based network detection algorithms have also been proposed for cloud environments [37]. Similarly, In this paper, we propose a Transformer-based network intrusion detection approach in a cloud environment. In addition, the model has a certain portability. By adjusting and optimizing the parameters according to the new dataset or other types of network environment, the model can effectively adapt to the new dataset or network environment.

Description and design

In this section, we mainly introduce the basics and the detailed implementation of our algorithm and the flow of network intrusion detection based on Transformer.

Implementation of our algorithm

Attention mechanism

The Seq2Seq (sequence-to-sequence) model is a mapping of one sequence to another, often used in machine dialogue, machine translation, and so on. Seq2Seq belongs to encoder-decoder structure, which is composed of encoder and decoder. In the Seq2Seq model, two RNN structures act as the two modules respectively. The

encoder encodes the input into vector C and transmits it to the decoder. The model structure is shown in Fig. 1a. According to the structure, in an ideal case, the complete information of the input is saved in the last state h_n of the encoder part. However, if the input is very long, the information in h_n is incomplete, and then the information obtained by the decoder part will be incomplete, resulting in incomplete output content.

In order to solve the problem that Seq2Seq model uses incomplete information when the input is long, which leads to incomplete output, the attention mechanism is introduced. The structure is shown in the Fig. 1b. The Seq2Seq model whose decoder module no longer uses a single vector C as input, but has multiple codecs such as C_1, C_2, and C_3. When we predict y_1, maybe y_1 is focused on C_1, so C_1 is encoded semantically; when we predict y_2, y_2 is focused on C_2, so C_2 is encoded semantically, and so on, so we simulate the human attention.

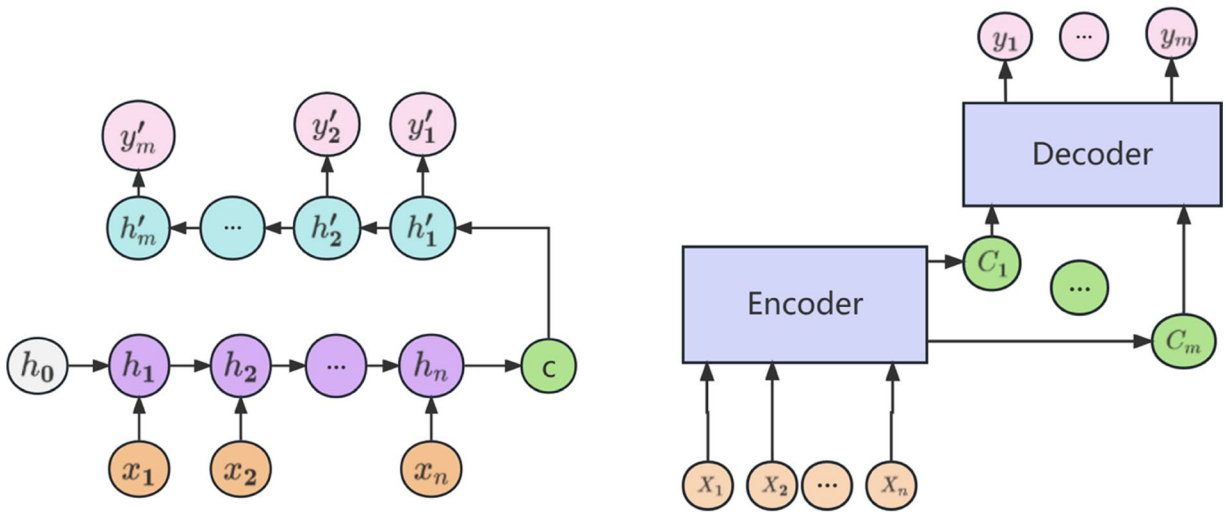
Transformer model structure

Transformer model adopts self-attention mechanism and completely abandons the structure of traditional RNN and CNN, which is widely used in NLP tasks and has achieved good results. Therefore, we propose a Transformer-based intrusion detection method to analyze the data characteristics of intrusion behaviors.

Consider that the encoder-decoder structure is primarily utilized to deal with variable-length sequences, such as translation problems, while network intrusion detection is fundamentally a classification problem. Therefore, this paper exclusively leverages the Transformer's encoder structure, discarding the decoder. The encoder output serves as the global feature, onto which a fully connected layer and a Softmax layer are added to produce the final output. The linear layer converts the weight matrix of the encoder output into the final required dimension, and the Softmax function normalizes the linear layer output into a value between (0,1), which is treated as a probability distribution and used as the target prediction value. The overall architecture of designed Transformer model is shown in Fig. 2.

The original dataset is preprocessed to obtain the feature vector X, which is used as the input of the encoder. In this paper, we stack the encoder with multiple layers, because with the increase of the number of layers, the capacity of the network is larger and the expressive power is stronger, which can make the model better handle long sequence data. The encoder consists of a multi-head attention layer, a feed-forward neural network layer, and a summation and normalization layer. The layers are described as follows.

Multi-head attention mechanism: The multi-head attention mechanism consists of multiple self-attention, which calculates the input matrix to obtain the output matrix Z_i, and the multi-head attention mechanism concatenates the Z_i to obtain the final output matrix Z. The essence of this layer is to weight the input information and the model allocates its attention according to the weights.



(a) Seq2Seq structure

(b) Introduce the Seq2Seq structure of Attention

Fig. 1 Attention mechanism

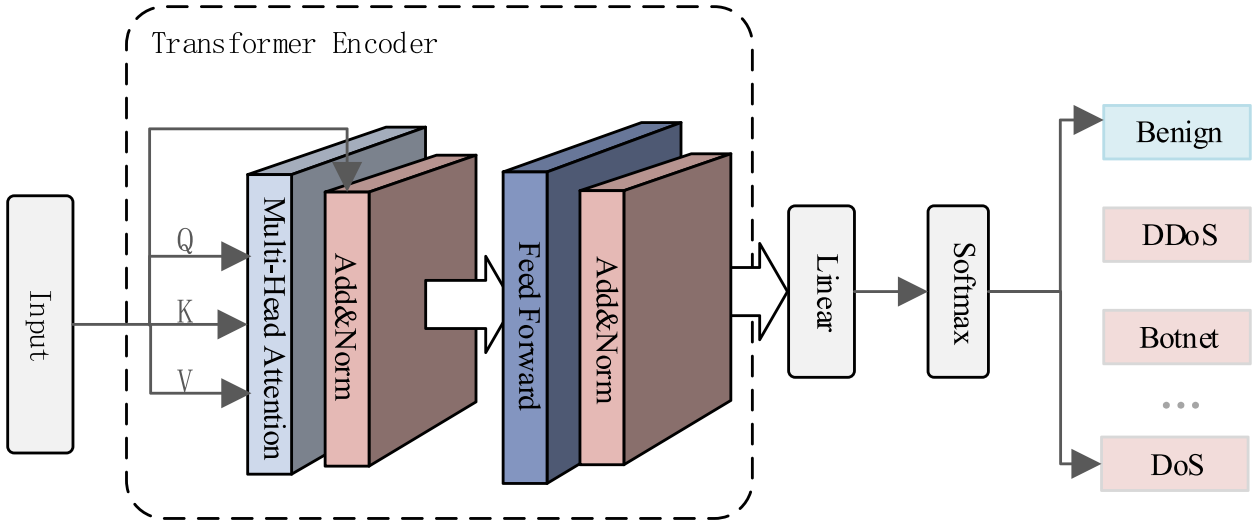


Fig. 2 The architecture of designed Transformer model

Feedforward neural network: The feedforward neural network is a two-layer fully connected layer, the first layer is activated using the ReLU function and the second layer is not activated using the activation function.

Summation and normalization layer: Two operations are performed in this layer: one is the residual join, which concatenates the input X of the previous layer with the output. The other is Layer Normalization, which normalizes the hidden layers to a standard normal distribution to speed up convergence.

NIDS based on Transformer

The overall architecture of network intrusion detection based on Transformer model is shown in Fig. 3, which is divided into three stages: data preprocessing, model training, and prediction and model evaluation.

Data preprocessing

As illustrated in Fig. 3, the preprocessing stage involves label coding, normalization, and batch processing of the dataset. Initially, the dataset is divided into a 7:3 ratio of training set and test set by label coding and normalization. Subsequently, the training set is utilized for model training while the test set is used for prediction and evaluation. This sequence of operations is described in further detail below.

Label coding: In the original dataset, the labels are in the form of strings that cannot be processed directly by the model. Therefore, it is necessary to convert them to numerical data for mathematical calculation and analysis. Label encoding involves encoding of the label of a data type into a one-hot vector. For example, botnets are coded as [0,1,0,0,0,0,0].

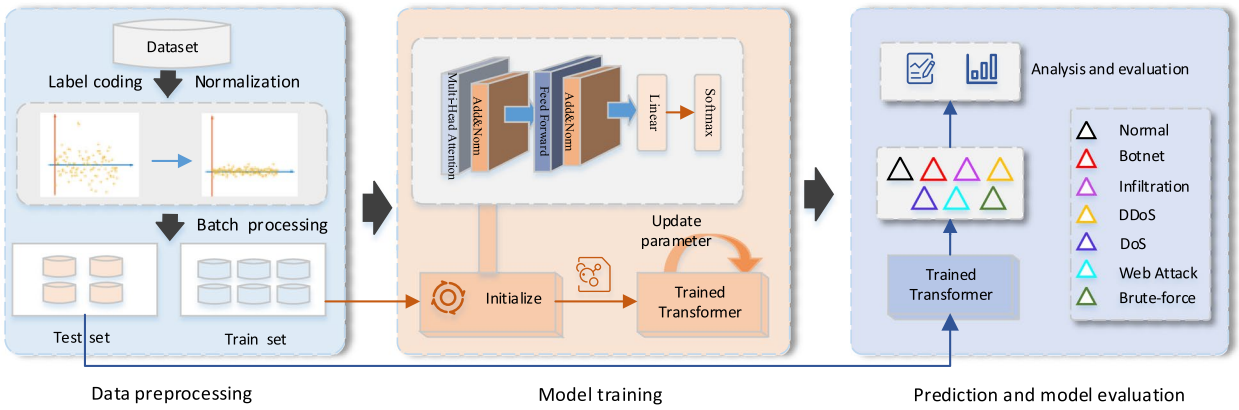


Fig. 3 The architecture of NIDS based on Transformer

Normalization: Due to the large difference of data in the dataset, which is not conducive to the training of the model, the dataset should be normalized first. In this paper, we use the MinMaxScaler function from sklearn, which normalizes each column of the data to maintain the shape of the original data, and the data range is (0,1) after processing. The equations are as follows:

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

$$x_{scaled} = x_{std} * (max - min) + min \tag{2}$$

Where x is the data in any column of dataset, x_{min} is the minimum value of the column data, x_{max} is the maximum value of the column data, x_{std} is the result after standardization, x_{scaled} is the data after the final normalization, max is the maximum value of the final mapping interval, min is the minimum value of the final mapping interval.

Batch data processing: Since there are tens of thousands of data in the dataset, in order to avoid the problems of insufficient memory and slow training speed, the data cannot be directly input into the model for training at one time, so the dataset is processed in batches first. In this paper, the DataLoader function in Pytorch is used for batch processing. The specific operation process is divided into three steps. Firstly, the TensorDataSet function was used to convert the processed training data. The converted data type was TensorDataSet, which could be identified by Pytorch. Secondly, input the dataset converted from the first step into the DataLoader function, and the function will generate an iterator to facilitate the training of subsequent batches of data. The batch size can be set through the function parameter batchsize. Thirdly, the iterator in the second step is used to obtain a small batch of data, and then the model is trained.

Model training

After the data is input to the model, the predicted value corresponding to the input data is output through forward propagation. Then the loss function is used to calculate the difference between the predicted value and the real value, that is, the loss value, and the model parameters are updated through reverse propagation to reduce the loss, so that the predicted value of the model is constantly close to the real value. After several iterations, the model with better performance is found by observing the decline rate of the loss value. Adam is used as the optimizer and CrossEntropyLoss function is used as the loss function. The two realize back propagation in the training of the whole model, and update the parameters of the model until the model converges.

The loss function is a measure of the difference between the predicted value of the model and the true value. The smaller the loss, the better the performance of the model. The CrossEntropyLoss function is used in this model, the function formula is shown in Eq. 3.

$$loss(y, lable) = -log\left(\frac{exp(y[lable])}{\sum_j exp(y[j])}\right) \tag{3}$$

Where y represents the predicted output of the model, and *lable* is the true label of the sample.

Experimental evaluation

In this section, we test the designed scheme and evaluate its feasibility. Firstly, the experimental setup is given, including the configuration of experimental environment and the description of dataset. Secondly, the data preprocessing process and evaluation index are described. Finally, the feasibility of the proposed method is verified by comparative analysis of different experimental results.

Experimental settings

Experimental environment and parameter configuration

In the experiment, the programming language we used is Python, and the Pytorch deep learning development framework is used to complete the design of the entire model. The parameter configuration of the experiment is shown in Table 1.

Dataset

The dataset we used is CIC-IDS 2018, which was developed by the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity Research for the purpose of intrusion detection research. The IDS 2018 dataset contains normal network traffic and attack data, totaling 12,212,461 items, and contains updated attack types compared to the previous dataset. Table 2 describes the concepts and quantities of each attack type. Each piece of data has 78 characteristic values in addition to a label value, including packet size, data stream length, duration, and data stream payload size.

Table 1 Parameter configuration of the model

Parameters	Value
Batch size	1024
Encoder layer	3,4,5
The number of neurons in the hidden layer of a fully connected network	512
Q,K,V dimensions	80
Heads of attention	8
Loss rate	0.1
Learning rate	0.001

Table 2 The concept and number of each attack type

Attack Type	Attack Description	Number of Attack
Benign	Normal.	10856019
Botnet	By infecting a large number of hosts with bot program viruses, a one-to-many control network is formed between the controller and the infected hosts.	144535
Infiltration	Exploit an application vulnerability to execute a backdoor on the victim's computer, using the victim's computer to scan the internal network and carry out an attack on other computers.	144336
DDoS Attack	Multiple distributed servers are used to send requests to the target, resulting in responses that affect correct and legitimate requests.	775955
DoS Attack	Attackers overload the system by carrying out a large number of attacks in a short period of time, making legitimate requests unresponsive.	196631
Web Attack	Web programs scan websites for attacks on vulnerable sites, such as SQL injection.	94101
Brute-force Attack	A common form of attack that uses programs to crack passwords by brute force, often to gain unauthorized access.	884

Model evaluation

Detection performance

We have adopted the normally used indicators: *Accuracy*, *precision*, *recall*, and $F1-score$ for the evaluation of the proposed system, which are defined as Eq. 4, 5, 6 and 7, respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FP} \tag{6}$$

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{7}$$

Where TP represents the amount of data that is actually normal and is predicted to be normal; TN indicates the amount of data that is actually abnormal and predicted to be abnormal. FP represents the amount of data that is actually abnormal but predicted to be normal; FN represents the amount of data that is actually normal but predicted to be abnormal.

Experimental results

In this part, we first conduct an experimental exploration of the Transformer-based network intrusion detection algorithm. After normalization and other preprocessing operations, the dataset is divided into test set and training set according to the ratio of 7:3, and sent to the model for training. Then, we train the model in different scenarios (the number of encoder layers is 3, 4, or 5) and evaluate the prediction effect. Finally, we compare the model with the CNN-LSTM model to evaluate the model performance.

In this experiment, the entire training set was used for training. When the number of encoder layers is 3, the number of training rounds is gradually increased, and then the performance of the model is tested using the test set. The detection results for each type of attack are shown in Table 3. It can be seen that the intrusion detection algorithm based on Transformer model has satisfactory detection effect on Botnet, DoS, Brute-force, and DDoS. Its predictive index value *Accuracy*, *Precision*, *Recall*, and $F1-score$ can reach 94% or even 99%.

Next, we gradually increase the number of encoder layers of the model for training. When the number of encoder layers is 3, 4, or 5, the comparison of the training accuracy trends and the comparison of the detection precision trends are shown in Figs. 4 and 5, respectively. And the comparison of the overall detection performance is shown in Fig. 6.

As can be seen from Fig. 4, with the increase of encoder layers, the fluctuation of model training accuracy becomes smaller and smaller, and the corresponding prediction performance is also improved. Specifically, the peak accuracy difference is 8.93% when there are three encoder layers, it is 6.13% when there are four encoder

Table 3 Detection of each type of attack

	Accuracy	Precision	Recall	F1-score
Normal	93.572	94.6515	98.3251	96.4533
Botnet	99.9891	99.7199	99.3589	99.5391
Infilteration	98.7943	44.2619	11.776	13.228
DDoS	94.7668	64.4014	39.4349	48.9167
DoS	99.8151	99.2901	89.1539	93.9494
Web	99.8675	85.5708	99.5962	92.0523
Brute-force	99.9612	17.2378	36.9811	12.1062

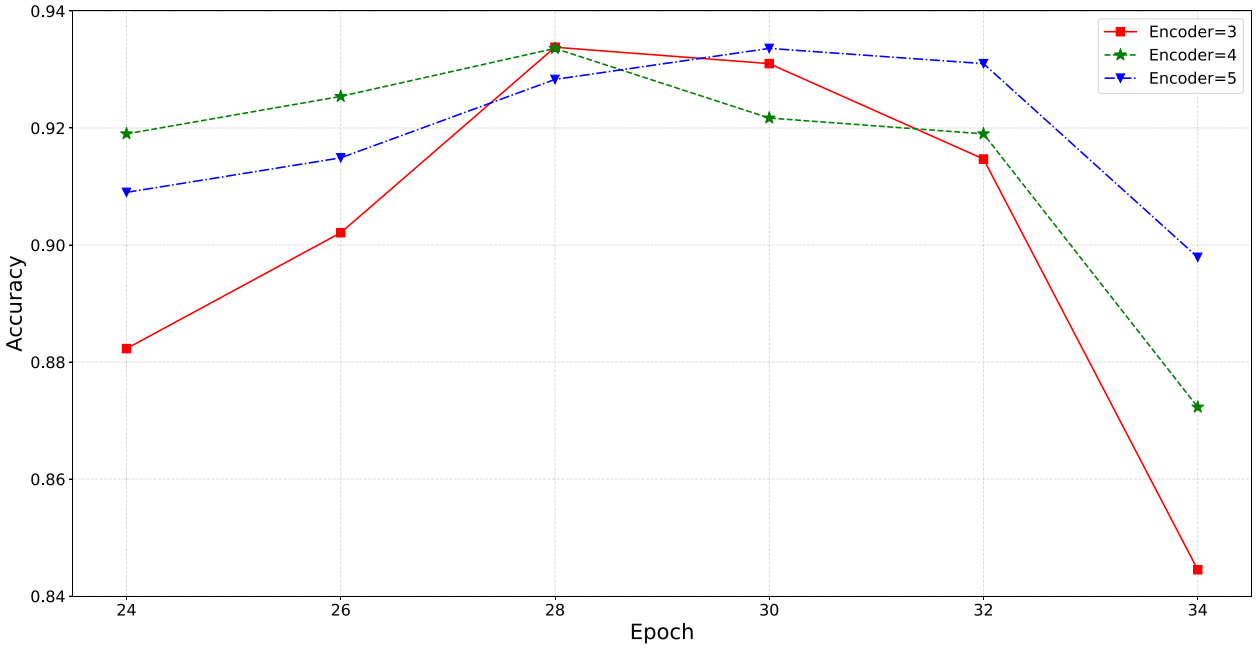


Fig. 4 Comparison of training accuracy trends when Encoder=3,4,5

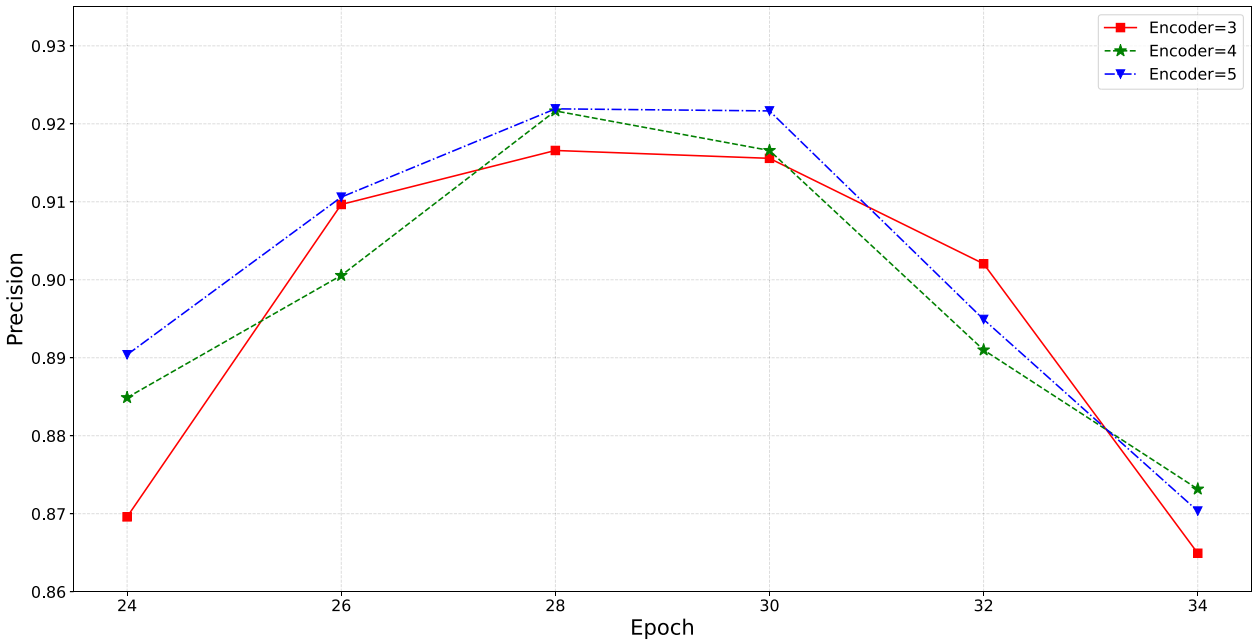


Fig. 5 Comparison of detection precision trends when Encoder=3,4,5

layers, and it is 3.57% when there are five encoder layers. At the same time, it can be seen from Fig. 5 that the detection precision curve for the encoder with five layers is slightly higher than the other two curves.

Figure 6 describes the numerical results of the four evaluation indicators when the number of encoder layers is 3, 4, or 5. When the number of encoder layers is 3, the values of four evaluation indexes were 93.38%, 91.7%, 93.38%, and 92.39%. When the number of encoder layers is 4, they are 93.36%, 92.16%, 93.36%, and 92.10%. And when the number of encoder layers is 5, they are 93.46%, 92.19%, 93.4%, and 92.16%. By comparison, with the

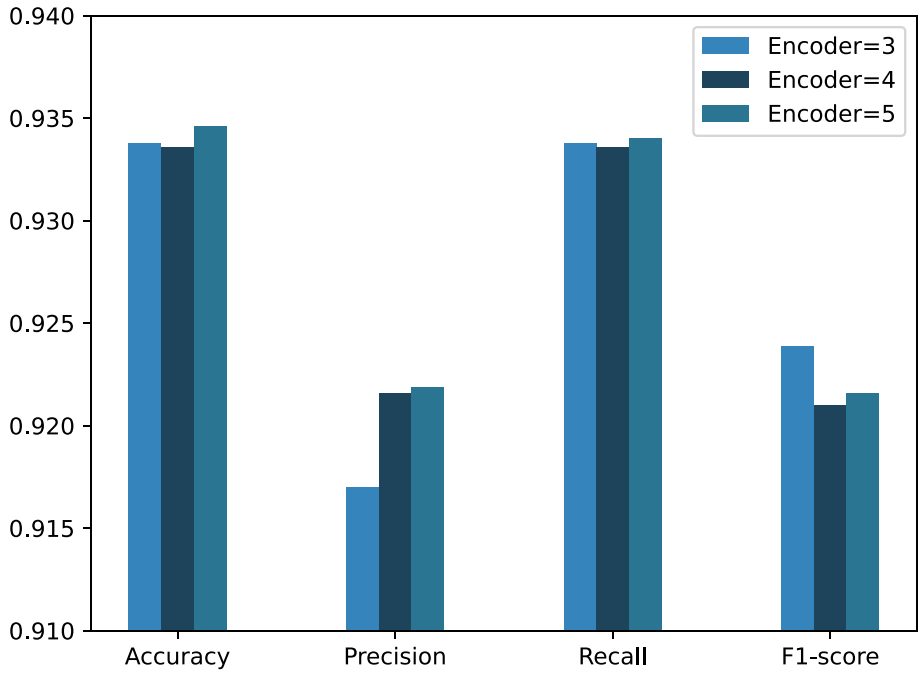


Fig. 6 Comparison of detection performance indicators when Encoder=3,4,5

increase of the number of encoder layers, the prediction accuracy increased by 0.46% and 0.03%.

In order to better evaluate the detection effect of the model, we compare the model with the CNN-LSTM model, and the experimental comparison results are shown in Table 4. Through comparison, it can be seen that under the current experimental conditions, our model has reached the accuracy of CNN-LSTM, and has some practical value initially.

Conclusion

In this paper, we proposed a Transformer-based network intrusion detection algorithm tailored for cloud environments, and assessed its viability and predictive accuracy. By harnessing the attention mechanism of the Transformer model along with network intrusion detection principles, we designed and implemented this algorithm. Our procedure entailed preprocessing the dataset, initially training the model with three encoder layers, and evaluating its predictive performance. Subsequently, we incrementally increased the encoder layers for further training and benchmarked our model against the CNN-LSTM model. The conclusive results reveal that under the specified experimental conditions, our Transformer-based network intrusion detection model attained a prediction accuracy surpassing 93%, on par with the latest method on CNN-LSTM model, showcasing its efficacy in predicting network intrusions within cloud environments.

Table 4 Comparison of intrusion detection capability with CNN-LSTM model

	Indicators	Benign	Botnet	Infilteration	DDoS	Web Attacks	Brute-force	DoS
Our	Accuracy	0.9357	0.9998	0.9879	0.9477	0.9988	0.9997	0.9982
	Precision	0.9465	0.9971	0.4427	0.6441	0.8557	0.1724	0.9929
	Recall	0.9833	0.9936	0.1178	0.3944	0.996	0.3698	0.8916
	F1-score	0.9645	0.9954	0.1323	0.4892	0.9205	0.1211	0.9395
CNN-LSTM	Accuracy	0.9457	0.9997	0.9476	0.9985	0.9988	0.9992	0.9993
	Precision	0.9074	0.9952	0.6373	0.9953	0.0165	0.9837	0.9912
	Recall	0.9816	0.9986	0.1747	0.9986	0.7143	0.9944	0.9996
	F1-score	0.9452	0.9992	0.2722	0.9975	0.0323	0.989	0.9953

Our future work mainly lies in the incorporation of Graph Neural Networks [38] into our Transformer-based model to enhance its ability to identify complex intrusion patterns in distributed environments such as cloud datacenters [39]. Moreover, we also plan to extend the application of our algorithm to more challenging environments, such as edge cloud systems [40]. These environments, with their decentralized nature, pose unique challenges that our model must adapt to. Generally, our long-term goal is to develop a system which can significantly enhance network intrusion detection across diverse cloud environments.

Acknowledgements
Not applicable.

Authors' contributions
Zhenyue Long: Conceptualization, Writing - review & editing. Huiru Yan: Methodology, Implementation, Writing - original draft. Guiquan Shen: Writing- review & editing. Xiaolu Zhang: Writing - review & editing. Haoyang He: Methodology, Writing - review & editing. Long Cheng: Conceptualization, Methodology, Writing - review & editing.

Funding
This work was supported by the Open Project Program of the Joint Laboratory on Cyberspace Security, China Southern Power Grid (No. CSS2022KF01).

Availability of data and materials
The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 17 October 2023 Accepted: 12 December 2023
Published online: 02 January 2024

References

1. Cheng L, Wang Y, Cheng F, Liu C, Zhao Z, Wang Y (2023) A deep reinforcement learning-based preemptive approach for cost-aware cloud job scheduling. IEEE Trans Sustain Comput
2. Zhang J, Cheng L, Liu C, Zhao Z, Mao Y (2023) Cost-aware scheduling systems for real-time workflows in cloud: An approach based on genetic algorithm and deep reinforcement learning. Expert Syst Appl 234:120972
3. Liu F, Huang J, Wang X (2023) Joint task offloading and resource allocation for device-edge-cloud collaboration with subtask dependencies. IEEE Trans Cloud Comput 11(3):3027–3039
4. Sun P (2020) Security and privacy protection in cloud computing: Discussions and challenges. J Netw Comput Appl 160:102642
5. Zhang X, Cui L, Shen W, Zeng J, Du L, He H, Cheng L (2023) File processing security detection in multi-cloud environments: a process mining approach. J Cloud Comput 12(1):100
6. Jangjou M, Sohrabi MK (2022) A comprehensive survey on security challenges in different network layers in cloud computing. Arch Comput Methods Eng 29(6):3587–3608
7. Li J, Tong X, Liu J, Cheng L (2023) An efficient federated learning system for network intrusion detection. IEEE Syst J 17(2):2455–2464
8. 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
9. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30:1–11
10. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al (2020) An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:201011929
11. Garg A, Maheshwari P (2016) A hybrid intrusion detection system: A review. In: 2016 10th International Conference on Intelligent Systems and Control (ISCO). IEEE, Coimbatore pp 1–5
12. Scarfone K, Mell P et al (2007) Guide to intrusion detection and prevention systems (idps). NIST Spec Publ 800(2007):94
13. Mantur B, Desai A, Nagegowda K (2015) Centralized control signature-based firewall and statistical-based network intrusion detection system (nids) in software defined networks (sdn). In: Emerging Research in Computing, Information, Communication and Applications: ERCICA 2015, vol 1. Springer, Bangalore pp 497–506
14. Liu Q, Cheng L, Alves R, Ozcelebi T, Kuipers F, Xu G, Lukkien J, Chen S (2021) Cluster-based flow control in hybrid software-defined wireless sensor networks. Comput Netw 187:107788
15. Liao HJ, Lin CHR, Lin YC, Tung KY (2013) Intrusion detection system: A comprehensive review. J Netw Comput Appl 36(1):16–24
16. Northcutt S, Novak J (2002) Network Intrusion Detection: An Analyst's Handbook, 3rd edn. New Riders Publishing, USA
17. García-Teodoro P, Díaz-Verdejo J, Maciá-Fernández G, Vázquez E (2009) Anomaly-based network intrusion detection: Techniques, systems and challenges. Comput Secur 28(1):18–28
18. Wang K, Stolfo SJ (2004) Anomalous payload-based network intrusion detection. Recent Advances in Intrusion Detection (RAID 2004). Sophia Antipolis, France, pp 203–222
19. Ahmad Z, Shahid Khan A, Wai Shiang C, Abdullah J, Ahmad F (2021) Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Trans Emerg Telecommun Technol 32(1):e4150
20. Erlacher F, Dressler F (2020) On high-speed flow-based intrusion detection using snort-compatible signatures. IEEE Trans Dependable Secure Comput 19(1):495–506
21. Alamiedy TA, Anbar M, Alqattan ZN, Alzubi QM (2020) Anomaly-based intrusion detection system using multi-objective grey wolf optimisation algorithm. J Ambient Intell Humanized Comput 11:3735–3756
22. Caville E, Lo WW, Layeghy S, Portmann M (2022) Anomal-e: A self-supervised network intrusion detection system based on graph neural networks. Knowl-Based Syst 258:110030
23. Mhawi DN, Aldallal A, Hassan S (2022) Advanced feature-selection-based hybrid ensemble learning algorithms for network intrusion detection systems. Symmetry 14(7):1461
24. Teng L, Teng S, Tang F, Zhu H, Zhang W, Liu D, Liang L (2014) A collaborative and adaptive intrusion detection based on svms and decision trees. In: 2014 IEEE International Conference on Data Mining Workshop. IEEE, Shenzhen pp 898–905
25. Ren X, Jiao W, Zhou D (2016) Intrusion detection model of weighted navie bayes based on particle swarm optimization algorithm. Comput Eng Appl 52(7):122–126
26. Velea R, Ciobanip C, Margarit L, Bica I (2017) Network traffic anomaly detection using shallow packet inspection and parallel k-means data clustering. Stud Inform Control 26(4):387–396
27. Ji S, Huang S (2021) Intrusion detection algorithm based on improved k-means. Comput Digit Eng 49(11):2184–2188
28. Wu, Fei, Ting Li, Zhen Wu, ShuLin Wu, and ChuanQi Xiao (2021) Research on network intrusion detection technology based on machine learning. Int J Wireless Inf Netw 28(no. 3):262–275

29. Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell 2(1):41–50
30. Javaid A, Niyaz Q, Sun W, Alam M (2016) A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). New York City, pp 21–26
31. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
32. Zhou H, Wang Y, Lei X, Liu Y (2017) A method of improved cnn traffic classification. In: 2017 13th International Conference on Computational Intelligence and Security (CIS). Hong Kong, pp 177–181
33. Kim J, Kim J, Thi Thu HL, Kim H (2016) Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon). Jeju, pp 1–5
34. Krishnaveni S, Sivamohan S, Sridhar S, Prabakaran S (2021) Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing. Clust Comput 24(3):1761–1779
35. Patil R, Dudeja H, Gawade S, Modi C (2018) Protocol specific multi-threaded network intrusion detection system (pm-nids) for dos/ddos attack detection in cloud. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, Bengaluru pp 1–7
36. Han X, Cui S, Liu S, Zhang C, Jiang B, Lu Z (2023) Network intrusion detection based on n-gram frequency and time-aware transformer. Comput Secur 128:103171
37. Ingle D, Ingle D (2023) An enhanced blockchain based security and attack detection using transformer in iot-cloud network. J Adv Res Appl Sci Eng Technol 31(2):142–156
38. Wu L, Cui P, Pei J, Zhao L, Guo X (2022) Graph neural networks: foundation, frontiers and applications. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Washington DC, pp 4840–4841
39. Cheng L, Wang Y, Liu Q, Epema DH, Liu C, Mao Y, Murphy J (2021) Network-aware locality scheduling for distributed data operators in data centers. IEEE Trans Parallel Distrib Syst 32(6):1494–1510
40. Chen Y, Zhao J, Hu J, Wan S, Huang J (2023) Distributed task offloading and resource purchasing in noma-enabled mobile edge computing: Hierarchical game theoretical approaches. ACM Trans Embed Comput Syst

Publisher's Note