EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Network abnormal traffic detection method based on fusion of chord similarity and multiple loss encoder

Xiang Lv[1], Dezhi Han[1*], Dun Li[1*], Lijun Xiao[1] and Chin-Chen Chang[2]

*Correspondence: dzhan@shmtu.edu.cn; lidunshmtu@outlook.com

[1] College of Information Engineering, Shanghai Maritime University, Shanghai, China
[2] Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan

Abstract

Fog computing, as a new distributed computing framework, extends the tasks originally done in the cloud data center to the edge of the network and brings more serious security challenges while providing convenience. Abnormal network traffic detection is an effective means to defense malicious behavior, can detect a variety of known attacks. Although the application of deep learning method in the field of network abnormal traffic detection is easier than traditional machine learning methods, there are still problems of poor recognition accuracy and false alarm rate. In this paper, we use the semi-supervised network anomaly detection model (NADLA) that combines the long-short-term memory neural network method and the self-encoder method to solve this problem. NADLA analyzes network traffic through the time characteristics and behavior characteristics of traffic, and optimizes the accuracy and false alarm rate of network traffic classification. In addition, we improved the preprocessing method to improve the sensitivity of the trained model to network abnormal traffic. The NADLA model is tested on NSL-KDD dataset, and the results show that the proposed model can improve the accuracy and *F*1-value of network anomaly traffic detection.

Keywords: Network security, Intrusion detection, LSTM, Autoencoder, Deep learning, NSL-KDD, Fog computing

1 Introduction

Emerging IoT applications such as augmented reality (AR) and virtual reality (VR) wearables, smart homes, and the Industrial Internet all require low-latency, low-energy data communication and processing. The original cloud computing, where data is stored and processed at a distance from the end device, makes network communication latency high [1–3]. Fog computing, proposed by Cisco as an extension of cloud computing, is expected to meet these needs, as shown in Fig. 1. With small servers with storage, routing devices, gateways and other devices close to the end-user to process and respond to data directly, while allowing the cloud computing centre to manage the state of these devices and data processing asynchronously [4, 5]. In this way, the fog devices can respond to user requests in near real-time for general situations, while for special situations, they can be processed with the help of the cloud computing centre [6–11].

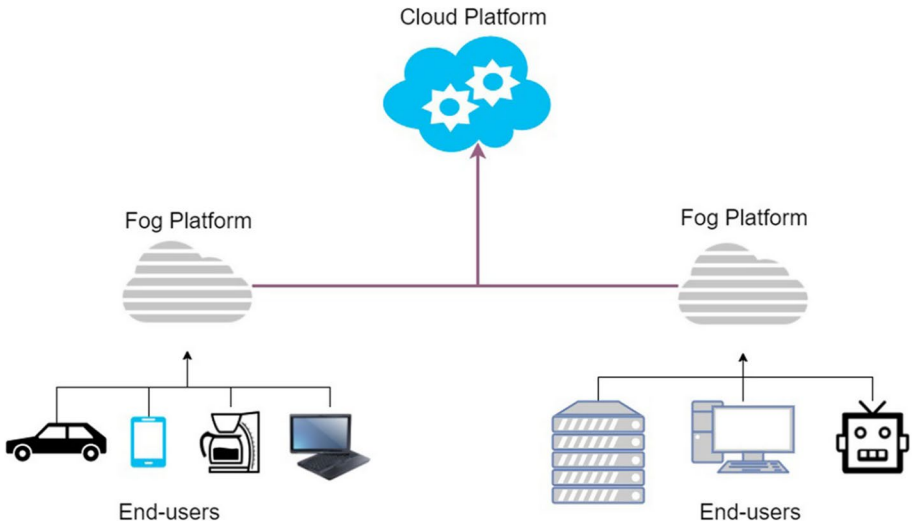

Fig. 1 Cisco's proposed fog computing model

However, the application of fog computing also brings some new issues, one of which is security. The fact that fog device nodes have hardware resources and the presence of valuable data information makes them an easy target for malicious entities [12]. Attackers exploit the characteristics of network communication protocol mechanisms to launch denial of service (DoS) attacks using large numbers of botnets sending carefully constructed network messages, which in turn make normal user device requests unanswerable [9, 13–19]. Such attacks can be catastrophic in an industrial IoT scenario [3, 15, 20–24]. In addition to this, there are also attacks such as override attacks, sniffing attacks, etc.

Currently, there are currently a number of countermeasures proposed to deal with cyber attacks, such as reducing the risk of intrusion by regularly updating passwords and by strictly verifying the identity of users and limiting their permissions to a fine-grained level, or using encryption technology to encrypt network communications so that attackers cannot analyse useful data, etc [25–28]. In addition to preventive methods, there are also countermeasures for network attacks that have already been launched, namely intrusion detection techniques [4, 29]. Intrusion detection techniques analyse certain indicators in network messages to determine whether they are attack traffic, and then filter and block them. Depending on how it is implemented, it can be divided into intrusion detection techniques based on feature recognition and intrusion detection techniques based on anomaly detection [30–32]. Feature-based intrusion detection requires that the characteristics of various network attacks are stored in a database in advance, and then the upcoming network is tested for the presence of network attack characteristics to determine whether it is a network attack [33–35]. For known attacks, feature-based intrusion detection techniques can accurately discriminate, but for unknown attacks, it is completely helpless, so the database of features of network attacks must be constantly updated so that new attacks can be detected. Anomaly-based intrusion detection technology identifies network attacks by determining how much network traffic deviates from normal traffic. It does not require pre-saving the characteristics of

various network attacks or real-time updating of the network attack characteristics database, so it is cheaper and more suitable for the current network environment.

Besides, the rapid development of data science and artificial intelligence technology and its excellent performance in natural language processing, image processing and other fields make it a research hotspot. By combining the specific characteristics of network traffic, many researchers propose many network intrusion detection methods based on artificial intelligence. These methods show that data science and artificial intelligence technology are powerful tools to solve the problems and challenges brought by network attacks. Deep learning, as a machine learning technology based on representation learning idea, does not require artificial feature design and feature extraction. It only needs to input the original data to the neural network, and it will automatically learn the high-level information in the original data [36, 37].

Although many researchers have done extensive research on the application of machine learning and deep learning in network intrusion detection and achieved remarkable results, there is still a problem of low accuracy in semi-supervised automatic encoder model. In this paper, we propose a semi-supervised intrusion detection model combining long-term and short-term memory neural network (LSTM) and automatic encoder (AE). It is a two-stage detection technique to carry out the classification problem of network traffic. In terms of data pre-processing, this model (NADLA) not only transforms the non-numerical features in the samples into numerical features and fixes the range of data values in a small interval, but also increases the sensitivity of the model to anomalies by using the 3-sigma idea to remove anomalies for the AE sub-model training data. For anomaly detection, the LSTM sub-model is first used to separate the sample set into normal and anomalous samples by using high-level information from the time series, and then the AE sub-model is used to collect the high-level information from the samples considered normal by the LSTM sub-model. The classification results are obtained by comparing the error between the input samples and the reconstructed samples with the set threshold. Experimental results on the NSL-KDD dataset show that the NADLA model achieves an average accuracy of 92.79% and an $F1$-score of 93.73%. The advantage of the model is that it uses a self-encoder model to reduce the reliance on labelled datasets to a certain extent, while the LSTM model is used to learn the temporal features in the dataset to further improve the performance of the model in terms of accuracy and recall. In addition, the model structure is optimised by conducting a large number of comparison experiments, allowing the model to be trained efficiently.

Specifically, the main contributions of this study are as follows.

- This paper proposes a new network anomaly detection model (NADLA) incorporating LSTM and AE, which is capable of learning both temporal information in the data and high-level features of normal data.
- In this paper, NADLA improves the data pre-processing method by not only performing data coding and normalisation operations but also introducing the removal of specific points operation, which has the effect of significantly improving the accuracy of the trained model in detecting anomalies.
- We further investigated the effect of different design structures and parameter settings in the NADLA model on the accuracy of the model in detecting anomalies.

The rest of this paper is organized as follows. Section 2 discusses the related work. This is followed by Sect. 3, which specifies the design details of the proposed NADLA model. In Sect. 4, we describe the experimental methods, including the dataset, pre-processing operations Then, Sect. 5 presents the experimental results. Finally, Sect. 6 summarises the work and suggests directions for future works.

2 Related Work

The research in the field of network intrusion detection can be divided into two parts. The first part uses the original machine learning method to detect network traffic, and the other part uses the deep learning method to detect network traffic. This section will introduce the related research results of these two parts.

2.1 The machine learning-based approach to network anomaly detection

Over the past two decades, many researchers have conducted various studies on network anomaly detection. Tavallaee et al. [38] conducted in-depth research on KDD-CUP99 and NSL-KDD datasets, and compared the performance of various machine learning methods on these datasets. Assiri et al. [39] proposed a random forest-based genetic algorithm for anomaly classification for network intrusion detection. Tao et al. [40] combined the genetic algorithm with SVM to improving the accuracy of the model and also reducing the model construction time. Chand et al. [41] proposed a stacked SVM model, which can effectively identify intrusions and outperform BayesNet, AdaBoost, SimpleCart and other classifiers proposed in past studies for intrusion detection.

After a single model encountered a bottleneck in improving classification accuracy, researchers changed their thinking to further improve detection accuracy. Agarwal et al. [42] proposed an integrated method combining three machine learning methods, Naïve Bayes, SVM and K-nearest neighbour (KNN), to improve classification accuracy and reduce processing time. Ling and Wu [43] used the random forest and trained an integrated method based on multiple classifiers using the selected features, which are effective in improving intrusion detection accuracy. Kim et al. [44] developed a hybrid system, which is useful to improve the accuracy of attack detection. Machine learning method requires artificial feature selection of network traffic, which may be complex and time-consuming. Therefore, researchers began to carry out scientific work in the field of network anomaly detection.

2.2 The deep learning-based approach to network anomaly detection

To date, researchers have paid increasing attention to deep learning methods, especially after the 2012 image classification competition made a splash and created a research boom in academia and industry [45]. Zhang et al. [46] proposed a hierarchical neural network model by fusing a modified LeNet-5 with an LSTM to allow the model to learn high-level features by extracting key components as training samples. The experimental results performed well. To address the problem of high false alarm rate, Imrana et al. [47] proposed a bi-directional long- and short-term memory neural network model (BiDLSTM) for intrusion detection experiments, which was innovated on the long- and short-term memory neural network model. Their experimental results showed good performance in metrics such as recall rate and $F1$-score.

For deep learning models, Shone et al. [48] evaluated the proposed model on the benchmark dataset KDD-CUP99, obtaining an attack classification accuracy of 97.87% and a false alarm rate of 2.15%. Ieracitano et al. [49] achieved an accuracy of 84.21% on the NSL-KDD dataset using data analysis and statistical methods and using a three-layer self-encoder model. In [50], the authors implemented different deep learning models, including self-encoders, RNNs and convolutional neural networks. The authors implemented a self-encoder-based anomaly detection model in [51] for automatic threshold learning and achieved an accuracy of 88.98%. Farahnakian and Heikkonen [52] proposed a deep self-encoder (DAE) model containing four self-encoders, which are trained using a hierarchical approach to prevent overfitting and local optima. In addition, they classify the incoming network input flows into normal and abnormal, and activate the function through softmax.

3 Design and implementation of NADLA

This section describes the structure of the NADLA model and the relevant details. In Sect. 3.1, the structure and workflow of the proposed model are presented. In Sects. 3.2 and 3.3, the specific details of the sub-modules of the model are described.

3.1 Model structure

The proposed NADLA model is an original and successful synthesis of its LSTM and self-encoder models, with the LSTM and self-encoder sub-models producing fine-grained and quantitative improvements. It has undergone extensive testing, and results on the intrusion detection benchmark dataset NSL-KDD have been surprisingly encouraging. Figure 2 depicts the whole layout of the buildings.

The data sample initially enters the LSTM module of the NADLA model after being processed and translated into the time sequence format. The module is composed of two layers of LSTM nerve cells. Information from the sample may be successfully extracted using the LSTM neurons unit's door control mechanism and cell state. The door control system consists of three doors. The input goalkeeper data sample is computed using the vector computation and activation function. The inner neuron receives trustworthy information. The forgotten door examines the incoming data to find the optimal amount of information. Information status is carried out by function processing being activated. The LSTM submodel uses the data time information to label the data sample. The

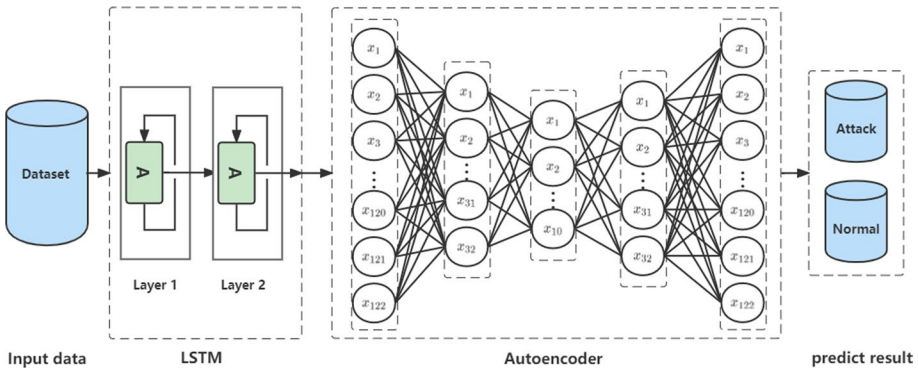

Fig. 2 Model structure of the NADLA

LSTM should then be provided with the self-encoder module while submitting the pre-standard as a data sample of the normal sample. The module is made up of two parts: an encoder and a decoder. The sample contains a lot of compressed, important data. The decoder, which is a sample data collection of the same dimension as the input samples, only uses this essential information. Due to information loss in important areas, both the original sample data and the reconstructed sample data will contain more or fewer errors. This error number may also be classified by comparing the model's training threshold against the sample.

Additionally improving is data pre-processing. To enable the features to be computed and quantified for the original samples of the data concentration, first transform the non-string feature value into the features of the value using the monopolymodes. then does data processing to guarantee speedy model convergence. Since the classification mechanism of self-encoder modules relies more on the high reconstruction error when the abnormal sample is corrected than it does on reconstructing errors from normal sample data, entering the module's data to the module's data will improve the module's generalization capabilities. The sample must be filtered at particular places. Figure 3 displays the distinctive overall process. The designs of the LSTM and self-encoder module will be addressed in detail in the next parts, and Sect. 4 will discuss the data pre-processing technique.

3.2 LSTM sub-model

LSTM is a modification of the RNN model. Gradient disappearance during training is an issue since only the BPTT approach, a BP algorithm that interprets data as a time series is employed to build conventional RNN models. This problem is addressed by the LSTM using a gating mechanism and cell states. The three control gates that make up the LSTM are the input gate, output gate, and forgetting gate. The input gate is made up

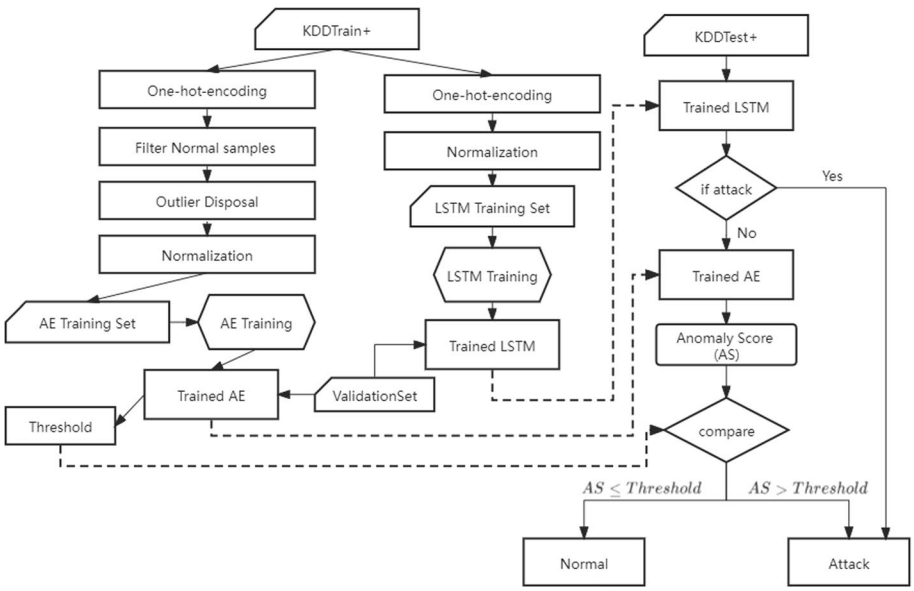

Fig. 3 The overall workflow of the model designed in this paper

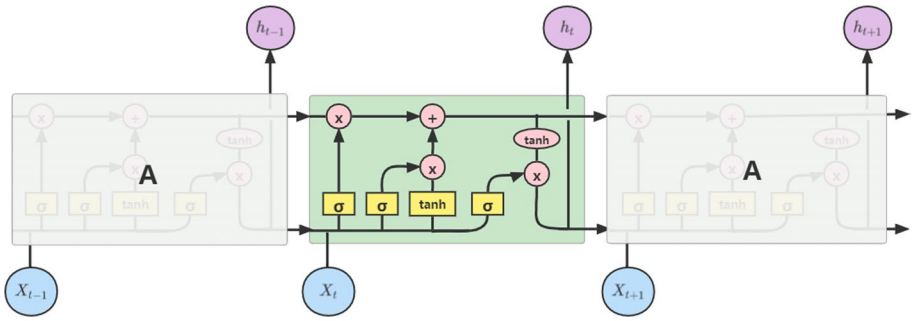

Fig. 4 LSTM structure

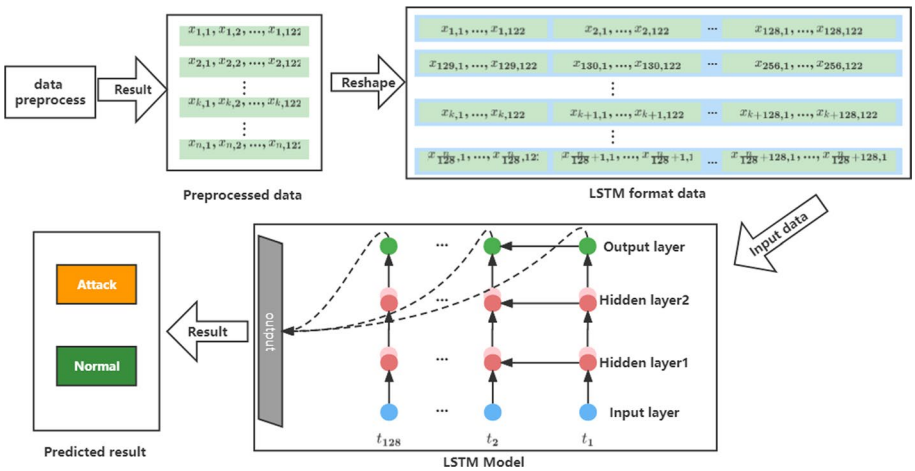

Fig. 5 Details of the designed LSTM sub-model

of a *sigmoid* function and a *tanh* function, which processes the beginning data and earlier information under pre-established principles to produce the information that has to be remembered right now. The forgetting gate selects which information should be forgotten by using the *sigmoid* function as an activation function. Figure 4 illustrates how the output gate multiplies the outputs of the *sigmoid* and *tanh* functions to create the information that is delivered to the LSTM neuron that follows [37].

The workflow of the LSTM sub-model in the NADLA model is shown in Fig. 5, which contains two LSTM implicit layer units. After data pre-processing, the dataset contains n samples, and each sample X_i is a d-dimensional vector as shown in Eqs. 1 and 2.

$$\text{dataset} = \{X_1, X_2, \ldots, X_n\} \tag{1}$$

$$X_i = \left\{x_{i,1}, x_{i,2}, \ldots, x_{i,d}\right\}, X_i \in \text{dataset} \tag{2}$$

Since the input format of the LSTM sub-model is a time series with a certain step size, it is necessary to reconstruct the input sample of the LSTM according to the step size of the original input data. We stipulate that the step size of the time series of the LSTM sub-model is 128. The 128 original samples are merged to form an input sample of LSTM sub-model, as shown in Eq. 3.

$$\text{LSTM input data} = \{S_1, S_2, \ldots, S_{n\%128}\} \tag{3}$$

where $S_i = \{X_{128\times i}, X_{128\times i+1}, \ldots, X_{128\times i+127}\}$.

The first step for each sample S_t to enter the LSTM sub-model is to determine how much information to discard through the forgetting gate. By reading the data information H_{t-1} of the previously hidden layer and the data information S_t of the input layer, and using the activation function to output a value between 0 and 1, it indicates how much information the previous LSTM unit has retained, as shown in Eq. 4.

$$f_t = \sigma\left(W_f \cdot [H_{t-1}, S_t] + b_f\right) \tag{4}$$

Using Eq. 5, the output i_t of the input gate can be calculated by using the data information H_{t-1} of the previously hidden layer and the data information S_t of the input layer. Then use the t function to create a candidate value vector $\widetilde{C}_t$ and add it to the LSTM state, as shown in Eqs. 6 and 7.

$$i_t = \sigma\left(W_i \cdot [H_{t-1}, S_t] + b_i\right) \tag{5}$$

$$\widetilde{C}_t = \tanh\left(W_C \cdot [H_{t-1}, S_t] + b_C\right) \tag{6}$$

$$C_t = f_t * C_{t-1} + i_{t-1} * \widetilde{C}_t \tag{7}$$

Finally, the information enters the output gate and the output value is determined according to the cell state of the LSTM. We process the cell state C_t by the *tanh* function and multiply it with o_t to obtain the part of H_t where the cell of this LSTM implicit layer determines the output, as shown in Eqs. 8 and 9.

$$o_t = \sigma\left(W_o \cdot [H_{t-1}, S_t] + b_o\right) \tag{8}$$

$$H_t = o_t * \tanh\left(C_t\right) \tag{9}$$

The LSTM sub-model of the NADLA model contains two LSTM cell units, the first LSTM cell unit uses 16 hidden layer units, and the second LSTM cell unit uses 8 hidden layer units. In addition, each layer adopts batch regularization operation and 20% cell failure operation to avoid the gradient disappearance problem and overfitting problem. After the above operation on H_t, H_t' is obtained, and then the classification results are obtained by using the full connection layer of the softmax activation function, as shown in Eq. 10.

$$\{\{\text{Attack}\}, \{\text{Normal}\}\} \leftarrow \text{Softmax}\left(H_t'\right) \tag{10}$$

where {Attack} and {Normal} are the set of LSTM samples $\{S_1, S_2, \ldots, S_k\} \subset$ LSTM input data.

3.3 AE sub-model

A self-encoder is a self-supervised learning model that is trained using only input data and is widely used in the semi-supervised and unsupervised areas of machine learning.

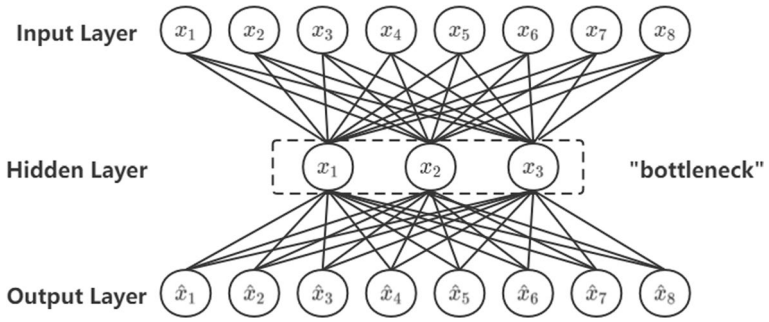

Fig. 6 Classical structure of AE

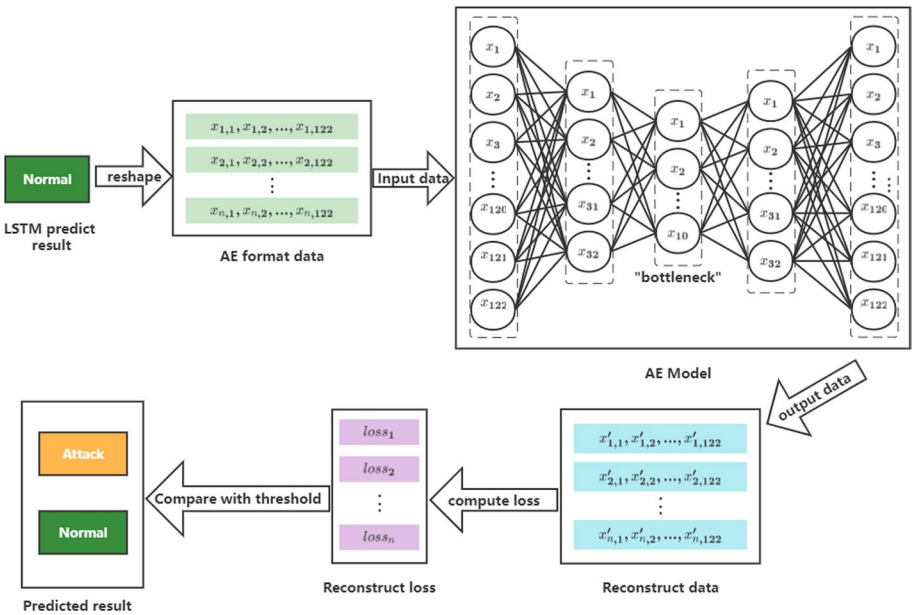

Fig. 7 The designed AE sub-model data processing process

The self-encoder network model f_θ can be sliced into two parts, the first part will try to learn the mapping relation $g_{\theta 1} : x \to z$, while the second part tries to learn the mapping relation $h_{\theta 2} : z \to \hat{x}$, the ultimate aim of the model is to make *hatx* and x as identical as possible. Thus, $g_{\theta 1}$ can be seen as a data encoding process that encodes the original high-dimensional sample data into a low-dimensional hidden variable z, and $h_{\theta 2}$ as a data decoding process that uses the encoded low-dimensional hidden variable z to decode a reconstructed sample $\hat{x}$ that is as high-dimensional as the input sample x. The feedback mechanism for model learning then relies on the magnitude of the error value between $\hat{x}$ and x. Therefore, the usual structure of a self-encoder model is to have input and output layers of the same dimensionality, and the model generally contains multiple implicit layers inside the model, which have a clear decreasing and then increasing dimensionality structure, the simplified model of which is shown in Fig. 6.

The AE sub-model processing flow in the NADLA model is shown in Fig. 7. The auto-encoder model consists of two operations, encoding, and decoding. The AE sub-model is

a five-layer structure. The input and output layers are both 122-dimensional feature variables, and the hidden layer consists of three layers, in the order of 32, 10, and 32 dimensions. Training is performed unsupervised using small-batch stochastic gradient descent [44]. Each layer was operated by a regularisation operation and let 20% of the neurons fail operation to avoid the gradient disappearance problem and overfitting problem.

The samples in the dataset will be divided into normal and abnormal sample sets after processing by the LSTM sub-model, as shown in Eq. 10. After the analysis of LSTM classification results, we found that some samples in the normal samples are actually abnormal samples. NADLA model uses AE sub-model to further classify this part of the sample set. The input data format of the AE sub-model is the original data format. Thus, to match the input format of the AE sub-model, each LSTM sample S_1 in the LSTM-judged normal sample set $\{\text{Normal}\} = \{S_1, S_2, \ldots, S_k\}, k \leq n$ needs to be decomposed according to the dimensionality of the original samples, as shown in Eq. 11.

$$X_{i*128}, X_{i*128+1}, \ldots, X_{i*128+127} \leftarrow S_i, i \in [1, k] \tag{11}$$

where $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,d}\}$ and d indicates the number of sample features.

In the AE sub-model, during the encoding phase, the d-dimensional input layer data X_i goes through two implicit layers [39, 53] to compress the dimensionality thereby obtaining the high-level information Y_i of the data representation, as shown in Eq. 12.

$$Y_i = F_1(WX_i + b), i \in [1, k] \tag{12}$$

where F_1 is the encoder function, W denotes the weight matrix, and b denotes the bias vector.

In the decoding operation, the high-level information Y_i is remapped as a d-dimensional vector $\widehat{X}_i = (\hat{x}_{i,1}, \hat{x}_{i,2}, \ldots, \hat{x}_{i,d})$, as shown in Eq. 13.

$$\widehat{X}_i = F_2\left(W'Y_i + b'\right), i \in [1, k] \tag{13}$$

where F_2 is the decoder function, W' and b' denote the decoder weight and bias respectively.

In the encoding and decoding operations, the dissimilarity between the input data and the reconstructed data is reduced by continuously optimising the parameters of the neural network $\theta = (W, W', b, b')$. In the NADLA model, mean absolute error (MAE) is used to calculate the degree of dissimilarity between the data samples X_i and the reconstructed samples $\widehat{X}_i$. The calculation process of MAE is shown in Eq. 14.

$$\text{Loss}\left(X_i, \widehat{X}_i\right) = \frac{1}{d}\sum_{j=1}^{d}\left|x_{i,j} - \hat{x}_{i,j}\right| \tag{14}$$

The reconstruction error $\{l_1, l_2, \ldots, l_k\}$ of each sample can be obtained by Eq. 14. We choose the maximum value of the reconstruction error in the training set samples as the threshold q to determine whether the data samples are anomalous, as shown in Eq. 15.

$$q = \max\{l_1, l_2, \ldots, l_k\}, k \in AE\text{train dataset} \tag{15}$$

Table 1 Environment of the experiment

CPU	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
GPU	GTX 1050ti 4GB
RAM	16GB
OS	Ubuntu 16.04

Table 2 Data distribution of NSL-KDD data

NSL-KDD	Total	Normal	Attack
KDD_{Train+}	125,973	67,343	58,630
KDD_{Test+}	22,544	9711	12,833

In discriminating samples discriminated as normal by the LSTM, anomalies are determined by comparing the magnitude of their reconstruction error l_i concerning the threshold q, as shown in Eqs. 16 and 17.

$$X_i \in \{\text{Normal}\}, l_i \leq q \tag{16}$$

$$X_i \in \{\text{Attack}\}, \quad l_i > q \tag{17}$$

4 Experimental methods

In this section, we first introduce the data set used in this experiment, and then introduce the preprocessing method designed in this paper and the evaluation index of the experiment. The environment for our experiments is shown in Table 1.

4.1 Dataset

The dataset for the experiments in this paper is the NSL-KDD dataset, which is a benchmark dataset in the field of network intrusion detection that addresses some of the problems inherent in the KDD99 dataset [45]. Although the NSL-KDD dataset still has some problems [54] and may not reflect the current network environment well, it is still a valid benchmark dataset.

The dataset has two files, KDD_{Train+} and KDD_{Test+}. These two files are complete training set and test set of NSL-KDD, which contain a variety of types of attacks. Since it has been classified, in the experiment, we will directly use the samples in KDD_{Train+} to train, and use KDD_{Test+} to observe the relevant indicators of NADLA model. At the same time, a variety of category labels are reclassified into two categories, namely normal samples and abnormal samples. Each sample in this dataset contains 41 features, including 38 numerical types and 3 character types. The distribution of data sets after reclassification is shown in Table 2.

4.2 Pre-processing

Before model training, pre-processing procedures are required for the NSL-KDD dataset. These pre-processing procedures include the unique thermal encoding as well as

the regularisation operations. In addition to this, for the auto-encoder model, filtering of labels and outlier removal operations were performed. To improve the efficiency of model training, we convert non-numerical features into numerical features. We do this using a unique thermal encoding technique, which converts non-numerical features into n features, with n representing the number of values taken by the non-numerical feature. Therefore, the three string features in the NSL-KDD sample set will become 84 features after the unique hot encoding, of which 3 are protocol_type, 70 are service and 11 are flag. In the NSL-KDD data set, there are 38 numerical features in addition to the above three non-numerical features. In addition, there are 38 numerical features in the NSL-KDD dataset. Therefore, after calculating the remaining data features and performing exclusive thermal coding, there will be 122 features in total. Since each feature has a different range of values, to eliminate the effect of different scales for different features and thus reduce the execution time for model training, we use maximum–minimum normalization, which is calculated as shown in Eqs. 18 and 19. This method will map each feature into a new interval.

$$X_{\text{std}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{18}$$

$$X_{\text{scaled}} = X_{\text{std}} * (\max - \min) + \min \tag{19}$$

where max and min are (0,1), so each feature maps to the value between [0,1].

For the LSTM model, only the above two operations are needed to complete the preprocessing work, but for the autoencoder (AE), before the maximum and minimum regularization, it is necessary to filter labels and remove outliers. The filter label is to retain that the label is a "normal" sample because the auto-encoder only uses the normal sample in training so that the effect of auto-encoder training can have a smaller reconstruction error for the "normal" sample, and a larger reconstruction error for the "abnormal" sample, to identify attacks. By analyzing the features in the NSL-KDD dataset, we can assume that the features in the dataset are independent of each other and obey a normal distribution. Moreover, normal network behaviour, because it is specified at design time, will in most cases not deviate too much from the value of the reference metric, so we use 3-sigma theory to make the determination. 3-sigma theory is also known as 68–95–99.7 theory. This rule states that 68% of instances lie within one standard deviation of the mean, 95% lie within two standard deviations, and 99.7% lie within three standard deviations [55]. Moreover, normal network behaviour, because it is specified at design time, will in most cases not deviate too much from the value of the reference metric, so we use 3-sigma theory to make the determination [56]. 3-sigma theory is also known as 68–95–99.7 theory. This rule states that 68% of instances lie within one standard deviation of the mean, 95% lie within two standard deviations, and 99.7% lie within three standard deviations [57, 58].

In this paper, we specify that if a feature takes a value outside of 3sigma (99.7%), it is an outlier and that sample is removed [53]. Algorithm 1 describes the outlier process used. Since the outlier removal is done on the "normal" samples in the KDDTrain+ dataset, the training set sample is reduced from 67,343 to 41,761 after the outlier removal. This remaining data will be used for the training of the self-encoder.

Algorithm 1 Outlier Disposal

Input: $\{X_1, X_2, \ldots, X_m\}$: samples of NSL-KDD dataset, $\{f_1, f_2, \ldots, f_{122}\}$: features in the sample, $\{\bar{f}_1, \bar{f}_2, \ldots, \bar{f}_{122}\}$: average of each feature, $\{\sigma_1, \sigma_2, \ldots, \sigma_{122}\}$: standard deviation of each feature

```
for X_i ∈ {X_1, X_2, ..., X_m} do
    for f_j ∈ {f_1, f_2, ..., f_122} do
        if |X_i · f_j − f̄_j| > 3 · σ_j then
            delete X_i
            break
        end if
    end for
end for
```

4.3 Evaluation metrics

To verify the performance of the model proposed in this paper, the classification accuracy, precision, recall rate, $F1$ score, and other indicators are used. The attack sample is regarded as category 0, and the normal sample is regarded as category 1. The confusion matrix is shown in Table 3. Among them, true-positive (TP) denotes the case of correctly marked as the first category, that is, the case of correctly marked as an attack. True-negative (TN) denotes the situation where the correct label is the second type, that is, the situation where the correct label is normal. False positive (FP) denotes the case of class 0 marked incorrectly as class 1, while false negative (FN) denotes the case of class 0 marked incorrectly as class 1.

Accuracy (Acc) measures the proportion of correct predictions and represents the number of correctly classified samples as a proportion of the total number of samples in a given dataset, as shown in Eq. 20.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{20}$$

True positive rate (TPR), also known as recall or sensitivity, reflects how many abnormal samples are identified out of all abnormal samples and is calculated as shown in Eq. 21.

$$\text{TPR/Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{21}$$

Precision indicates how many of the data marked as attack samples are true attack samples out of the total number of data marked as attack samples, as shown in Eq. 22.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{22}$$

Table 3 Data distribution of NSL-KDD data

Total samples	**Predicted class**	
	Attack/0	**Normal/1**
Actual class		
Attack/0	TP	FN
Normal/1	FP	TN

$F1$-score is a measure of test accuracy, calculated by taking the summed average of precision and recall, as shown in Eq. 23.

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{23}$$

5 Experiment and results

In this section, we present the exact steps of the experimental execution and discuss the results.

5.1 The performance of NADLA

We first measured the performance of the model proposed in this paper (NADLA) using the method mentioned above. Considering the possible volatility and chance in the performance of the model, all experiments below will be repeated ten times and then averaged. Table 4 shows the performance of the NADLA model under the above-mentioned metrics. While earning an average performance of 93.73% in F1, it achieved 92.79% in Accuracy.

We also examined the performance of the sub-modules individually in classifying the data samples because the NADLA model includes both LSTM and self-encoder sub-modules. The results are displayed in Table 5 and allow us to precisely quantify the impact of each module on the overall model in terms of each metric. According to the results, the self-encoder and LSTM sub-modules individually underperform the NADLA model as a whole by more than 3% on each metric. The performance of the NADLA as a whole was then further examined in detail, namely how the LSTM and self-encoder sub-modules enhanced it. We were able to determine the answer to the question of why the LSTM sub-module can misclassify normal samples in the data samples as attack samples by analyzing the confusion matrix of NADLA and the two sub-modules. On the other hand, the AE sub-module wrongly classifies the attack samples in the data samples as normal samples since it is unable to distinguish them from other samples. The results of the confusion matrix are shown in Figs. 8 and 9.

In addition, we further analysed the labelling of specific samples by the model, and we found that the LSTM sub-module and AE sub-module were very reliable in making consistent judgements for attack samples, so the NADLA model cleverly exploited this and significantly improved the overall performance, compared to other semi-supervised machine learning methods as shown in Table 6. NADLA shows a 10–20% improvement over traditional machine learning methods and a 2–15% improvement over similar deep learning models. Additionally, the NADLA model beat the self-encoder model suggested by Wu et al. [55] in 2021 by 2%.

Table 4 Performance of NADLA Model

Metric	Value (%)
Accuracy	92.786
Precision	92.678
Recall	94.844
F-measure	93.734

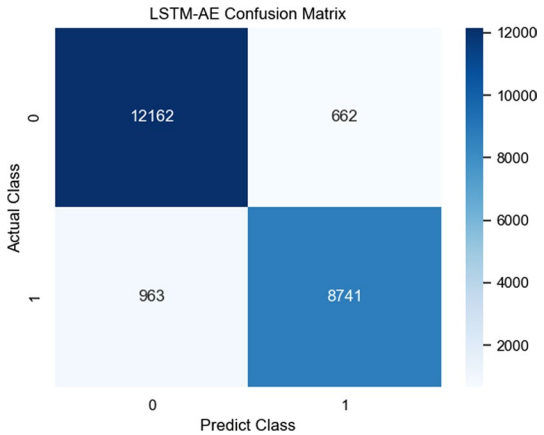

Fig. 8 Mixing matrix for each sub-model classification of the NADLA model

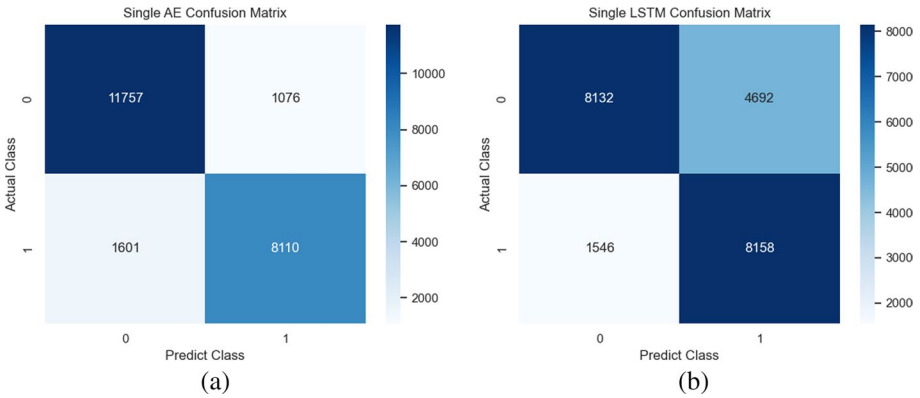

Fig. 9 Mixing matrix for NADLA model classification

Table 5 Comparison of individual sub-models and overall model metrics

Model	Accuracy (%)	Precision (%)	Recall (%)	*F*1 score (%)
LSTM	72.146	85.402	63.066	72.416
AE	88.126	88.026	91.618	89.76
NADLA	**92.786**	**92.678**	**94.844**	**93.734**

Bold indicates the performance of the NADLA model under each metric

Since the input to the LSTM is a time series, n samples need to be concatenated first and last into one LSTM sample, followed by the input to the LSTM model. The input and forgetting gates of the LSTM sub-model selectively extract some of the information from the input samples to add to the neural unit, and the length of the input samples affects the process of information extraction by the LSTM sub-model. To explore the effect of the number of original samples contained in an LSTM sample on the performance of the NADLA model, we conducted further experiments, the results of which are shown in Fig. 10. The experiments shows that the number of original samples included in the LSTM input samples did not show a significant correlation with

Table 6 Summary of the performance of each model on the NSL-KDD dataset

No.	Method	Accuracy (%)	Type
1	Multi-layer Preceptron [38]	77.41	Deep Learning based methods
2	Recurrent Neural Network [54]	83.28	
3	AE+Guassian Naïve Bayes [59]	83.34	
4	STL+SVM [60]	84.96	
5	AE+SMR [61]	88.39	
6	AE by Sadef et al. [51]	88.98	
7	AE by Xu et al. [55]	90.61	
8	DNN [51]	89.00	
9	DCNN [62]	84.58	
10	SVM [38]	69.52	Conventional classification methods
11	Naïve Bayes [38]	76.56	
12	J48 [38]	81.05	
13	NB tree [38]	82.02	
14	Random forest [38]	80.67	
15	Random tree [38]	81.59	
16	Fuzzy approach [56]	84.12	
17	NADLA	**92.79**	

The accuracy of the model we present is marked with bold font

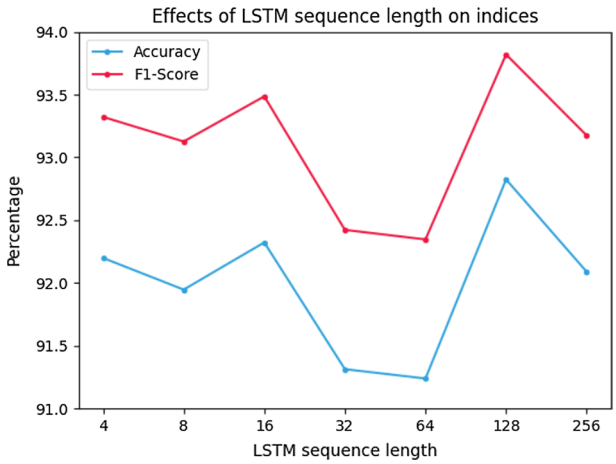

Fig. 10 Performance of the proposed NADLA model with different number of LSTM samples containing original samples

the model performance, so we finally selected the time series length of 128 for the LSTM input samples.

5.2 The metrics results of NADLA

Because the input format of the LSTM sub-module is the time sequence, we cleverly take each data sample as a time point, and the multiple data samples form a time sequence as the input of the LSTM sub-module. The input doors in the LSTM subsub-module process the time sequence to get the information that needs to be memory, and the forgetful door will abandon some of the information in the time sequence, so the length of the time sequence will have a certain impact on the NADLA model. To quantify the specific relationship between the number of time points and models contained in the time sequence, we have selected seven 2 index multiple values for analysis. The experimental results are shown in Fig. 11. The length of the time sequence is not clearly correlated with model performance, as can be seen from the results, but it also turns out that it is not longer and better, so we ultimately decide to set the length of the time sequence for LSTM input to be 128.

Additionally, we investigated the effect of the sub-module design on the performance of the NADLA model. The major factors in architectural design are the network's layer count and the number of neurons in each layer. With the correct architecture, it will be simpler to avoid either overfitting or underfitting. The experimental results from our examination of how the number of neurons in each layer affects the model metrics for the LSTM sub-module, which we created using Sunanda Gamage's paper [63], are shown in Fig. 11. The LSTM sub-module generates results that are insufficient when more neurons are added, thus we chose the structures of the first and second LSTM cells, which each employ 16 and 8 hidden layer cells, respectively. For the self-encoder structure comparison study, we selected the most popular peer structures. The outcomes are shown in Table 7. The self-encoder operates better on average when it has three hidden layers, and the innermost layer functions best overall when the dimensionality is decreased to 10 dimensions, obtaining an accuracy of 92.13%, according to the findings analysis.

Finally, we investigate the thresholds for reconstruction errors in the self-encoder judgement sub-module. In our proposed model, after the training of the auto-encoder

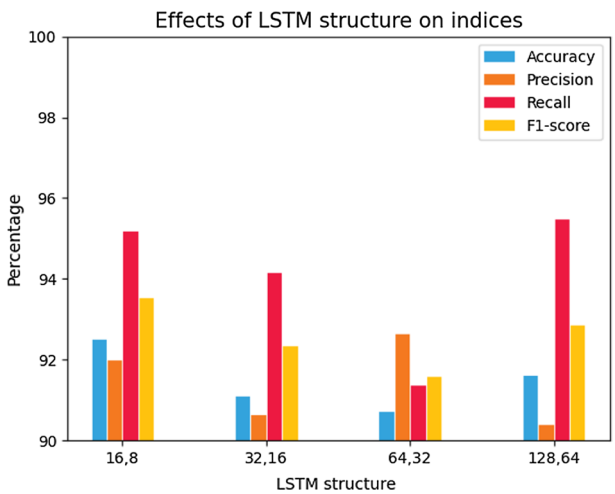

Fig. 11 Performance of the proposed NADLA model under different LSTM structures

Table 7 Performance of the proposed model under different AE structures

AE structure	Accuracy (%)	Precision (%)	Recall (%)	*F*1 score (%)
5	80.15	98.27	66.32	78.68
10	88.75	94.05	86.03	89.58
32, 5, 32	91.19	91.22	93.59	92.35
32, 10, 32	**92.13**	**92.00**	**94.39**	**93.17**
64, 5, 64	92.03	92.33	93.80	93.04
32, 16, 5, 16, 32	91.22	90.93	94.06	92.40
64, 32, 5, 32, 64	92.19	92.54	93.86	93.18

The best-performance AE structure is marked with bold styles

Table 8 Performance of the proposed model under different thresholds

AE threshold	Accuracy (%)	Precision (%)	Recall (%)	*F*1 score (%)
Min loss	83.904	78.604	98.598	87.466
Max loss	**91.644**	**91.714**	**93.812**	**92.744**
Mean loss	87.224	83.036	97.64	89.722

The best-performing AE threshold is marked with bold styles

sub-model, an auto-encoder trained with normal samples and an anomaly score threshold is obtained, and during testing, if a sample is reconstructed by the auto-encoder with a reconstruction loss higher than this anomaly score threshold, it will be judged as an attack sample. Therefore, the choice of the anomaly score threshold affects the classification accuracy of the auto-encoder. The choice of the abnormal score threshold is related to the reconstruction error obtained in the auto-encoder for all normal samples used for training. To investigate the impact of the selection of the abnormal score threshold on the proposed model, different methods of selecting the abnormal score threshold were chosen, three in total, the maximum, the minimum, and the average of the reconstruction errors of the normal samples used were selected as the abnormal score threshold. The results obtained by repeating the experiment are shown in Table 8. We found that the best classification result was obtained by selecting the maximum reconstruction error of the normal samples as the abnormal score threshold.

6 Conclusion

This paper proposes a semi-supervised network anomaly detection model, NADLA, which combines a long and short-term memory neural network (LSTM) and an autoencoder (AE). The LSTM sub-model uses its powerful temporal feature learning capability for anomaly detection and classification, and when the LSTM considers a sample to be normal, it is further fed into the AE sub-model to learn its high-level information to reconstruct the sample, and the reconstruction error is compared with a set threshold to obtain the final anomaly determination. The NADLA model has been averaged over several iterations of the NSL-KDD dataset to achieve an accuracy of 92.79% and an *F*1 score of 93.73%, which is better than other semi-supervised machine learning models. Considering the changing network attack, based on a semi-supervised learning model is still needed to play tag in the network traffic, and the task itself is complicated and

time-consuming, so the future research direction will focus on further improving the accuracy of the model and improved model structure research and unsupervised learning methods for network intrusion detection and related research.

Abbreviations

NADLA	Network anomaly detection model combining LSTM and autoencoder
LSTM	Long short-term memory neural network
AE	Autoencoder
AR	Augmented reality
VR	Virtual reality
SVM	Support vector machine
KNN	K-nearest neighbour
BiLSTM	Bi-directional long short-term memory neural network
DAE	Deep autoencoder
RNN	Recurrent neural network
TP	True positive
TN	True negative
FP	False positive
FN	False negative
ACC	Accuracy
NB tree	Naïve Bayes decision tree
DCNN	Deep convolution neural network
STL	Self-taught learning
SMR	Soft-max regression

Author contributions

XL proposed and developed the new idea of the paper and drafted it. HD has substantially revised it. LD conducted the data analysis and text combing. CC is responsible for supervision. All authors approved the submitted version. All authors read and approved the final manuscript.

Funding

This research is supported by the National Natural Science Foundation of China under Grant 61873160, Grant 61672338 and Natural Science Foundation of Shanghai under Grant 21ZR1426500.

Availability of data and materials

The dataset used for the experiments is the public dataset NSL-KDD, which is a dataset contributed by the Canadian Institute for Cybersecurity Research and is available for download at https://www.unb.ca/cic/datasets/nsl.html. It is an improvement on the KDDCUP99 dataset, with the redundant records present in the KDD99 dataset being trimmed and duplicates removed. The CUP99 dataset, on the other hand, is a network environment that Lincoln Laboratory simulated in 1998 by building an Air Force LAN and collecting TCPdump network connection and system audit data over a 9-week period, emulating various user types, different network traffic and attacks to make it resemble a real network environment. This raw data collected by TCPdump was divided into two parts: the training data over a 7-week period contained roughly 5,000,000+ network connection records, and the test data over the remaining 2 weeks contained roughly 2,000,000 network connection records. after the NSL-KDD dataset was processed, $\mathrm{KDD}_{\mathrm{Train}+}$ contained 125,973 records, $\mathrm{KDD}_{\mathrm{Test}+}$ contains 22,544 records, and each record contains 41 features, including 38 numeric types and 3 character types.

Declarations

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Received: 26 April 2022 Accepted: 21 September 2022
Published online: 22 October 2022

References

1. M. Cui, D. Han, J. Wang, An efficient and safe road condition monitoring authentication scheme based on fog computing. IEEE Internet Things J. **6**(5), 9076–9084 (2019)
2. M. Cui, D. Han, J. Wang, K.-C. Li, C.-C. Chang, Arfv: an efficient shared data auditing scheme supporting revocation for fog-assisted vehicular ad-hoc networks. IEEE Trans. Veh. Technol. **69**(12), 15815–15827 (2020)
3. H. Li, D. Han, M. Tang, A privacy-preserving charging scheme for electric vehicles using blockchain and fog computing. IEEE Syst. J. **15**(3), 3189–3200 (2020)
4. D. Han, N. Pan, K.-C. Li, A traceable and revocable ciphertext-policy attribute-based encryption scheme based on privacy protection. IEEE Trans. Depend. Secure Comput. (2020)

5. T. Xiao, D. Han, J. He, K.-C. Li, R.F. de Mello, Multi-keyword ranked search based on mapping set matching in cloud ciphertext storage system. Connect. Sci. **33**(1), 95–112 (2021)
6. M. Satyanarayanan, The emergence of edge computing. Computer **50**(1), 30–39 (2017)
7. P. Hu, S. Dhelim, H. Ning, T. Qiu, Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. **98**, 27–42 (2017)
8. C. Mouradian, D. Naboulsi, S. Yangui, R.H. Glitho, M.J. Morrow, P.A. Polakos, A comprehensive survey on fog computing: state-of-the-art and research challenges. IEEE Commun. Surv. Tutor. **20**(1), 416–464 (2017)
9. W. Liang, Y. Li, K. Xie, D. Zhang, K.-C. Li, A. Souri, K. Li, Spatial-temporal aware inductive graph neural network for c-its data recovery. IEEE Trans. Intell. Transp. Syst. (2022)
10. W. Liang, Y. Li, J. Xu, Z. Qin, K. Li, Qos prediction and adversarial attack protection for distributed services under dlaas. IEEE Trans. Comput. **2021**, 1–14 (2021)
11. S. Sarkar, S. Misra, Theoretical modelling of fog computing: a green computing paradigm to support iot applications. Iet Netw. **5**(2), 23–29 (2016)
12. D. Han, Y. Yu, K.-C. Li, R.F. de Mello, Enhancing the sensor node localization algorithm based on improved dv-hop and de algorithms in wireless sensor networks. Sensors **20**(2), 343 (2020)
13. W. Liang, J. Long, K.-C. Li, J. Xu, N. Ma, X. Lei, A fast defogging image recognition algorithm based on bilateral hybrid filtering. ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) **17**(2), 1–16 (2021)
14. W. Xiao, Z. Tang, C. Yang, W. Liang, M.-Y. Hsieh, Asm-vofdehaze: a real-time defogging method of zinc froth image. Connect. Sci. **34**(1), 709–731 (2022)
15. H. Li, D. Han, M. Tang, A privacy-preserving storage scheme for logistics data with assistance of blockchain. IEEE Internet Things J. (2021)
16. L. Zhang, X. Zhang, D. Li, H. Tan, Research on power quality control method of v2g system of electric vehicle based on apf, in *2019 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pp. 186–189 (2019). IEEE
17. Z. Li, S. Chen, S. Zhang, S. Jiang, Y. Gu, M. Nouioua, Fsb-ea: Fuzzy search bias guided constraint handling technique for evolutionary algorithm. Expert Syst. Appl. **119**, 20–35 (2019)
18. C. Diao, D. Zhang, W. Liang, K.-C. Li, Y. Hong, J.-L. Gaudiot, A novel spatial-temporal multi-scale alignment graph neural network security model for vehicles prediction. IEEE Trans. Intell. Transp. Syst. (2022)
19. S. Zhang, Q. Liu, Y. Lin, Anonymizing popularity in online social networks with full utility. Futur. Gener. Comput. Syst. **72**, 227–238 (2017)
20. M. Mukherjee, R. Matam, L. Shu, L. Maglaras, M.A. Ferrag, N. Choudhury, V. Kumar, Security and privacy in fog computing: Challenges. IEEE Access **5**, 19293–19304 (2017)
21. H. Liu, D. Han, D. Li, Fabric-iot: a blockchain-based access control system in iot. IEEE Access **8**, 18207–18218 (2020)
22. D. Li, D. Han, T.-H. Weng, Z. Zheng, H. Li, H. Liu, A. Castiglione, K.-C. Li, Blockchain for federated learning toward secure distributed machine learning systems: a systemic survey. Soft Comput. **2021**, 1–18 (2021)
23. D. Han, Y. Zhu, D. Li, W. Liang, A. Souri, K.-C. Li, A blockchain-based auditable access control system for private data in service-centric iot environments. IEEE Trans. Ind. Inf. **18**(5), 3530–3540 (2021)
24. H. Liu, D. Han, D. Li, Behavior analysis and blockchain based trust management in vanets. J. Parallel Distrib. Comput. **151**, 61–69 (2021)
25. D. Li, D. Han, Z. Zheng, T.-H. Weng, H. Li, H. Liu, A. Castiglione, K.-C. Li, Moocschain: a blockchain-based secure storage and sharing scheme for moocs learning. Comput. Stand. Interfaces **81**, 103597 (2022)
26. D. Li, D. Han, N. Crespi, R. Minerva, Z. Sun, Fabric-scf: A blockchain-based secure storage and access control scheme for supply chain finance. arXiv preprint arXiv:2111.13538 (2021)
27. Y. Zhang, X. Chen, D. Guo, M. Song, Y. Teng, X. Wang, Pccn: parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows. IEEE Access **7**, 119904–119916 (2019)
28. J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 Darpa intrusion detection system evaluations as performed by Lincoln laboratory. ACM Trans. Inf. Syst. Secur. (TISSEC) **3**(4), 262–294 (2000)
29. M. Li, D. Han, X. Yin, H. Liu, D. Li, Design and implementation of an anomaly network traffic detection model integrating temporal and spatial features. Secur. Commun. Netw. **2021** (2021)
30. S. Cai, D. Han, D. Li, Z. Zheng, N. Crespi, An reinforcement learning-based speech censorship chatbot system. J. Supercomput. **2022**, 1–23 (2022)
31. M. Li, D. Han, D. Li, H. Liu, C.-C. Chang, Mfvt: an anomaly traffic detection method merging feature fusion network and vision transformer architecture (2021)
32. S. Cai, D. Han, X. Yin, D. Li, C.-C. Chang, A hybrid parallel deep learning model for efficient intrusion detection based on metric learning. Connect. Sci. **2022**, 1–27 (2022)
33. X. Zhang, L. Zhang, D. Li, Transmission line abnormal target detection based on machine learning yolo v3, in *2019 International Conference on Advanced Mechatronic Systems (ICAMechS)* (IEEE, 2019), pp. 344–348
34. D. Li, D. Han, X. Zhang, L. Zhang, Panoramic image mosaic technology based on sift algorithm in power monitoring, in *2019 6th International Conference on Systems and Informatics (ICSAI)* (IEEE, 2019), pp. 1329–1333
35. R.A.R. Ashfaq, X.-Z. Wang, J.Z. Huang, H. Abbas, Y.-L. He, Fuzziness based semi-supervised learning approach for intrusion detection system. Inf. Sci. **378**, 484–497 (2017)
36. W. Wang et al., *Deep Learning for Network Traffic Classification and Anomaly Detection* (University of Science and Technology of China, Hefei, 2018), pp.61–75
37. Z. Ran, D. Zheng, Y. Lai, L. Tian, Applying stack bidirectional lstm model to intrusion detection. Comput. Mater. Continua **65**(1), 309–320 (2020)
38. M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (IEEE, 2009), pp. 1–6
39. A. Assiri, Anomaly classification using genetic algorithm-based random forest model for network attack detection. Comput. Mater. Continua **66**(1), 767–778 (2021)
40. P. Tao, Z. Sun, Z. Sun, An improved intrusion detection algorithm based on ga and svm. IEEE Access **6**, 13624–13631 (2018)

41. N. Chand, P. Mishra, C.R. Krishna, E.S. Pilli, M.C. Govil, A comparative analysis of svm and its stacking with other classification algorithm for intrusion detection, in *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring)* (IEEE, 2016), pp. 1–6
42. A. Agarwal, P. Sharma, M. Alshehri, A.A. Mohamed, O. Alfarraj, Classification model for accuracy and intrusion detection using machine learning approach. PeerJ Comput. Sci. **7**, 437 (2021)
43. J. Ling, C. Wu, Feature selection and deep learning based approach for network intrusion detection, in *The 3rd International Conference on Mechatronics Engineering and Information Technology* (2019)
44. G. Kim, S. Lee, S. Kim, A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Syst. Appl. **41**(4), 1690–1700 (2014)
45. Z. Zhou, Q.J. Wu, Y. Yang, X. Sun, Region-level visual consistency verification for large-scale partial-duplicate image search. ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) **16**(2), 1–25 (2020)
46. Y. Zhang, X. Chen, L. Jin, X. Wang, D. Guo, Network intrusion detection: Based on deep hierarchical network and original flow data. IEEE Access **7**, 37004–37016 (2019)
47. Y. Imrana, Y. Xiang, L. Ali, Z. Abdul-Rauf, A bidirectional lstm deep learning approach for intrusion detection. Expert Syst. Appl. **185**, 115524 (2021)
48. N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Comput. Intell. **2**(1), 41–50 (2018)
49. C. Ieracitano, A. Adeel, F.C. Morabito, A. Hussain, A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. Neurocomputing **387**, 51–62 (2020)
50. S. Naseer, Y. Saleem, S. Khalid, M.K. Bashir, J. Han, M.M. Iqbal, K. Han, Enhanced network anomaly detection based on deep neural networks. IEEE Access **6**, 48231–48246 (2018)
51. K. Sadaf, J. Sultana, Intrusion detection based on autoencoder and isolation forest in fog computing. IEEE Access **8**, 167059–167068 (2020)
52. F. Farahnakian, J. Heikkonen, A deep auto-encoder based approach for intrusion detection system, in *2018 20th International Conference on Advanced Communication Technology (ICACT)* (IEEE, 2018), pp. 178–183
53. F. Pukelsheim, The three sigma rule. Am. Stat. **48**(2), 88–91 (1994)
54. C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access **5**, 21954–21961 (2017)
55. W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, F. Sabrina, Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. IEEE Access **9**, 140136–140146 (2021)
56. G.S. Maddala, *Introduction to Econometrics* (Macmillan, New York, 1992)
57. Y. Wei, J. Jang-Jaccard, F. Sabrina, T. McIntosh, Msd-kmeans: A novel algorithm for efficient detection of global and local outliers. arXiv preprint arXiv:1910.06588 (2019)
58. Y. Wei, J. Jang-Jaccard, F. Sabrina, H. Alavizadeh, Large-scale outlier detection for low-cost pm10 sensors. IEEE Access **8**, 229033–229042 (2020)
59. M. Yousefi-Azar, V. Varadharajan, L. Hamey, U. Tupakula, Autoencoder-based feature learning for cyber security applications, in *2017 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017), pp. 3854–3861
60. M. Al-Qatf, Y. Lasheng, M. Al-Habib, K. Al-Sabahi, Deep learning approach combining sparse autoencoder with svm for network intrusion detection. IEEE Access **6**, 52843–52856 (2018)
61. A. Javaid, Q. Niyaz, W. Sun, M. Alam, A deep learning approach for network intrusion detection system. Eai Endorsed Trans. Secur. Saf. **3**(9), 2 (2016)
62. S. Bandyopadhyay, R. Chowdhury, A. Roy, B. Saha, A step forward to revolutionise intrusion detection system using deep convolution neural network (2020)
63. S. Gamage, J. Samarabandu, Deep learning methods in network intrusion detection: a survey and an objective comparison. J. Netw. Comput. Appl. **169**, 102767 (2020)

Publisher's Note