Exploring Self-Attention Mechanism of Deep Learning in Cloud Intrusion Detection

Chenmao Lu[1], Hong-Ning Dai[1], Junhao Zhou[1], and Hao Wang[2]

[1] Macau University of Science and Technology, Macau SAR
cmlu.sec@gmail.com; hndai@ieee.org; junhao_zhou@qq.com
[2] Norwegian University of Science and Technology, Gjøvik, Norway
hawa@ntnu.no

Abstract. Cloud computing offers elastic and ubiquitous computing services, thereby receiving extensive attention recently. However, cloud servers have also become the targets of malicious attacks or hackers due to the centralization of data storage and computing facilities. Most intrusion attacks to cloud servers are often originated from inner or external networks. Intrusion detection is a prerequisite to designing anti-intrusion countermeasures of cloud systems. In this paper, we explore deep learning algorithms to design intrusion detection methods. In particular, we present a deep learning-based method with the integration of conventional neural networks, self-attention mechanism, and Long short-term memory (LSTM), namely CNN-A-LSTM to detect intrusion. CNN-A-LSTM leverages the merits of CNN in processing local correlation data and extracting features, the time feature extracting capability of LSTM, and the self-attention mechanism to better exact features. We conduct extensive experiments on the KDDcup99 dataset to evaluate the performance of our CNN-A-LSTM model. Compared with other machine learning and deep learning models, our CNN-A-LSTM has superior performance.

Keywords: Deep Learning · Convolution Neural Network · Self-Attention · Long Short-Term Memory · Network Intrusion Detection.

1 Introduction

Cloud computing can greatly complement to computing insufficiency of mobile devices or personal computers and the Internet of Things (IoT) nodes. Moreover, the recent advances in artificial intelligence, such as deep learning also put forth more stringent requirements on the computing capability of end devices. Moreover, the massive volumes of various data also drive the high storage capacity of end devices. However, either mobile devices and IoT nodes cannot cater to the rising demands on computing and storage capacity due to the built-in limitations while cloud computing facilities can fulfill the stringent computing requirements and provide users with elastic and ubiquitous computing services.

However, cloud computing is also faced with more and more security concerns [14, 15, 27]. For example, malicious cloud users (or tenants) may install

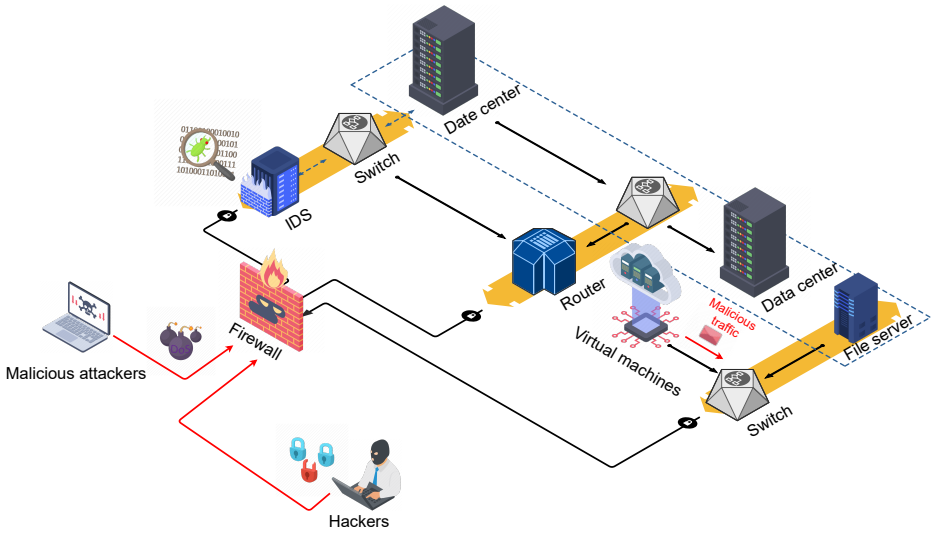


Fig. 1. An overview of cloud intrusion detection

malicious software (or malware) to a virtual machine (VM), which may infect other VMs. Meanwhile, malicious VMs may launch malicious attacks, such as distributed denial of services (DDoS) attack to affect other VMs or even paralyze the entire cloud. It is worth mentioning that those malicious attacks mainly happen inside the cloud network (i.e., internal network traffic) or outside the cloud network (i.e., external network traffic) as shown in Fig. 1.

There are a diversity of solutions to malicious attacks in cloud computing systems. Among them, intrusion detection for cloud network is the most crucial method since it is often the prerequisite for other countermeasures. The idea of intrusion detection for cloud networks is to analyze cloud network traffic and identify abnormal traffic. Thus, intrusion detection is essentially equivalent to a classification problem in machine learning. As a result, many recent studies attempt to apply machine learning methods to solve this classification problem.

Although conventional machine learning classifiers such as naive Bayes, logistic regression, decision tree, random forest have been adopted in intrusion detection systems (IDS), most of them suffer from poor performance in terms of classification accuracy. The root cause of the poor performance of these machine learning methods can mainly owe to the fact that they are incapable of properly processing and analyzing network intrusion detection data, which has the characteristics of high dimension and feature redundancy. Thus, sophisticated feature engineering is often required to process the network intrusion detection data. Different from conventional machine learning classifiers, the recent advances in deep learning methods can handle high-dimensional and redundant data without extract efforts on feature engineering.

Although deep learning methods have been used in intrusion detection and demonstrate the performance improvement over conventional machine learning methods [3,8,22,29], most of them adopt a singular structure, which may be beneficial to several dimensions of intrusion data while is struggling to handle other dimensions properly. Therefore, we present a composite deep learning model in this paper for cloud intrusion detection.

The key contributions of this paper are listed as follows.

- We propose a novel framework combining conventional neural networks, self-attention mechanism, and Long short-term memory, namely CNN-A-LSTM, for network intrusion detection.
- CNN-A-LSTM can well conduct network intrusion detection task. In particular, the CNN structure can process network intrusion data with spatial correlation while the self-attention mechanism can improve the learned parameters. The LSTM module can extract key time features from intrusion data. In this way, the characteristics of the network intrusion dataset can be better extracted.
- We evaluate our proposed CNN-A-LSTM model by conducting extensive experiments. Experimental results compare our model with other state-of-the-art methods and show that our CNN-A-LSTM achieves superior performance than other methods.

Following the introduction, Section 2 reviews related work. We then conduct a problem analysis in Section 3. Section 4 next briefs the main framework of CNN-A-LSTM and Section 5 presents the implementation details of CNN-A-LSTM. Section 6 gives the experiments results. Finally, Section 7 summarizes the paper.

2 Related Work

This section presents a literature survey on cloud security and challenges as well as traditional methods and deep learning models for detecting related security events.

2.1 Cloud security and challenges

Although cloud computing can provide users with elastic and ubiquitous computing servers, the centralization of cloud architecture also results in security vulnerabilities. Many research scholars have done relevant studies. Khalil et al. [7] conducted a comprehensive study of cloud security and privacy issues, categorizing known security threats and attacks, identifying 28 cloud security threats, and dividing them into five categories. Meanwhile, Singh et al. [19] introduced cloud computing challenges in eight categories. The main security challenges in cloud computing mainly include illegal access, data security, etc [19, 21, 23, 24]. In particular, the authors make a comparative analysis of existing data security

and privacy protection technologies in cloud computing in [23]. From another dimension, the work of [20] provides an overview of security issues that affect cloud computing and some security issues related to public and private cloud management. As in [5], the authors attempt to address human and technology-related threats and find security solutions.

2.2 Network Intrusion Detection

There is no denying that quickly identifying the security vulnerabilities of the network is critical to secure cloud computing. Network intrusion detection system (NIDS) has great potential to help detect threats and address security concerns. NIDS can detect abnormal behaviors that compromise the security of computer systems. We roughly divide the recent research on NIDS into two types: conventional methods and deep learning methods.

Conventional Methods Behl et al. [2] delved into several ways to secure cloud infrastructure and also compare their weaknesses. A scalable architecture for the deployment of IDS in the cloud for new application scenarios is proposed in [16]. Some traditional detection methods have been adopted in cloud computing, such as rule-based [1,11,12] for monitoring network traffic. They [10,12,18] have the limitations as follows: i) An unknown attack in the dataset cannot be detected. ii) The model needs to be regularly training to maintain a high detection rate. iii) Getting tag data is extremely difficult. iv) False positives are still high.

Deep Learning Methods Encouraged by the advent of machine learning and deep learning methods in computer vision and natural language processing fields, some related deep learning methods [3,8,13,17,22,28,29,31] have been applied to intrusion detection. The work [29] proposed the CNN-based NIDS model to extract intrusion identification information through supervised learning. An RNN IDS based on computational efficiency is proposed in [3]. Kim et al. [8] presented a host IDS based on exceptions, which adopts the LSTM model and system call language modeling method. Furthermore, Wang et al. [28] combine CNN and LSTM to learn the spatial and temporal respectively among multiple network packets.

Despite the progress of deep learning methods in intrusion detection, most of them adopt a singular structure, which cannot properly handle the high-dimensional data.

3 Problem Analysis

In related research experiments, the KDDcup99 network intrusion dataset [25] is one of widely used training sets. In previous studies such as [13], the KDDcup99 dataset was adopted to test the semi-supervised machine learning model using a trapezoidal network. We also choose the KDDcup99 dataset to conduct

Category R (1~9)					Category G (10~22)				Category Y (23~31)			Category B (32~41)		label
duration	…	service	…	urgent	hot	…	num_shells	…	…	serror_rate	…	…	dhsc	…	label
0		ecr_i		0	0		0			0.00			147		attack
1		smtp		0	0		0			0.00			246		attack
75		telnet		0	2		2			0.00			101		attack
198		telnet		0	3		2			0.00			1		attack
137		login		2	0		0			0.00			1		attack
98		telnet		1	1		0			0.00			4		attack
804		telnet		3	7		0			0.00			4		attack
36438		private		0	0		0			0.00			1		attack
8		pop_3		0	0		0			0.00			59		attack
0		smtp		0	0		0			0.99			77		normal
2399		telnet		3	0		0			0.00			1		normal
38		telnet		0	0		0			0.00			20		normal

Fig. 2. The example records of KDDcup99 dataset

the intrusion detection analysis. The KDDcup99 dataset contains 41 attributes, including both discrete and continuous attributes. As shown in Fig. 2, these attributes can be divided into four categories: i) network connection basic characteristics in Category R, ii) network connection content characteristics in Category G, iii) network traffic time-based statistical characteristics in Category Y, iv) network traffic host-based statistical characteristic in Category B. In these categories, we also show the representative attributes (e.g., 'duration', 'service' and 'hot'). Moreover, we add a new attribute (namely 'label') to represent the state of each network connection, marking as 'normal' or 'attack'.

3.1 Preliminary Analysis

We conduct a preliminary analysis by classifying and analyzing all the attributes of KDDcup99. Since the dataset contains 41 attribute features, it has more complex spatial features.

- Category R contains attribute 1 to 9 in KDDcup99 dataset, representing the basic characteristics of a transmission control protocol (TCP) connection. In particular, the 'duration' attribute corresponds to continuous data and the 'service' attribute corresponds to discrete data.
- Next, the following 13 attributes in Category G represent TCP connection content characteristics. For example, 'hot' attribute represents the times of access system-sensitive files and directories, and 'num_shells' attribute represents the number of shell terminals opened. In addition, all the data in Category G is continuous data.
- Category Y contains 9 attributes, i.e., 23 to 31, which are mainly applied to depict time-based network traffic statistical characteristics. In this category,

the data of every attribute is continuous data (e.g., 'serror_rate': the percentage of connections with synchronized sequence numbers (SYN) errors in the past two seconds).
- The remaining 10 attributes fall into Category B, being mainly used to describe host-based network traffic statistical characteristics. For example, 'dst_host_srv_count (dhsc)' is the number of connections with the same target host and service as the current connection in the previous 100 connections.

Since the dataset consists of the continuous data and discrete data with heterogeneous types of attributes and the attributes contain the complex temporal and spatial features, it is difficult to properly select and fully extract the attribute features via the traditional methods.

3.2 Challenges

Preliminary analysis in Section 3.1 implies the following characteristics of network intrusion detection datasets.

1. *The dataset contains heterogeneous types of attributes, resulting in the complex spatial relationship between them.* Traditional methods are difficult to extract the spatial content characteristics and to reflect the intrusion behavior effectively.
2. *There are a lot of attributes based on time characteristics in the dataset.* However, the statistical time of the network intrusion is too short (only 2 seconds). Therefore, it is difficult to learn the features and get an effective relationship between these attributes.
3. *The dataset has a strong temporal correlation.* In general, it is difficult to identify whether the state of a connection is 'attack' or 'normal', by only making statistics of connections between the current and the previous connection record in a period of time.

Therefore, the above characteristics pose challenges in conducting network intrusion detection with existing methods. To this end, we propose a CNN-A-LSTM model to address the above challenges. In particular, we adopt our approach to solve the problems of complex spatial and time relationships, obtaining the interdependent features of content over long distances.

4 Overview of Architecture

We propose CNN-A-LSTM model to detect network intrusion in this paper. The proposed model is composed of the following components, and its overview architecture is shown in Fig. 3.

1. **Data Preprocessing.** At this stage, we conduct preliminary analysis and preprocessing on network intrusion traffic in the KDDcup99 dataset.

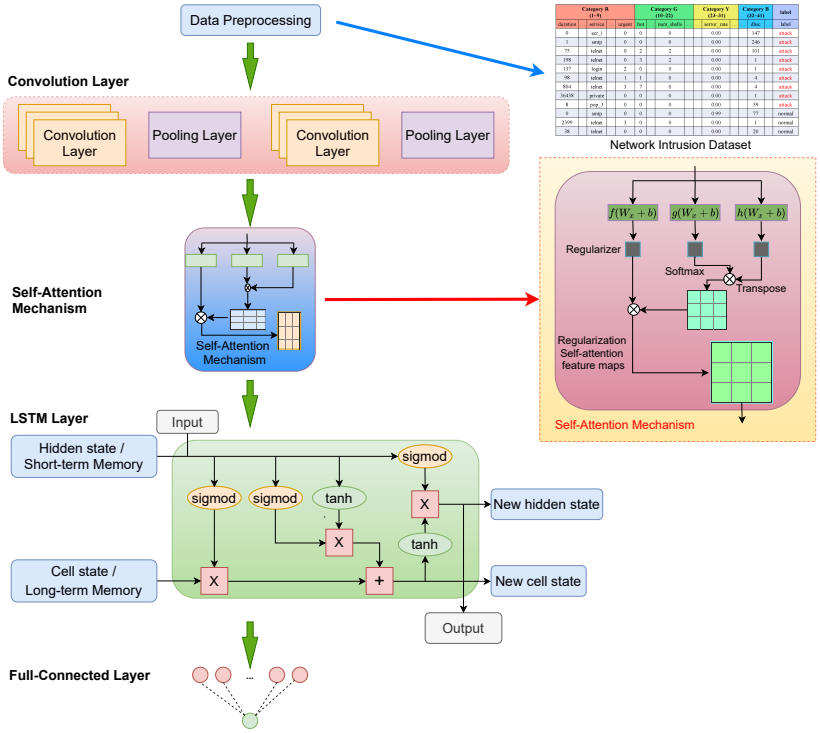


Fig. 3. An architecture of our CNN-A-LSTM model

2. **Convolution Layer.** The Convolution layer can effectively extract features from the dataset through a neural network. Motivated by [28], we exploit the CNN structure since the components of CNN can extract the spatial dependent characteristics of network intrusion traffic.
3. **Self-attention Mechanism.** For the sake of better filter out the features to get the network flow volume from the CNN layer, which is helpful to realize more accurate network traffic classification, we leverage the self-attention mechanism. The attention mechanism is used to analyze the importance of the packet vector and to obtain more prominent fine-grained features for malicious traffic detection.
4. **LSTM Layer.** Fundamentally, LSTM is an improved version of RNN proposed by [6] and further improved by [9]. As shown in Fig. 3, the LSTM layer in our model consists of three LSTM units (input gate, output gate and forget gate) with end-to-end training.
5. **Fully-connected Layer.** After the above several layers of training, at the bottom layer, we use a fully-connected layer composed of multiple neurons to extract key features.

Our target is to design a stable model for network intrusion detection. To achieve this goal, we employ 1 CNN layer to capture the features for the net-

work intrusion traffic. The importance of packet vectors is analyzed by using the attention mechanism to obtain more prominent fine-grained features for malicious traffic detection. Moreover, based on the timing dependence of network intrusion data, the LSTM layer is used to extract the timing characteristics of network intrusion data. Finally, at the bottom of the model, a fully-connected layer is utilized. It can reduce the dimension of spatial representations used for detection.

5 Implementation

In this section, we present the details on the implementation of CNN-A-LSTM.

5.1 Data Preprocessing

In this paper, we choose KDDcup99 dataset to conduct the intrusion detection experiments. First, we need to preprocess the dataset, due to the symbolic data attributes of KDDcup99 dataset. The dataset pretreatment has two processes, namely: numerical standardization and numerical normalization.

Numerical Standardization First, the average value and average absolute error of each attribute can be calculated by the following Eq. (1) and Eq. (2), respectively:

$$\overline{a_k} = \frac{1}{n}\sum_{p=1}^{n} a_{pk}, \tag{1}$$

$$T_j = \sqrt{\frac{1}{n}\sum_{p=1}^{n}(a_{pj} - \overline{a_j})^2}, \tag{2}$$

where a_j represents the mean value of the j-th attribute, T_j represents the average absolute error of the j-th attribute, and a_{pj} represents the j-th attribute recorded in p-th records.

A standardized measurement is then performed for each data record, as shown in Eq. (3):

$$Y_{pj} = \frac{a_{pj} - \overline{a_j}}{T_j}, \tag{3}$$

where Y_{pj} represents the j-th attribute value of the normalized p data record.

Numerical Normalization As shown in Eq. (4), each value is normalized to the interval $[0, 1]$:

$$a^* = \frac{a - d_{\min}}{d_{\max} - d_{\min}}, \tag{4}$$

where $d_{\max}$ is the maximum and $d_{\min}$ is the minimum value of the sample data, and a is the standardized data.

5.2 CNN

In general, network traffic (e.g., KDDcup99 dataset) consists of several different types of packets. Since the features in different types of packets are quite discrepant, we need to extract the features of them separately. In our CNN-A-LSTM model as shown in Fig. 3, we use a convolution layer to obtain the spatial features of each network traffic. The convolution layer can effectively maintain the spatial continuity and facilitate the extraction of local characteristics of network traffic. In order to reduce the dimension of the hidden layer and the computation of the subsequent layer, the pooling layer uses the max-pooling or mean-pooling, in addition to providing rotation invariance.

In particular, we also employ a one-dimensional CNN layer to process time series analysis and analyze signal data with a fixed length period, since the internal features in our model can be easily extracted and mapped from one-dimensional sequence data via the one-dimensional CNN layer. Therefore, due to the simplicity of the one-dimensional CNN layer, our model can further reduce the computational complexity.

5.3 Self-attention Mechanism

Then, in order to obtain the interdependent features of content over long distances, we use the self-attention mechanism. Using the self-attention mechanism, our model can pay more attention to the significant features of network intrusion in KDDcup99 dataset. This mechanism can act on the internal elements of the source or the target [30]. Therefore, it can improve the effectiveness of the learning features in the training phase via self-attention mechanism calculation.

Furthermore, self-attention is capable of linking the connection directly between any two parts of the relevant content through a calculation step in the calculation process. The distance between the long-distance dependence features is greatly shortened, which is conducive to the effective use of these features.

The equation for calculating the output result of self-attention is as follows,

$$u_t = \tanh\left(W \cdot h_t\right), \tag{5}$$

$$\alpha_{t,i} = \frac{\exp\left(\mathsf{score}\left(u_t, u\right)\right)}{\sum_{t=1}^{T} \exp\left(\mathsf{score}\left(u_t, u\right)\right)}, \tag{6}$$

$$s = \sum_{t=1}^{T} \alpha_t \cdot h_t. \tag{7}$$

In Eq. (5), h_t gets the output u_t. The allocation coefficient is obtained by comparing u_t in Eq. (6) with a trainable parameter matrix u (random initialization) used to represent context information. The softmax normalization is then carried out. Finally, as shown in Eq. (7), the focused vector s is obtained.

5.4 LSTM

In order to better extract and learn the characteristics of traffic bytes in each packet data, we add an LSTM layer after the self-attention mechanism. Using LSTM can greatly improve the efficiency of our experimental training. Since the dataset we used in the experiment has 41 attributes, the LSTM forget gate can discard the attributes with less correlation. In our model, the LSTM layer can make use of the previous information of the data for effective feature learning and can learn the sequential features within the traffic bytes. The traffic bytes of each packet are input into the LSTM layer sequentially. Therefore, we can get a vector for the packet data finally.

The key implementation of the LSTM layer in our model is to control the long-term state d. In particular, the LSTM layer controls the information transfer by comparing the internal storage unit d through the design of three gates (input gate, forget gate, and output gate). The first gate is the input gate to control the continuation of the long-term state d. The second gate is the forget gate to control the information preserving or discarding. For example, when the input information is satisfied the requirements of our model, the learning features will be retained. Otherwise, the feature will be forgotten by the forget gate. Moreover, the third gate is the output gate to control the long-term state d as the output of the current LSTM layer.

Equations for calculating the output result of the LSTM layer are as follows,

$$p_i = \sigma\left(X_p \cdot [q_{i-1}, n_i] + g_p\right), \tag{8}$$

$$a_i = \sigma\left(X_a \cdot [q_{i-1}, n_i] + g_a\right), \tag{9}$$

$$\widetilde{d}_i = \tanh\left(X_d \cdot [q_{i-1}, n_i] + g_d\right), \tag{10}$$

$$d_i = p_i \cdot d_{i-1} + a_i \cdot \widetilde{d}_i, \tag{11}$$

$$m_i = \sigma\left(X_m\left[q_{i-1}, n_i\right] + g_m\right), \tag{12}$$

$$q_i = m_i \cdot \tanh\left(d_i\right). \tag{13}$$

In the above equations, Eq. (8) is the formula for the forget gate, where the weight matrix of forget gate is X_p. The term $[q_{i-1}, n_i]$ represents a valid way to connect two vectors into a longer vector. Besides, the Eq. (9) refers to the calculation formula of the output gate. Meanwhile, we employ Eq. (10) to calculate the current input $\widetilde{d}_i$. Furthermore, we use Eq. (11) to calculate the current unit state d_i. Moreover, the impact of long-term memory on the current output is controlled by Eq. (12). Lastly, Eq. (13) refers to the final output of LSTM depends on both the output gate and the unit state.

6 Performance Evaluation

We conduct the experiments to evaluate the performance of the proposed CNN-A-LSTM model in this section. Most importantly, in Section 6.1, we present the detailed experimental setup and performance metrics. The comparison results of our proposed CNN-A-LSTM method with other baseline models are also presented. Finally, we conduct the parameter study in Section 6.2.

6.1 Preliminary Results

Experiment Settings *a) Dataset description:* We use the network intrusion data from KDDcup99 dataset to conduct the experiments. The attack types are divided into four categories, including Probe, Dos, U2R, R2L. Moreover, they can also be subdivided into 39 subcategories. In our experiment, we employ 22 types of attack data as a training set and the remaining data as a testing set [32].

b) Model setting: In our experiment, we conduct experiments by configuring 1 CNN layer, then add the self-attention mechanism and LSTM component. We vary different dropout values as well as the different number of epochs. In every model, we set dropout equal to 0.01, 0.1, 0.5, and epoch equal to 50, 100, and 1000.

c) Performance metrics: In these experiments, we adopt four metrics, namely Accuracy, Precision, Recall, and F1-score (F1) to compare the proposed model with other baseline models. In particular, we calculate these metrics via four parameters, including True positives (TP), False positives (FP), False negatives (FN), True negatives (TN).

- **Accuracy:** It is one of the most common metrics. The number of the correct samples is divided by the number of total samples. In general, the higher accuracy results represent better performance.

$$\mathsf{accuracy} = \frac{\mathsf{TP} + \mathsf{TN}}{\mathsf{TP} + \mathsf{FP} + \mathsf{TN} + \mathsf{FN}}, \tag{14}$$

- **Precision:** It is a measure of the accuracy of the algorithm. The rate of the number of the exact positive samples is divided into the number of actually positive samples.

$$\mathsf{precision} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FP}}, \tag{15}$$

- **Recall:** It is a measure of coverage. Obviously, the calculation method and result of recall rate are identical with the sensitivity.

$$\mathsf{recall} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FN}}, \tag{16}$$

- **F1:** It is an index used to evaluate the accuracy of the binary classification model, considering both precision and recall rate. It can be regarded as a harmonic average of precision rate and recall rate.

$$\mathsf{F1} = \left(1 + \beta^2\right) * \frac{\mathsf{precision} * \mathsf{recall}}{\beta^2 \left(\mathsf{precision} + \mathsf{recall}\right)}, \tag{17}$$

Table 1. Summary of test result for KDDcup99

Algorithm	Accuracy	Precision	Recall	F1
Naive Bayes	0.4533	0.93	0.07	0.61
Logistic Regression	0.7114	0.87	0.58	0.69
Decision Tree	0.8080	0.97	0.69	0.80
Random Forest	0.7759	0.97	0.63	0.76
KNN	0.924037	0.984880	0.908875	0.945352
DNN	0.930129	0.997936	0.915116	0.954733
CNN	0.929582	0.998391	0.914018	0.954343
LSTM	0.922038	0.943180	0.965217	0.954071
CNN-LSTM	0.934726	0.998423	0.920387	0.957818
CNN-A-LSTM	**0.945126**	0.997432	0.925494	**0.960117**

where $\beta = 1$ (means that Precision is as important as Recall).

Performance Comparison *Baseline models:* After performing these experiments, we select the following representative baseline models for comparisons with the proposed model.

- **KNN:** The main idea of KNN algorithm is to infer your category by your neighbors [26]. We use KNN to conduct the network intrusion detection experiments.
- **DNN:** DNN is a feedforward neural network with at least one hidden neural layer. For the comparison experiments, we use both ReLU and Sigmoid as the activation function and the threshold function, respectively. We set the dropout to be 0.01 and set 100 epochs.
- **CNN:** CNN is a feedforward neural network, which is usually composed of a convolution layer, a pooling layer, and a full connection layer. It can effectively utilize the two-dimensional structure of input data. In this paper, we use a CNN layer model to carry out the experiment of network intrusion detection.
- **LSTM:** LSTM is a time recursive neural network, which plays a very good role in processing and predicting data based on time series. We use the 1-layer LSTM model to conduct network intrusion detection experiments.
- **CNN-LSTM:** In CNN-LSTM and CNN-A-LSTM, we set dropout = 0.1 and also epoch = 100 to conduct the network intrusion detection experiment.

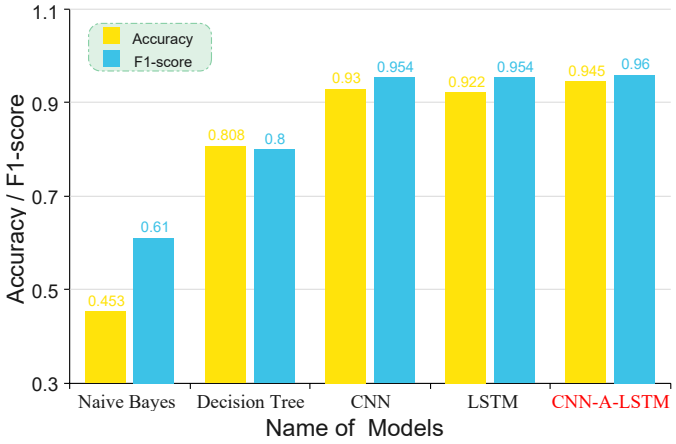


Fig. 4. Accuracy and F1-score of different models

We consider Keras as a wrapper on top of TensorFlow as the software framework [4]. The experiment is performed on a personal laptop MSI GL63 8RE, which has a configuration of an Intel Core i7-8750H CPU @ 2.20 GHz, 24 GB memory and using GPU acceleration. In the binary classification experiments, we have compared the performance with a Naive Bayes, Logistic Regression, Decision Tree, Random Forest, KNN, DNN, CNN, LSTM, CNN-LSTM, and CNN-A-LSTM. The results are as shown in Table 1.

From these experiments, we find that the CNN-A-LSTM model has the highest accuracy, compared with other algorithms. It is worth noting that although the DNN framework is very simple, it still shows its ability in dichotomy.

Moreover, we select Naive Bayes, Decision Tree, CNN, and LSTM and compare them with our CNN-A-LSTM model. Fig. 4 plots accuracy and F1-score and made the comparison. We can observe that our CNN-A-LSTM model achieves the highest scores in both accuracy and F1-score, implying the superior performance of our model.

6.2 Parameter Study

We next evaluate the impacts of parameters on the performance of our CNN-A-LSTM model. We use 1 CNN layer to do the experiment. Then, we add the LSTM layer and self-attention mechanism. In 1 CNN layer, we vary different dropouts (0.01, 0.1 and 0.5) and different epochs (50, 100 and 1000). As shown in Fig. 5, when dropout is 0.01 or 0.1, the accuracy of the training set is very high. In order to reduce the number of dropouts and improve the accuracy, dropout is selected as 0.1. Fig. 6 shows that when the dropout is equal to 0.1 and the dense is equal to 128, the accuracy of each model will be very high. So, we set dropout = 0.1, dense = 128 and epoch = 100 in CNN, CNN-LSTM as

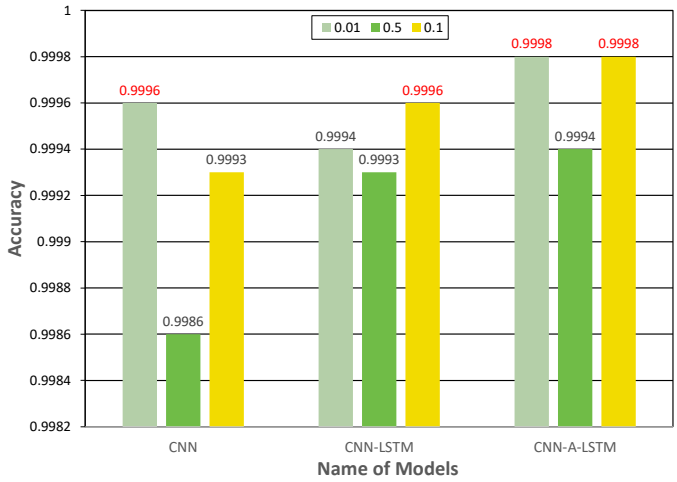


Fig. 5. Accuracy of different dropout of different models

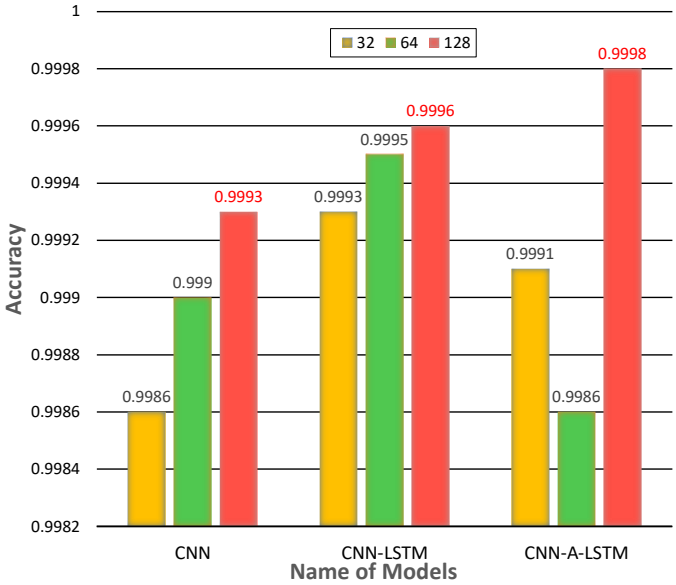


Fig. 6. Accuracy of different dense of different models

well as CNN-A-LSTM. Fig. 7 shows the accuracy of testing, and our model is the most stable and accurate one, and it demonstrates the best performance.

We also compare the CNN-A-LSTM model along with different models in every epoch in terms of Accuracy and Loss values. Fig. 8 shows the results. The accuracy of the CNN-A-LSTM model is always higher than those of the

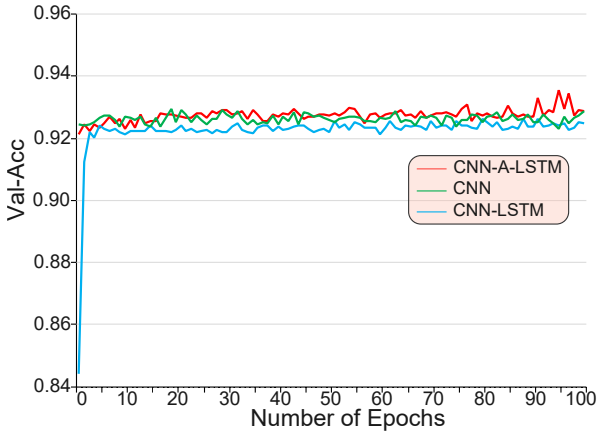
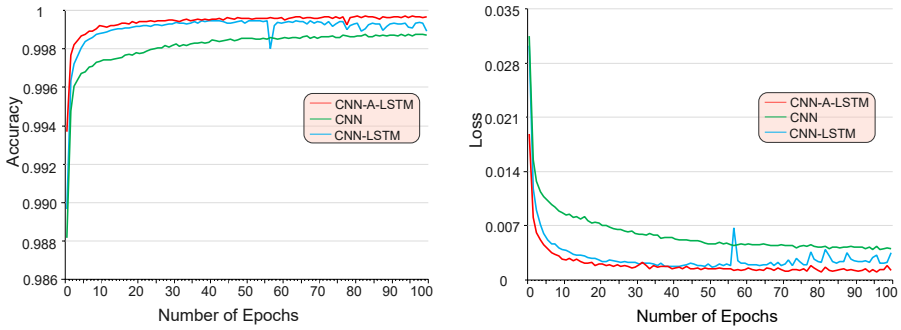


Fig. 7. Testing Val-Acc of different models



(a) Accuracy (b) Loss

Fig. 8. Accuracy (a) and Loss (b) of different models

other two models. It shows stable performance with the lowest loss. Therefore, its performance is the best among those models.

7 Conclusion

Cloud computing can provide users with elastic and ubiquitous computing and data storage services. However, cloud servers have also become targets for malicious attacks. The network intrusion detection is the prerequisite for taking countermeasures against malicious attacks. The recent advances in deep learning bring the chances to network intrusion detection. In this paper, we propose a new intrusion detection model namely CNN-A-LSTM by integrating CNN, LSTM, and the attention mechanism together. Experimental results show that

our CNN-A-LSTM model achieves superior performance compared with other existing methods. In the future, we will explore the usage of our CNN-A-LSTM model for more complex network intrusion detection tasks, such as multinomial classification.

Acknowledgement

The work described in this paper was partially supported by Macao Science and Technology Development Fund under Grant No. 0026/2018/A1.

References

1. Alfaro, J.G., Boulahia-Cuppens, N., Cuppens, F.: Complete analysis of configuration rules to guarantee reliable network security policies. International Journal of Information Security **7**(2), 103–122 (2008)
2. Behl, A.: Emerging security challenges in cloud computing: An insight to cloud security challenges and their mitigation. In: 2011 World Congress on Information and Communication Technologies. pp. 217–222. IEEE (2011)
3. Chawla, A., Lee, B., Fallon, S., Jacob, P.: Host based intrusion detection system with combined cnn/rnn model. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 149–158. Springer (2018)
4. Géron, A.: Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media (2019)
5. Ghaffari, F., Gharaee, H., Arabsorkhi, A.: Cloud security issues based on people, process and technology model: A survey. In: 2019 5th International Conference on Web Research (ICWR). pp. 196–202. IEEE (2019)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
7. Khalil, I.M., Khreishah, A., Azeem, M.: Cloud computing security: a survey. Computers **3**(1), 1–35 (2014)
8. Kim, G., Yi, H., Lee, J., Paek, Y., Yoon, S.: Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems. arXiv preprint arXiv:1611.01726 (2016)
9. Kim, J., Kim, J., Thu, H.L.T., Kim, H.: Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon). pp. 1–5. IEEE (2016)
10. Kimani, K., Oduol, V., Langat, K.: Cyber security challenges for iot-based smart grid networks. International Journal of Critical Infrastructure Protection **25**, 36–49 (2019)
11. Kumar, V., Sangwan, O.P.: Signature based intrusion detection system using snort. International Journal of Computer Applications & Information Technology **1**(3), 35–41 (2012)
12. Modi, C.N., Patel, D.R., Patel, A., Rajarajan, M.: Integrating signature apriori based network intrusion detection system (nids) in cloud computing. Procedia Technology **6**, 905–912 (2012)
13. Nadeem, M., Marshall, O., Singh, S., Fang, X., Yuan, X.: Semi-supervised deep neural network for network intrusion detection (2016)

14. Peng, K., Leung, V., Zheng, L., Wang, S., Huang, C., Lin, T.: Intrusion detection system based on decision tree over big data in fog environment. Wireless Communications and Mobile Computing **2018**
15. Rafique, W., Qi, L., Yaqoob, I., Imran, M., u. Rasool, R., Dou, W.: Complementing iot services through software defined networking and edge computing: A comprehensive survey. IEEE Communications Surveys & Tutorials pp. 1–1 (2020)
16. Roschke, S., Cheng, F., Meinel, C.: Intrusion detection in the cloud. In: 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing. pp. 729–734. IEEE (2009)
17. Roy, S.S., Mallik, A., Gulati, R., Obaidat, M.S., Krishna, P.V.: A deep learning based artificial neural network approach for intrusion detection. In: International Conference on Mathematics and Computing. pp. 44–53. Springer (2017)
18. Saenko, I., Kotenko, I.: Administrating role-based access control by genetic algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 1463–1470. Association for Computing Machinery (2017). https://doi.org/10.1145/3067695.3082509, https://doi.org/10.1145/3067695.3082509
19. Singh, A., Chatterjee, K.: Cloud security issues and challenges: A survey. Journal of Network and Computer Applications **79**, 88–115 (2017)
20. Singh, S., Jeong, Y.S., Park, J.H.: A survey on cloud computing security: Issues, threats, and solutions. Journal of Network and Computer Applications **75**, 200–222 (2016)
21. Sood, A.K., Enbody, R.J.: Targeted cyberattacks: a superset of advanced persistent threats. IEEE security & privacy **11**(1), 54–61 (2012)
22. Staudemeyer, R.C.: Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal **56**(1), 136–154 (2015)
23. Sun, Y., Zhang, J., Xiong, Y., Zhu, G.: Data security and privacy in cloud computing. International Journal of Distributed Sensor Networks **10**(7), 190903 (2014)
24. Takabi, H., Joshi, J.B., Ahn, G.J.: Security and privacy challenges in cloud computing environments. IEEE Security & Privacy **8**(6), 24–31 (2010)
25. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. pp. 1–6. IEEE (2009)
26. Vinayakumar, R., Soman, K., Poornachandran, P.: Applying convolutional neural network for network intrusion detection. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). pp. 1222–1228. IEEE (2017)
27. Wang, W., Du, X., Shan, D., Qin, R., Wang, N.: Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine. IEEE Transactions on Cloud Computing pp. 1–1 (2020)
28. Wang, W., Sheng, Y., Wang, J., Zeng, X., Ye, X., Huang, Y., Zhu, M.: Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access **6**, 1792–1806 (2017)
29. Xiao, Y., Xing, C., Zhang, T., Zhao, Z.: An intrusion detection model based on feature reduction and convolutional neural networks. IEEE Access **7**, 42210–42219 (2019)
30. Yang, R., Qu, D., Gao, Y., Qian, Y., Tang, Y.: nlsalog: An anomaly detection framework for log sequence in security management. IEEE Access **7**, 181152–181164 (2019). https://doi.org/10.1109/ACCESS.2019.2953981
31. Yin, C., Zhu, Y., Fei, J., He, X.: A deep learning approach for intrusion detection using recurrent neural networks. Ieee Access **5**, 21954–21961 (2017)

32. Zheng, W.F.: Intrusion detection based on convolutional neural network. In: 2020 International Conference on Computer Engineering and Application (ICCEA). pp. 273–277. IEEE (2020)