Computer Networks 169 (2020) 107049

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning

Ying Zhong[a], Wenqi Chen[a], Zhiliang Wang[a,f,*], Yifan Chen[b], Kai Wang[c], Yahui Li[d], Xia Yin[d,f], Xingang Shi[a,f], Jiahai Yang[a,f], Keqin Li[e]

[a] Institute for Network Sciences and Cyberspace at Tsinghua University, Beijing, China
[b] Beijing University of Posts and Telecommunications, Beijing, China
[c] University of Electronic Science and Technology of China, Chengdu, China
[d] Department of Computer Science and Technology at Tsinghua University, Beijing, China
[e] Department of Computer Science, State University of New York, New Paltz, USA
[f] Beijing National Research Center for Information Science and Technology, China

ARTICLE INFO

Article history:
Received 5 April 2019
Revised 11 November 2019
Accepted 9 December 2019
Available online 24 December 2019

Keywords:
Traffic anomaly detection
Ensemble learning
LSTM forecast
Deep learning
Adjustable threshold

ABSTRACT

Network traffic anomaly detection is an important technique of ensuring network security. However, there are usually three problems with existing machine learning based anomaly detection algorithms. First, most of the models are built for stale data sets, making them less adaptable in real-world environments; Second, most of the anomaly detection algorithms do not have the ability to learn new models again based on changes in the attack environment; Third, from the perspective of data multi-dimensionality, a single detection algorithm has a peak value and cannot be well adapted to the needs of a complex network attack environment. Thus, we propose a new anomaly detection framework, and this framework is based on the organic integration of multiple deep learning techniques. In the first step, we used the Damped Incremental Statistics algorithm to extract features from network traffic; Second, we train Autoencoder with a small amount of label data; Third, we use Autoencoder to mark the abnormal score of network traffic; Fourth, the data with the abnormal score label is used to train the LSTM; Finally, the weighted method is used to get the final abnormal score. The experimental results show that our HELAD algorithm has better adaptability and accuracy than other state of the art algorithms.

1. Introduction

The importance of intrusion detection systems (IDS) is critical because networks can be vulnerable to attacks from internal and external intruders [1,2]. Network traffic anomalies will lead to a decline in network communication performance and network service interruption. The definition of network anomalies is that the current network traffic is seriously deviating from normal traffic. Network anomalies are mainly caused by malicious network attacks, e.g. Denial of Service (DoS), Distributed Denial of Service (DDoS), port scan, worm propagation, etc., as well as network configuration errors and other exception [3] caused by the interruption of the line. As a detection system put in place to monitor computer networks, IDS has been in use since 1980s [4]. By analysing patterns of captured data from a network, IDS helps to detect threats [5]. Traffic anomaly detection has always been the research direction of network security academics and industry, and many related detection methods and systems have been developed.

The constant change of the attack mode makes it more difficult to solve the traffic anomaly detection problem. Traditional intrusion detection tools, such as rule-based Snort [6], are no longer able to meet the growing demand for network security. We need to design a smarter intrusion detection tool. This anomaly detection tool requires the ability to learn dynamically and requires environmental adaptation to defend against unknown attacks.

The current mainstream method of traffic anomaly detection is machine learning. Our design choices will be analyzed from different categories of machine learning.

(1) Machine learning methods can be divided into shallow machine learning and deep learning according to the number of layers of neural networks involved: Shallow machine learning [7] has the advantage of short training time, and deep learning [8] has stronger representation ability. The trend of GPU [9,10] acceleration methods allow us to choose a deep learning approach.

* Corresponding author.
E-mail addresses: wzl@cernet.edu.cn, zhongy18@mails.tsinghua.edu.cn (Z. Wang).

https://doi.org/10.1016/j.comnet.2019.107049
1389-1286/

(2) Specific classification tasks can be divided into single classifiers and ensemble learning classifiers: The idea of ensemble learning is to improve machine learning performance by combining multiple models, which is better than a single model in common sense. Ensemble learning is the research hotspot of machine learning in the field of traffic anomaly detection [11,12]. The ensemble learning model is superior to the single model in both predictive power and generalization ability, so ensemble learning is introduced in the design of our model.

(3) Choose supervised learning or unsupervised learning: Supervised learning [13,14] interacts with the external environment through a label-guided approach, so that the trained model can better integrate into human domain knowledge. Therefore, it is necessary to incorporate a supervised learning model.

Based on the above design choices, we have the following challenges:

(1) Ensemble learning is well used in the field of anomaly detection. However, deep learning method has not been used as component learners [61] of heterogeneous ensemble learning in network intrusion detection.

(2) Algorithms based on supervised learning require a large number of labeled training data to obtain good detection results. However, real network traffic data lacks of a large number of truly labeled data sets, which makes it difficult to use supervised deep learning.

(3) The attack environment of the network changes constantly. If the model does not have the ability to relearn, the performance of the detector will decrease.

(4) Many of the deep learning based anomaly detection models have not been evaluated with real traffic data. The main manifestation: the training data of anomaly detection model comes from the idealized data set, for example, the data set has been used for too long or it is just generated by the attack tool.

Inspired by the above observations, this paper attempts to absorb the advantages of heterogeneous ensemble learning and deep learning techniques and propose a more effective method.

To summarize, our main contributions in this paper are listed as follows:

• We have integrated various deep learning techniques and proposed the Heterogeneous Ensemble Learning Anomaly Detection (HELAD) algorithm framework. This framework is composed of four parts: feature dimension reduction, abnormal score generation, abnormal score prediction, and anomaly detection result combination. Each module can choose the appropriate technology according to its own design.

• We apply ensemble learning to anomaly detection. Specifically, the unsupervised Autoencoder and the supervised Long Short-Term Memory (LSTM) are combined in a heterogeneous way. The Autoencoder gains the profile of normal network traffic as one of the base learners, and provides learned RMSE as the label needed to train the LSTM. The LSTM can detect continuous attacks well, as it can record historical information and predict whether the attack will occur next time. In order to be able to use the supervised LSTM, we introduce the concept of a temporary label (TL), which is generated by an unsupervised Autoencoder.

• We introduce the concept of retraining time slices to retrain the model. This time slice is the time required to train the anomaly detection model in the previous round. We design dynamic thresholds and integrate learning parameters in the model so that the anomaly detection effect does not degrade.

• To evaluate the HELAD model, we conduct experiments on the latest data sets that reflect the real environment. And, we further evaluate our algorithm by comparing it with different state of the art algorithms. The experimental results consistently prove the superiority and competitiveness of our proposed model.

The rest of the paper is organized as follows. Section 2 presents the related work. We present our system model and problem formulation in Section 3. The model training and strategy optimization are presented in Section 4. We do the experiment and evaluate the performance using real traffic trace data in Section 5. We discuss some of the details of our models and experimental methods in Section 6. Finally, we conclude our work in Section 7.

2. Related work

In this section we review some literature work. Network intrusion detection is a classic network security issue. We focus on analyzing the related work from four aspects: traditional statistics, machine learning, deep learning and ensemble learning based methods. Next we discuss some of the literature for relearning.

We first discuss the traditional statistics method. There are several examples of statistical methods that are widely used in attack detection [15]. Lee et al. [16] proposed to use several information-theoretic measures, such as entropy, conditional entropy, relative conditional entropy, information gain, and information cost for anomaly detection. Other examples include the Cumulative Sum(CUSUM) algorithm[17], the exponentially weighted moving average (EWMA) algorithm [18], the Holt-Winter algorithm [19], and so on. Their advantage is that they do not need to know the prior knowledge of cyber attacks in advance. But most statistical methods rely on the assumption of a static detection process [20], which is not always realistic. The models in these references [15–20] are more suitable for anomaly detection of scenarios with small network environment changes. Our HELAD model incorporates prior knowledge and deep learning models to respond to changing environments. Such prior knowledge is embodied in the design of the features.

Next, we introduce some excellent methods based on machine learning. The purpose of machine learning is to create explicit or implicit models. In the field of anomaly detection, machine learning has the advantages of high detection rate and continuous learning and updating [21–23].

Deep learning [24] is the further development of neural networks. Deep learning uses a subsequent information processing layer in some hierarchies for classification or feature representation. For the strong presentation capabilities, existing IDS can be improved based on this latest technology. Many algorithms for anomaly detection of network traffic are based on deep learning. We will introduce the Autoencoder and Long Short-Term Memory (LSTM) that are most relevant to our work. About Autoencoder, Shone et al. [25] proposed nonsymmetric deep Autoencoder (NDAE) for unsupervised feature learning. Furthermore, they also proposed novel deep learning classification model constructed using stacked NDAEs. Khan et al. [26] proposed a novel two-stage deep learning model based on a stacked Autoencoder with a softmax classifier for efficient network intrusion detection. Specifically, their proposed model is able to learn useful feature representations from large amounts of unlabeled data and classifies them automatically and efficiently. Mirsky et al. [27] proposed a neural network based NIDS which has been designed to be efficient and plug-and-play. It accomplishes this task by efficiently tracking the behavior of all network channels, and by employing ensemble of Autoencoders for anomaly detection. On the other hand, Du et al. [28] proposed DeepLog, a deep neural network model utilizing LSTM, to model a system log as a natural language sequence. Wang et al. [29] proposed a novel IDS called the hierarchical spatial-temporal features-based intrusion detection system (HAST-IDS), which first learns the low-level spatial features of network traffic using deep convolutional neural networks (CNNs) and then learns high-level temporal features using LSTM. Jiang et al. [30] proposed a novel multi-channel intelligent attack detection

method based on long short term memory recurrent neural networks (LSTM-RNNs). The references [24–27] use Autoencoder for anomaly detection. Our HELAD model differs from these tasks. In our method, Autoencoder has two functions. One is to generate RMSE labels to train LSTM, and the other is to be one of the base classifiers. The references [28–30] all use LSTM as part of the model. They mainly consider LSTM for feature processing or time series establishment, but in our HELAD model, LSTM is used for RMSE prediction. As the labeled data set becomes larger, the prediction effect of LSTM will be better and better. Therefore, combining the LSTM and Autoencoder in our model to train the threshold of abnormal scores is better than a single deep learning model.

In addition, ensemble learning plays an important role in the field of anomaly detection [31–33]. In these works, many excellent anomaly detection methods are based on homogeneous ensemble learning [34–36]. Ensemble learning is also used in the most relevant paper in our research. However, this is a homogeneous learning based on stacking [27]. Moreover, the advantages of heterogeneous ensemble learning in the field of classification are constantly being explored [37]. For these reasons, we hope to seek better heterogeneous ways to make ensemble learning work well in the field of anomaly detection.

For relearning, we are mainly inspired by the following literatures. Papadimitriou et al. [53] proposed arbitrary window stream modeling method (AWSOM), which allows sensors in remote or hostile environments to efficiently and effectively discover interesting patterns and trends. They developed a one-pass algorithm to incrementally update the patterns. Ippoliti et al. [54] developed an enhanced dynamic anomaly detector for network traffic, which use auxiliary set to give online feedback to the model. Its importance lies in providing anomaly detection based on general traffic and keep updating constantly to achieve online adaptation in the meanwhile. Viegas et al. [55] presented BigFlow, a reliable stream learning intrusion detection engine that can maintain its accuracy over long periods of time. The labeled samples will be sent to an administrator periodically for judging, and the wrong-labeled ones will be used to re-train the model. Their solution evaluates the classification reliability, while it allows to incrementally update the intrusion detection engine. We conduct comparative analysis, finding that [53] maintains a window and updates the model when new instances reach. [54] uses false positives as an update condition. When the false positive rate increases, the auxiliary set is changed and the model is retrained. [55] specifies the period of the update. In order to reduce the cost of the update, the models are incrementally updated only with instances that were previously rejected. Because we are dealing with the actual complex network environment, we are looking forward to updating our HELAD model as quickly as possible. So we take the training time as an interval and add as much of the latest labeled data as possible to the training. Relearning is part of our model to prevent model degradation.

To sum up, the developments of deep learning and ensemble learning have brought new opportunities to the development of anomaly detection. We adopt these advantages to design a more intelligent and adaptable anomaly detection tool.

3. HELAD: Heterogeneous Ensemble Learning Anomaly Detection model

In this section, first motivation of this paper are presented. Next, we provide a basic model description that includes the meaning of statistical features, the meaning of formulas, and the way in which features are expressed. After that, we build the submodel for each part of the HELAD model. Finally, the abnormality detection result can be obtained by discriminant formula. Fig. 1 details the training process of the HELAD model and the abnormality determination process.

3.1. Motivation

In this section, we discuss our design choice of feature extraction and anomaly detection. Feature extraction has an important impact on the performance of the machine learning model. The network traffic has the following characteristics: 1) Packets of dif-

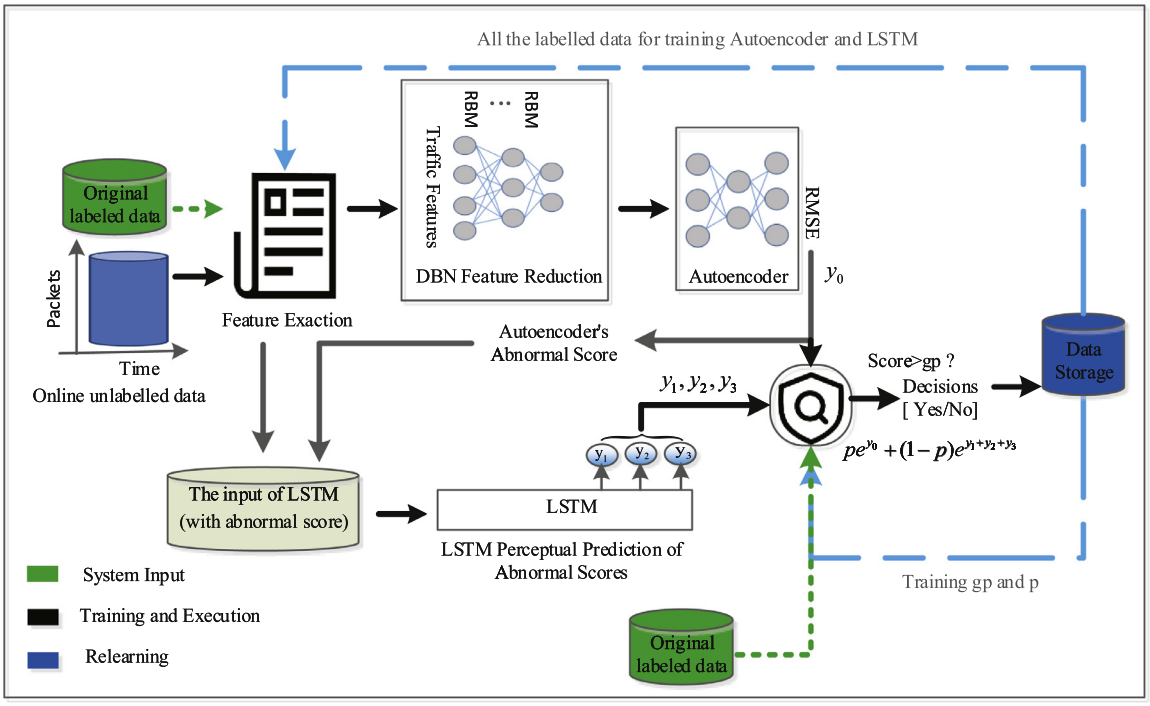


Fig. 1. The process diagram of HELAD model.

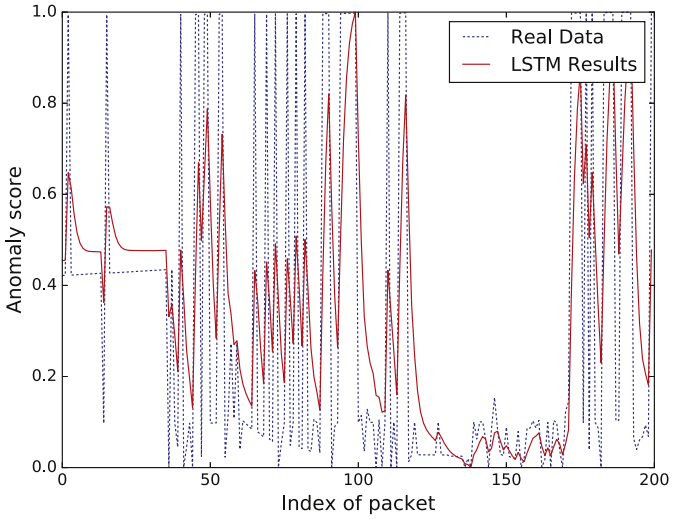


Fig. 2. Prediction effect of LSTM on mirai dataset.

ferent sessions are related; 2) there are many sessions at the same time; 3) the rate of packet arrival is high. In order to extract features efficiently, we decide to use the improved Damped Incremental Statistics algorithm [27].

For example, we analyze a TCP SYN packet in a network that does not adopt the SYN Cookies [58] defense mechanism. This can be the network packet generated by the client when it normally accesses the server. This packet can also be one of millions of attack packets that cause DoS attack. It depends on the context of the information. Furthermore, for IP-based video streaming, although the packets are normal, if there is significant jitter in the traffic, then there may be a man-in-the-middle attack. If the feature can capture this jitter, then this attack can be detected. We need contextual information to make the decision and hope that our features can be incorporated into these metrics that reflect anomalies in traffic. The Damped Incremental Statistics algorithm can meet these conditions, which is why we choose it as feature extraction method.

The Damped Incremental Statistics algorithm can consider the relationship between network traffic packets, but cannot consider the relationship between consecutive attacks. Thus, we discuss the second question. Is there a way to detect the relationship between successive attacks. Based on the previous research [11,12], we found that an effective combination of Autoencoder and LSTM can achieve this idea. The specific analysis is as follows. First, if the anomaly score calculated by Autoencoder alone is directly classified, it is equivalent to completely ignoring the relationship of attack behavior between consecutive network traffic packet. This is a disadvantage of Autoencoder. Second, if a single LSTM network is used to train an anomaly detector, this network does not effectively use Autoencoder to characterize anomalous features and abandon the benefits of ensemble learning. However, the unsupervised nature of Autoencoder can improve the context-independent detection. Third, LSTM has a recording function for historical samples. As the labeled data set becomes larger, the prediction effect of LSTM will be better and better. Therefore, combining the LSTM and Autoencoder networks to train the threshold of abnormal scores is better than a single neural network.

We use the mirai[1] dataset in Kitsune [27] to explore the effects of LSTM predictions. Fig. 2 shows prediction effect of LSTM on mirai dataset. The x-axis represents the index of packet. The y-axis represents the value of the anomaly score. The solid line in dark green in the figure is the predicted value of LSTM, and the dashed line in blue is the value of the actual root mean square error (RMSE). The trend predicted by LSTM is similar to the actual value.

[1] https://github.com/ymirsky/KitNET-py/blob/master/dataset.zip.

Table 1
Element definitions of HELAD model.

Expression	Meaning
S_i	S_i represents a sequence, the elements of this sequence can be the size of the packet, the jitter value, etc.
$d_\lambda(t)$	decay function, $\lambda > 0$ is the decay factor, and t is the time elapsed since the last observation from stream S_i
Statistics ST	Statistical characteristics of attributes of network traffic $ST = \{A_1, A_2, A_3, A_4\}$
Aggregated AG	The way network traffic is aggregated $AG = \{B_1, B_2, B_3, B_4\}$
Operation OP	The operation of the formula on the specified property $OP = \{op_i \in OP, 1 \le i \le 4 \mid < A_i, B_i > \}$
op_1	op_1 represents the bandwidth of the outbound traffic, and statistical feature st_1 operates on the aggregation ag_1.
op_2	op_2 represents Outbound and inbound traffic bandwidth, and statistical feature st_2 operates on the aggregation ag_2.
op_3	op_3 represents Outbound traffic packet rate, and statistical feature st_3 operates on the aggregation ag_3.
op_4	op_4 represents Inter-packet delay for outbound traffic, and statistical feature st_4 operates on the aggregation ag_4.
A_1	$\{\mu_i, \sigma_i\}$
A_2	$\{\|s_i, s_j\|, Rs_i s_j, Cov_{s_i s_j}, ps_i s_j\}$
A_3	$\{w_i\}$
A_4	$\{w_i, \mu_i, \sigma_i\}$
B_1	{*SrcIP, Channel, Socket*}
B_2	{*Channel, Socket*}
B_3	{*SrcIP, Channel, Socket*}
B_4	{*Channel*}
λ^*	$\{\lambda_1, \lambda_2, \cdots, \lambda_{n_1}\}$, λ_i represents the i^{th} decay factor.
f_i	$f_i = < \lambda_i \in \lambda^*, st \in A_i\ ag \in B_i \mid t_j, st_k, ag_k >$, The original i^{th} feature, the feature is expressed as a triple.
f_1	$f_1 = < \lambda_1, \mu_i, SrcIP >$ Statistic μ_i operate on attribute SrcIP as a feature.
$\vec{x_0}$	$\vec{x_0} = (f_1, f_2, \cdots, f_{n_0})^T$, Original feature vector
$\vec{x_R}$	$\vec{x_R} = (f_1, f_2, \cdots, f_n)^T$ Feature vector after dimensionality reduction.
$\vec{x_L}$	$\vec{x_L} = (f_1, f_2, \cdots, f_{n_0}, RMSE)^T$ Feature vector for training LSTM.
RMSE	Root mean square error, Method for calculating abnormal scores.
p	Coordinate the weight between the predicted and detected values.
TL	The RMSE output by Autoencoder serves as temporary training label for the LSTM.
gp	Dynamically changing anomaly detection threshold
K	Environment timer

3.2. Feature extraction

We take the Damped Incremental Statistics for feature extraction [27]. A slight difference from the approach of Kitsune is that we remove the SrcMAC-IP field and we change the number of features per sample (packet). The element definitions of HELAD model are listed in Table 1.

The Damped Incremental Statistics algorithm treats one of the attributes of each packet as an unbounded stream. These attributes can be the count, size, or jitter (inter-packet delay) of the corresponding packets aggregated by the same SrcIP (source IP), Channel (source and destination IPs) and Socket (source and destination IPs and ports). For a packet attribute stream S, the mean, variance, and standard deviation of S can be updated by the tuple of $IS = (N, LS, SS)$. N represents the total number of network packets arriving, LS is linear sum, and SS is squared sum. Each time a new network traffic packet p_i arrives, feature extraction can be performed according to the feature extraction formula in the table I, and the tuple of IS can be updated by $IS \leftarrow (N+1, LS + x_i, SS + x_i^2)$.

Table 2
Incremental statistics index for $IS_{i,\lambda}$ in [27].

Statistic	Notation	Calculation
Weight	w	w
Mean	μ_{s_i}	LS/W
Standard deviation	σ_{s_i}	$\sqrt{\lvert SS/W - (LS/W)^2 \rvert}$
Magnitude	$\lVert S_i, S_j \rVert$	$\sqrt{{\mu_{s_i}}^2 + {\mu_{s_j}}^2}$
Radius	$R_{S_iS_j}$	$\sqrt{(\sigma_{s_i})^2 + (\sigma_{s_j})^2}$
Covariance	$Cov_{S_iS_j}$	$\frac{SR_{ij}}{W_i+W_j}$
Correlation Coefficient	$P_{S_iS_j}$	$\frac{Cov_{S_iS_j}}{\sigma_{s_i}\sigma_{s_j}}$

x_i is the statistic of p_i. The statistics at any time are $\mu_s = \frac{LS}{N}, {\sigma_s}^2 = \lvert \frac{SS}{N} - (\frac{LS}{N})^2 \rvert$, and $\sigma_s = \sqrt{{\sigma_s}^2}$.

In the case of using a sliding window, the memory and runtime complexity required for the *IS* method is $O(n)$. For reducing the complexity to O(1), $IS_{i,\lambda}$ data structure is used to maintain the latest snapshot of the feature. The main idea of the Damped Incremental Statistics algorithm is that the weight of a sample decreases with time. The decay model for this weight uses the formula: $d_\lambda(t) = 2^{-\lambda t}$, where $\lambda > 0$ is the decay factor, and the t is the timestamp difference between the current packet and the previous network packet.

Specifically, $IS_{i,\lambda} = (w, LS, SS, SR_{ij}, t_{last})$ is maintained in real time. w is the current weight and one of the one-dimensional feature calculation methods, which is calculated by first adding one and then multiplying by decay value. The first three lines of Table 2 are used to calculate one-dimensional features. t_{last} is the timestamp of the previous packet. SR_{ij} is the sum of residual products between two attribute streams, which is used to calculate two-dimensional features. $r_i r_j$ represents one of the two-dimensional feature calculation methods. The methods in the last four rows of Table 2 are used to calculate two-dimensional features. For updating the $IS_{i,\lambda}$ in real time, Damped Incremental Statistics has the following update steps:

Step 1: Calculate the decay factor, $\phi \leftarrow d_\lambda(t_{cur} - t_{last})$;

Step 2: Calculate $IS_{i,\lambda} \leftarrow (\phi w, \phi LS, \phi SS, \phi SR, t_{cur})$ and integrate newly arrived packet into data structures $IS_{i,\lambda} \leftarrow (w+1, LS + x_{cur}, SS + {x_i}^2, SR_{ij} + r_i r_j, t_{cur})$;

Step 3: $t_{last} = t_{cur}$, goto Step 1.

Next we introduce the formation of features. For instance, op_1 represents statistical feature st_1 operates on the aggregation ag_1 in the case of the bandwidth of the outbound traffic. st_1 is from A_1, where A_1 represents the statistical method. ag_1 is from B_1, where B_1 represents what the packets are aggregated by. Therefore, the first feature can be represented as $f_1 = < \lambda_1, \mu_i, SrcIP >$. This means that in the case of first decay factor λ_1, statistic μ_i operates on attribute SrcIP as a feature.

After selecting the original features $\vec{x_0} = (f_1, f_2, \cdots, f_{n_0})^T$, we introduce the expression of each sub-model in the HELAD algorithm. The HELAD algorithm has four parts: feature dimension reduction, abnormal score generation, abnormal score prediction, and abnormal detection result combination.

3.3. Feature dimension reduction

The first is the HELAD-DBN dimension reduction submodel. There are two reasons why DBN technology is applied to our model. First, the DBN has the ability to find good features in incomplete information. Second, the DBN can achieve dimensionality reduction, so that the efficiency of training Autoencoder is improved. This submodel is essentially a multi-layer HELAD-RBM [38] model. Therefore, the focus of our discussion is the training and solution of the original feature vector $\vec{x_0}$ in the HELAD-RBM model.

The elements of the HELAD-RBM model are defined as follows. n_0, m represent the number of neurons in the visible layer and the hidden layer, respectively. $\vec{x_0} = (f_1, f_2, \cdots, f_{n_0})^T$ is the state vector of the visible layer, and $\vec{h_0} = (h_1, h_2, \cdots, h_m)^T$ is the state vector of the hidden layer. The vectors $\vec{a} = (a_1, a_2, \cdots, a_{n_0})^T$, $\vec{b} = (b_1, b_2, \cdots, b_{n_0})^T$ represent the offset vectors of the visible and hidden layers, respectively. The RBM model is an energy-based model. For a given set of states $(\vec{x_0}, \vec{h_0})$, the system energy of HELAD-RBM can be defined as:

$$\mathrm{E}\left(\vec{x_0}, \vec{h_0} \middle| \vec{\theta}\right) = -\sum_{i=1}^{n} a_i f_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} f_i w_{ij} h_j \tag{1}$$

In the formula, $\vec{\theta}$ represents the parameter set of HELAD-RBM. w_{ij} is the connection weight between the i^{th} neuron of the visible layer and the j^{th} neuron of the hidden layer, b_j represents the offset of the j^{th} neuron of the hidden layer, and a_i represents the offset of the i^{th} neuron of the visible layer.

To solve this model, we can use the contrast divergence algorithm proposed by [39]. The specific DBN dimension reduction solution can be referred to the literature [40,41]. After HELAD-DBN dimension reduction we get a new feature vector $\vec{x_R} = (f_1, f_2, \cdots, f_n)^T$.

3.4. Abnormal score generation

The second sub-model is the HELAD-Autoencoder anomaly score calculation model. This submodel calculates the anomaly score by the value of the loss function. Autoencoder [42] tries to learn the function $S(x)$, which satisfies: $S_{w,w',b_1,b_2}(\vec{x_R}) = \vec{x_R}$. Where w, w', b_1, and b_2 are model parameters. $S(x)$ can be represented in two phases. The first is the coding phase from the input layer to the hidden layer. The second is the decoding phase from the hidden layer to the output layer.

$$h = f\left(w\vec{x_R} + b_1\right) \tag{2}$$

$$\vec{y_R} = g\left(w'h + b_2\right) \tag{3}$$

The learning goal of this network is $\vec{y_R} \approx \vec{x_R}$. In order to reconstruct the input $\vec{x_R}$ as much as possible, the root mean square error (RMSE) is used as the loss function: $RMSE(\vec{x_R}, \vec{y_R}) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}}$. Autoencoder is a special artificial neural network (ANN). Commonly used way for training an ANN is known as the back-propagation algorithm [43] and the activation function we use is sigmoid.

3.5. Abnormal score prediction

The third sub-model is the HELAD-LSTM anomaly score prediction model. This sub-model predicts the anomaly scores of the next three samples based on the anomaly scores of the historical samples. Inspired by [44], we decide to use LSTM to perform timing prediction on anomalies. $\vec{x_L} = (f_1, f_2, \cdots, f_{n_0}, RMSE)^T$ is the original feature vector plus the RMSE label (TL). This is a new feature vector for training HELAD-LSTM which can predict abnormal scores.

At training time t, the parameters of HELAD-LSTM are updated as follows:

$${b_i}^t = f\left(w_{xi}\vec{x_L}^{(t)} + w_{hi}{b_h}^{t-1} + w_{ci}{b_c}^{t-1} + \sigma_i\right) \tag{4}$$

$${b_f}^t = f\left(w_{xf}\vec{x_L}^{(t)} + w_{hf}{b_h}^{t-1} + w_{cf}{b_c}^{t-1} + \sigma_f\right) \tag{5}$$

$${b_c}^t = {b_f}^t \times {b_c}^{t-1} + {b_i}^t \times f\left(w_{xc}\vec{x_L}^{(t)} + w_{hc}{b_c}^{t-1} + \sigma_c\right) \tag{6}$$

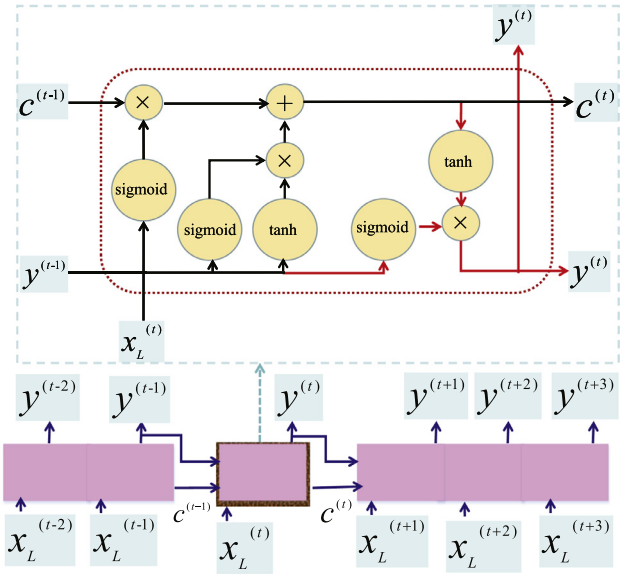


Fig. 3. Relationship between internal variables of LSTM unit.

$$b_o{}^t = f\left(w_{xo}\overrightarrow{x_L}^{(t)} + w_{ho}b_h{}^{t-1} + w_{co}b_c{}^{t-1} + \sigma_o\right) \tag{7}$$

$$b_h{}^t = b_o{}^t \times f\left(b_c{}^t\right) \tag{8}$$

$$a_k{}^t = w_k b_h{}^t \tag{9}$$

Since LSTM is a time series model, $\overrightarrow{x_L}^{(t)}$ in the formula is the representation of $\overrightarrow{x_L}$ at time t. $b_i{}^t$, $b_f{}^t$, $b_c{}^t$, $b_o{}^t$ represent the input value of the input gate, the forgetting gate, the memory state, and the output gate respectively at time t; $a_k{}^t$ represents the output value of the network output layer at time t, which is the value of $y^{(t)}$. We can calculate the values involved in the above formula (4)-(9) in turn. Fig. 3 indicates the relationship between internal variables of LSTM unit. Finally, we will get the predicted RMSE values $y^{(t+1)}, y^{(t+2)}, y^{(t+3)}$ for the three moments $t+1$, $t+2$, and $t+3$, and each prediction yields a predicted value. The specific LSTM network solving algorithm can be found in the literature [45].

3.6. Abnormal detection result combination

In the final step of the model, we use the discriminant formula $pe^{y_0} + (1-p)e^{y_1+y_2+y_3}$ to calculate the anomaly detection score for each packet. We use the simulated annealing algorithm to optimally select the gp and p values. Specifically, it is shown in Algorithm 1. If the value of the anomaly detection score exceeds the threshold gp, this packet will be recognized as an abnormality. The specific details are shown in Algorithm 2.

3.7. Analysis of heterogeneous ensemble of HELAD model

Deep learning method has not been used as component learners of heterogeneous ensemble learning in network intrusion detection. In addition, HELAD model does not have a more common ensemble learning mechanism like stacking, voting, etc. We mainly analyse our heterogeneous model from the perspective of the ensemble strategy and base classifier selection based on reference [61].

About ensemble strategy, ensemble learning models can be defined as models using multiple learners to do classification, and there is no need to use specific ensemble methods. So although we don't use classic ensemble methods such as stacking, our model is still an ensemble learner. Our discriminant formula is actually a variant of weighted averaging method defined as combination methods in [61], which uses the weighted average of the Autoencoder and the LSTM outputs as the anomaly score. In terms of base classifier selection, we use Autoencoder and LSTM as component learners, and the advantage of HELAD model is that both base classifiers are deep learning methods.

In summary, we propose a new heterogeneous ensemble learning approach which is different from classic ensemble methods such as stacking, where we use the Autoencoder to model normal traffic profile and generate the label to train the LSTM, and use heuristic algorithm to do weighted averaging with results of the Autoencoder and LSTM in the final stage. This is essentially the voting method, which is a method of heterogeneous ensemble learning.

Algorithm 1 Searching p and gp by simulated annealing algorithm.

Input:
Data; //Testing data
HELAD; //The parameters are constant
$\{T_0, r, T_f, L\}$; //The hyper-parameter set. T_0 is the initial value of temperature. r is the coefficient for reducing temperature. T_f is the lowest temperature. L is the amount of searching for every value of temperature.

Output:
p;//Coordinate the weight between the predicted and detected values.
gp;//Overall threshold, because it is an exponential function, only the upper limit

```
//Initial phase
T = T_0;
p_0 = rand();//The function, rand(), returns a random floating number in (0, 1)
gp_0 = rand();
x=(p_0, gp_0);
f = F1 − score computed with Data, HELAD, x;
//F1-score is the metric (mentioned in Section V-A)
//Searching phase
while T > T_f do
  for k = 0 to L − 1 do
    x' = x + (rand() − 0.5, rand() − 0.5) × 0.5^k ;
    //the formula makes the whole solution domain is involved
    if x'[0] < 0 or x'[0] > 1 or x'[1] < 0 or x'[1] > 1 then
      continue;
    end if
    f' = F1 − score computed with Data, HELAD, x';
    if f' > f or rand() < e^{−(||f'−f||)/T} then
      x = x';
      f = f';
    end if
  end for
  T = r × T;
end while
//x[0] represent the optimal value of p
//x[1] represent the optimal value of gp
```

4. Model training & strategy optimization

The previous section mainly introduces our model from the perspective of technology construction. This section is now elaborated from the perspective of specific training details and the optimiza-

Algorithm 2 HELAD model anomaly detection.

Input:

pss_i;//Represents the i-th arriving packet.
pp_1;//Indicates the first network packet required for Autoencoder to generate abnormal score
tt_1;//Indicates the time required for Autoencoder to generate abnormal score
tt_2;//Indicates the time of LSTM training
tt_3;//Indicates the parameter adjustment time

Output:

Abnormal detection result $Y_{decision}$
Training time of the HELAD algorithm t_{final}
//Begin procedure
//1)Begin Autoencoder training and generate abnormal score
//Train Autoencoder with labeled data
$PP = pp_1$;//PP adds one at a time to indicate a new packet arrives.
$t = 0$;//t is used to record timing relationships
for $i = 1$ to PP **do**
  $\overrightarrow{x_0}$=Feature_Extraction(pps_i);//Extracting feature of a sample using the Damped IncrementalStatistics algorithm;
  $\overrightarrow{x_R}$=DBN($\overrightarrow{x_0}$);
  $\overrightarrow{y_R}$=Autoencoder($\overrightarrow{x_R}$);
  $TL = RMSE(\overrightarrow{x_R}, \overrightarrow{y_R}) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}}$;
end for
$t = tt_1$;
//2)Begin LSTM training
//Adding TL as a feature to feature vector $\overrightarrow{x_0}$
//Forms a new feature vector $\overrightarrow{x_L}$
//Train the LSTM network with $\overrightarrow{x_L}$
LSTM network construction;
LSTM network training;
$t = t + tt_2$;
//3)Initial parameter training
//Use expert-labeled data with abnormal score
Use Algorithm 1 to derive the values of gp and p;
$t = t + tt_3$, $t_{final} = t$;
//4)Begin online monitoring and data storage
//PPO represents the arriving packet after time t
while PPO++ **do**
  Autoencoder(PPO)=y_0;
  LSTM (PPO-2)=y_1;
  LSTM (PPO-1)=y_2;
  LSTM (PPO)=y_3;
  $dv = pe^{y_0} + (1-p)e^{y_1+y_2+y_3}$;
  if $dv > gp$ **then**
    $Y_{decision}$=abnormal;
  else
    $Y_{decision}$=normal;
  end if
end while

tion of the effects of the entire model. The training of the model is to obtain the values of the parameters gp and p in the discriminant function after the neural network is stable. The values of gp and p are detailed in Table 2. All specific training steps can be seen in Fig. 1. We divide the overall training model into three stages as follows.

4.1. Initialization

The original small amount of expert annotated (abnormal or normal) sample dataset is acquired by Damped Incremental Statistics algorithm to obtain a high-dimensional feature vector $\overrightarrow{x_0}$. Then, the feature reduction is performed using the DBN to obtain the feature vector $\overrightarrow{x_R}$. Next, the feature vector $\overrightarrow{x_R}$ is taken as the input of the Autoencoder, and then the training of the Autoencoder is completed. During training, the root mean square error (RMSE) of each sample reconstruction is accurately recorded. This RMSE is used as an anomaly detection score. Next, the RMSE of each sample is added as a feature to the feature vector $\overrightarrow{x_0}$ to form a new feature vector $\overrightarrow{x_L}$. then, The feature vector $\overrightarrow{x_L}$ is trained as an input to the LSTM network. After the Autoencoder network and LSTM network training is completed, the data set marked by the expert is used again. This time it is used to train the values of gp and p. Specifically, each sample collected will have an output y_0 through the Autoencoder, and then output y_1, y_2, y_3 through the LSTM. We establish the discriminant formula $pe^{y_0} + (1-p)e^{y_1+y_2+y_3}$ and then use the label (abnormal or normal) of each sample for parameter tuning. In this way, we can initialize the values of gp and p of the HELAD algorithm (the case where the detection rate is the highest) by the label marked by the expert.

4.2. Real time traffic detection

The HELAD anomaly detector has been initialized in the above stage. Next, the HELAD model can begin to receive actual network traffic (as indicated by the red arrow). Network traffic continues through the Autoencoder network and the LSTM network, and y_0, y_1, y_2, y_3 are obtained for each sample taken online.

Then we can get the value dv of the formula $pe^{y_0} + (1-p)e^{y_1+y_2+y_3}$. If dv is greater than gp, then decision is yes, which is an abnormal. On the contrary, it is normal. In this way, online traffic data is continuously labeled and stored.

4.3. Relearning HELAD model

At this stage, we can set an environment timer K that the user can adjust (in the time of this K, the expert can modify the erroneous label in the log according to experience, if time and effort allow). When the cumulative time of K reaches a certain value, we merge the original expert data with the tagged data in the log. We then re-trained the HELAD model by replacing the expert annotation data for the first phase with this new data set. It is important to note that the anomaly detection detector formed by the last gp and p is used for detection before the next retraining is completed. The reason for this is that the training time and the detection time are parallel, and there is no waiting time for training. The specific details are shown in Algorithm 3 .

5. Experiments & evaluation

This section covers our experimental results. Our codes are available at the open-source code repository.[2] In order to systematically evaluate our model, we want to check the following four points: (1) In what circumstances can our algorithm achieve the best performance. (2) Whether ensemble learning and the relearning function are effective. (3) How does our algorithm compare to the performance of other state of the arts algorithms on different data sets. (4) How does our algorithm compare to the state-of-the-art homogeneous ensemble learning algorithms.

5.1. Metrics for evaluating anomaly detection algorithm

Time consuming is one of the shortcomings of machine learning. Parallel computing, Graphics Processing Unit and other acceleration methods will be used in our future work. Therefore, the level we discuss is the effect of anomaly detection in the latest

[2] https://github.com/cdogemaru/CPIP.

Algorithm 3 HELAD strategy optimization algorithm.

Input:
$Data_0$ //Data set marked by experts before the training time point
$Data_{10}$ //Labeled data set by last round of HELAD algorithm detecting
$Data_{20}$ //Labeled data set by this round of HELAD algorithm detecting
$Data_2$ //Data set marked by experts during this round of detection
K //User time timer
SW // Anomaly detector switch (boolean variable)
Output:
gp, p, and new trained Autoencoder and LSTM neural networks for each K time period
//Begin procedure

```
t=0, SW = ture;
T = K;//The value of K is greater than the training time of the HELAD algorithm.
flag=true;//If it is the first time, the model will use 2T time.
while SW do
  if flag then
    while t < T do
      t++;
      Store(HELAD(pss_i)) to Data_10;
    end while
    t=0;
    while t < T do
      t++;
      Store(HELAD(pss_i)) to Data_20;
      Data = Data_10 + Data_2 + Data_0;
      Use Data to replace the training data in Algorithm 2;
      Updates gp and p;
    end while
    Data_10 +=Data_20;
    t=0;
    flag=false;
  else
    while t < T do
      t++;
      Store(HELAD(pss_i)) to Data_20;
      Data = Data_10 + Data_2 + Data_0;
      Use Data to replace the training data in Algorithm 2;
      Updates gp and p;
    end while
    Data_10 +=Data_20;
    t=0;
  end if
end while
```

real network traffic and comparison with other machine learning algorithms.

The effectiveness of Machine Learning based anomaly detection algorithm can be evaluated by the following indicators: Precision (P): $\frac{TP}{TP+FP}$, Recall (R): $\frac{TP}{TP+FN}$, F1-Score (F1): $\frac{2\times P\times R}{P+R}$, False Positive Rate (FPR): $\frac{FP}{FP+TN}$, Area Under Curve (AUC): the area enclosed by the ROC curve and the coordinate axis. Among them, True Negative (TN): a measure of the number of normal events rightly classified as normal ones. True Positive (TP): a measure of the number of abnormal events rightly classified as abnormal ones. False Positive (FP): a measure of normal events misclassified as attacks. False Negative (FN): a measure of attacks misclassified as normal. We use these five indicators to measure the performance of our HELAD algorithm.

5.2. Datasets and experimental settings

5.2.1. Datasets

Two data sets, MAWILab[3] and IDS 2017,[4] are used in this paper. The experiments are all based on the MAWILab dataset. How ever, the IDS 2017 dataset will be tested in contrast experiment, which means comparison with other algorithms.

MAWILab [46,47] is a database that assists researchers to evaluate their traffic anomaly detection methods. MAWILab annotates traffic anomalies in the MAWI archive with four different labels: anomalous, suspicious, notice, and benign. In order to use these anomaly detection labels marked by different anomaly detector. We classify the two categories anomalous and suspicious as abnormal and classify benign and notice into normal. Table 3 are examples of the anomalies we use in our experiments. The first line identifies that *network_scan_SYN* is detected by the anomaly detectors. This is a SYN attack, and the original label is anomalous. Then, the label we give is an abnormal. The sixth line identifies that *network_scan_ICMP_ecrq* is detected by the anomaly detector Hough. This is a ping flood, and the original label is suspicious. The label we give is abnormal.

The IDS2017 data set collection time is from July 3, 2017 to July 7, 2017. The data set contains benign traffic and some of the latest common attacks. The format of the data set includes real data (pcap file) and the results of analysis using CICFlowMeter. Generating real attack scenes is the main task of this data set. This data set includes attacks such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. We use the original packet and its corresponding label to our experiment.

5.2.2. Experiment environment

In order to use a variety of algorithms more effectively, we use Python to implement our model. The hardware and software configurations are shown in Table 4.

5.3. Experimental results & implications

This section illustrates a group of experiments to verify the effectiveness of HELAD Model. We study the proposed anomaly detection using MAWILab data for the period of June 3, 2018. Table 5 shows the distribution of the data set. Here, we have a total of 300,000 network traffic packets. All of these packets take about 15 min to collect. Among them, 125,374 network traffic packets are abnormal. Table 6 illustrates number of labels in training set (including verification set) and testing set. In order to accurately predict the future real-predictive environment, the predictor must retain data for events that occur after the event that fits the model [59]. Therefore, for time-series data such as network traffic, we do not use k-fold cross-validation, but use hold-out. The hold-out method directly divides the data set D into two mutually exclusive sets. One of the sets is used as the training set S, and the other is used as the test set T, ie D=S∪T, S∩T=0. After training the model on S, the test error is evaluated by T, and as evaluation of generalization error. To assess the performance accurately, we use three partition methods to train and evaluate the data set. Table 7 lists the comparison result of average P, R, F1 for three partition methods. Therefore, the latter experiments are based on partition 3 to train and evaluate the data set.

5.3.1. In what circumstances can our algorithm achieve the best performance

In order to confirm in what circumstances can our algorithm achieve the best performance, we analyze the influence of four fac-

[3] http://www.fukuda-lab.org/mawilab/index.html.
[4] https://www.unb.ca/cic/datasets/ids-2017.html.

Table 3
Example of the anomalies in June 3, 2018.

Taxonomy	Heuristic	Original Label	Detectors	Final Label
network_scan_SYN	SYN attack	anomalous	*Hough, PCA*	Abnormal
network_scan_SYN	SYN attack	anomalous	*Hough, Gamma, PCA*	Abnormal
network_scan_TCP_RST_ACK_response	RST attack	anomalous	*Hough, PCA*	Abnormal
small_network_scan_SYN	SYN attack	anomalous	Gamma	Abnormal
network_scan_SYN	SYN attack	anomalous	Hough, KL, PCA	Abnormal
network_scan_ICMP_ecrq	Ping flood	suspicious	Hough	Abnormal

Table 4
The software and hardware configurations.

Resource Type	Configuration
Software environment	ubuntu 14.04, conda 4.4.10, python 3.5.0, numpy 1.15.0, cython 0.28.5, scapy 2.4.0, scipy 1.1.0, keras 2.2.2, tensorflow 1.9.0
CPU	i7-5500 2.40 GHz
Memory	32G
Number of server	1

Table 5
Number of labels in the MAWILab data set.

DataSet	Abnormal	Normal
300,000	125,374	174,626

Table 6
Number of labels in training set and testing set (number/portion).

partition method	Tag	Training set	Testing set
partition 1	Normal	115,253	59,373
	Abnormal	82,747	42,627
	(Sum)	198,000/66%	102,000/33%
partition 2	Normal	130,969	43,657
	Abnormal	94,031	31,343
	(Sum)	225,000/75%	75,000/25%
partition 3	Normal	152,798	21,828
	Abnormal	109,702	15,672
	(Sum)	262,500/87.5%	37,500/12.5%

Table 7
Comparison of average P, R, F1 for different partition method (200 features, 300,000 packets, gp = 0.65, p = 0.8).

partition method	Average		
	Precision	Recall	F1-Measure
partition 1	0.793	0.837	0.815
partition 2	0.821	0.860	0.840
partition 3	0.901	0.880	0.891

tors (the dimension of the feature, the size of the data set, the value of *p*, and the anomaly detector threshold *gp*) on the performance of the HELAD algorithm.

In the case of selecting 100 features:

Damped Incremental Statistics algorithm is used to sample the features. We set the value of λ to [5,3,1,0.1,0.01] to get the 100-dimensional feature. After using DBN, our features are reduced to 30 dimensions. We use Algorithm 1 to get the optimal values for *gp* and *p*. For further detailed analysis, we use the growth step size method to observe the effect of *p* and *gp* on the anomaly detection effect. In the iteration we found that *p* is 0.8 or *gp* is 0.65, and the abnormality detection result is better. The method of controlling variables is used to determine the values of *gp* and *p*. As shown in

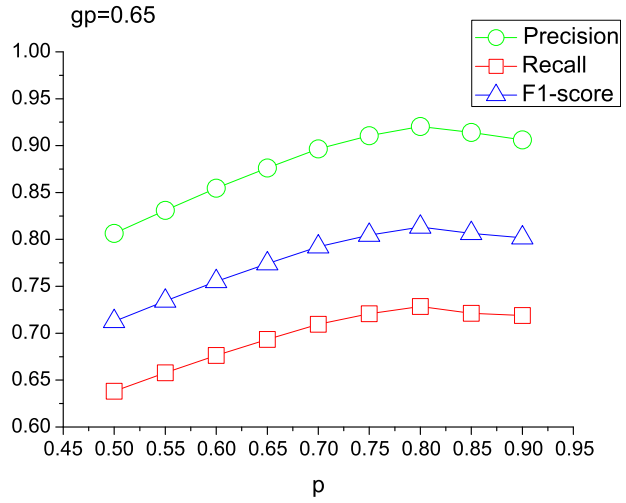


Fig. 4. Performance of HELAD model under the conditions of 100 features, 300,000 packets. When the gp value is fixed, observe the effect of p value on the detection effect.

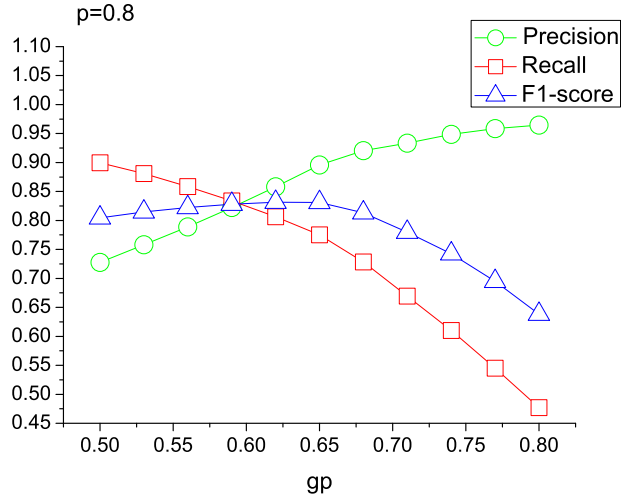


Fig. 5. Performance of HELAD model under the conditions of 100 features, 300,000 packets. When the p value is fixed, observe the effect of gp value on the detection effect.

Fig. 4, we use 300,000 packets and set the value of *gp* to 0.65 to observe the effect of p-value on the detection effect in the HELAD algorithm. It can be seen that the Precision, Recall, and F1 value all reach the highest when the *p* value is 0.8.

In order to determine the effect of the value of *gp* on the detection effect in the HELAD algorithm, we conduct the further experiment. Fig. 5 shows that the F1 value reaches the highest when the *gp* value is 0.65.

After determining the values of *gp* and *p*, we verify the effect of the data set size on the detection of the HELAD algorithm. As can be seen from Table 8, when the over all data set size is 300,000,

Table 8
Evaluation indicators under different dataset sizes (100 features, gp = 0.65, p = 0.8).

Data set size	Precision	Recall	F1-score
30000	0.761	0.761	0.757
100000	0.851	0.855	0.853
300000	0.871	0.864	0.861

the average Precision is 0.871, the average recall is 0.864, and the average of the F1 value is 0.861. When the data set is smaller, the detection effect is even worse.

In the case of selecting 200 features:

In order to verify the effect of the number of features on the detection of the HELAD algorithm, we set 200 features for each sample, and the value of λ is [10,5,3,2,1.5,1,0.5,0.2,0.1,0.01]. After using DBN, our features are reduced to 50 dimensions.

When the feature is increased to 200 dimensions, the computing power of a single node is limited. So we selected 10,000, 30,000, and 100,000 data packets for analysis. Fig. 6 shows the performance of HELAD anomaly detection model under the conditions of 200 features with the varying p-value. As shown in Fig. 6, from the comparison of each subgraph, as the data set expands, the effect of anomaly detection is getting better and better. This is because anomaly detection algorithms are based on machine learning algorithms, which depend on the data set. As shown in Fig. 6(a), when the data packet is 10,000, the value of the Precision is 0.7, and the *p* value has not yet played a role. As shown in Fig. 6(b), in the case where the data packet is 30,000, the Precision, Recall and F1 increase as the *p* value increases. This shows that the effects of Autoencoder come into play. So for the selection of p value, we use the evaluation index of F1 value. When the data packet shown in Fig. 6(c) is 100,000, the detection effect is best when the *p* value is 0.8. This illustrates that as the further expansion of the data set, the role of LSTM begins to emerge. It also reveals that LSTM is a deep learning technology that requires a larger data set if it needs to work better.

Fig. 7 reveals the performance of HELAD anomaly detection model under the conditions of 200 features with the varying *gp* value. As shown in Fig. 7, from the comparison of each subgraph, the Precision increases as the *gp* value increases. However, the Recall is reversed because there are many false negatives due to an increase in the threshold. So for the selection of *gp* value, we use the evaluation index of F1 value. When the value *gp* is 0.65, the comprehensive detection effect is the best. The larger the data set, the better the detection effect, which is characteristic of the machine learning algorithm itself. Table 9 shows the effect of different data sets on the experiment under the 200-dimensional feature.

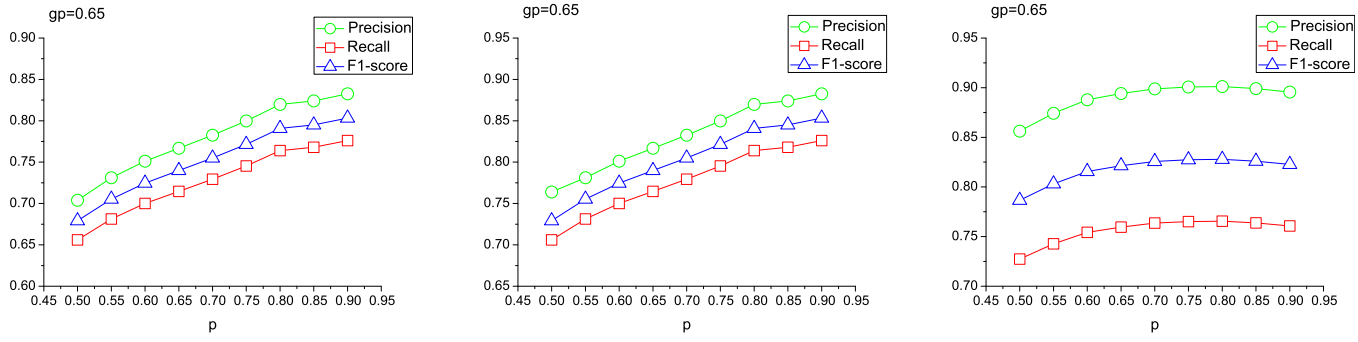
Based on the above experiments, we can draw the following conclusions. In the case of insufficient data (data size 100000), our model can improve the precision by increasing the feature dimension. In the case where the feature dimension is specified (100-dimensional feature), increasing the dataset size can improve the precision. We can see that the size of the data set has a greater impact than the feature. Therefore, if our model is to achieve good performance on the MAWILab dataset, the feature requires 200 dimensions, the dataset size is 300,000, the value of *gp* is 0.65, and the value of *p* is 0.8. If we change the network environment (dataset source), our model will also set the values of these variables according to the network environment to achieve the best detection effect.

5.3.2. Whether ensemble learning and the re-learning function are effective

In order to verify the effectiveness of ensemble learning. We use MAWILab's 300,000 data on June 1, 2018 for training and another 600,000 data for testing. Through training, we get three models: our model, our model without LSTM, and our model without RMSE. The experimental results are shown in Table 10. We can see that our ensemble model is better than a single model for anomaly detection.

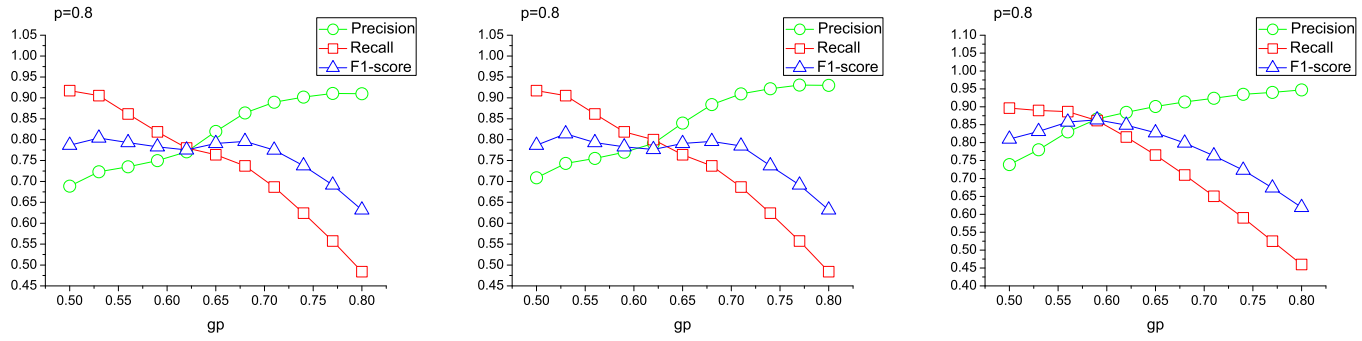
For verifying the importance of re-learning, we conduct two sets of experiments on MAWILab, using data from the first three days of April, June, July, and September, and intercepting 300,000 data per day. The first set of experiments is a function of re-learning. We use the data of the 1st of each month for training, and the data of the 2nd and 3rd of the month is used for testing. The second set of experiments do not have the function of re-learning, that is to say, the training is carried out with the data of April 1st, and the latter model is not re-trained, and the data of the 2nd and 3rd of the four months is used for testing.

In order to evaluate the detection effect of our HELAD model when the network flow characteristics change drastically, we add some analytical experiments. We analyze the experimental data for 6 days and 0402 stands for the date of collection of the MAWILab data set is April 2. For a better comparison, we count two sets of data. As shown in Table 11, the first group is the *#select* data set. This is our experimental data, which contains 300,000 network packets. The second group is the *#all* data set, which is a whole day of data. It is about 90 million packets a day, and the total amount of data is different every day. Previous represents the number of IP addresses that have appeared in the previous date. New expresses a completely new IP address. It is important to note that each IP address corresponds to multiple network packets. Table 11 shows changes in the IP address of data sets on different dates, indicating that the network IP is almost al-



(a) 10,000 packets (b) 30,000 packets (c) 100,000 packets

Fig. 6. Performance of HELAD anomaly detection model under the conditions of 200 features. When the gp value is fixed, observe the effect of p value on the detection effect.



(a) 10,000 packets (b) 30,000 packets (c) 100,000 packets

Fig. 7. Performance of HELAD anomaly detection model under the conditions of 200 features. When the p value is fixed, observe the effect of gp value on the detection effect.

Table 9
Evaluation indicators under different dataset sizes (200 features, gp = 0.65, p = 0.8).

Data set size	Precision	Recall	F1-score
30000	0.781	0.779	0.779
100000	0.873	0.879	0.875
300000	0.901	0.880	0.891

Table 10
Evaluation indicators for ensemble learning under MAWILab dataset (200 features, 300,000 packets, gp = 0.65, p = 0.8).

Method	Precision	Recall	F1-score
RMSE	0.812	0.772	0.791
LSTM	0.782	0.663	0.717
RMSE+LSTM	0.901	0.880	0.891

Table 11
Changes in the IP address of datasets on different dates.

Date	Previous(#select)	New(#select)	Previous(#all)	New(#all)
0403	3	66087	105,414	14,703,225
0702	6	59543	214,780	15,492,070
0703	124	59524	411,137	15,669,049
0902	10	118285	474,783	15,788,790
0903	16	94018	601222	15,575,598

ways changing. Fig. 8 illustrates that traffic volumes vary widely from day to day (during the specified time period). The experimental results are shown in Fig. 9. The model with re-learning is better than the model with no re-learning. It also shows that our re-learning model performs well in the case of severe network turbulence. Table 12 shows that FPR does not change much before and after re-learning. This shows that FPR is mainly related to the algorithm itself.

Table 12
The FPR before and after relearning.

Date	Relearning-FPR	Non-relearning-FPR
0402	0.072	0.072
0403	0.100	0.100
0702	0.100	0.104
0703	0.135	0.118
0902	0.162	0.197
0903	0.176	0.181

5.3.3. How does our algorithm compare to the performance of other state-of-the-art algorithms on different data sets

we use Isolation Forests (IF) [48] and Gaussian Mixture Models (GMM) [49]. IF is an ensemble based method of outlier detection, and GMM is a statistical method based on the expectation maximization algorithm. Then we use support vector machine (SVM) from [50], sparse autoencoder finetuned neural network (SAE) from [51], restricted boltzmann machine fine-tuned neural network (RBM) from [52] and kitsune from [27].

We have compared the effectiveness of these methods by working out the Precision, Recall and F1-score of these models. We can see that the Precision of the SVM algorithm is only 0.598. The Precision of the RBM is slightly better and can reach 0.708. Then there are SAE and IF, and the detection results are very similar. The detection effect of GMM is also not ideal. Kitsune still perform well

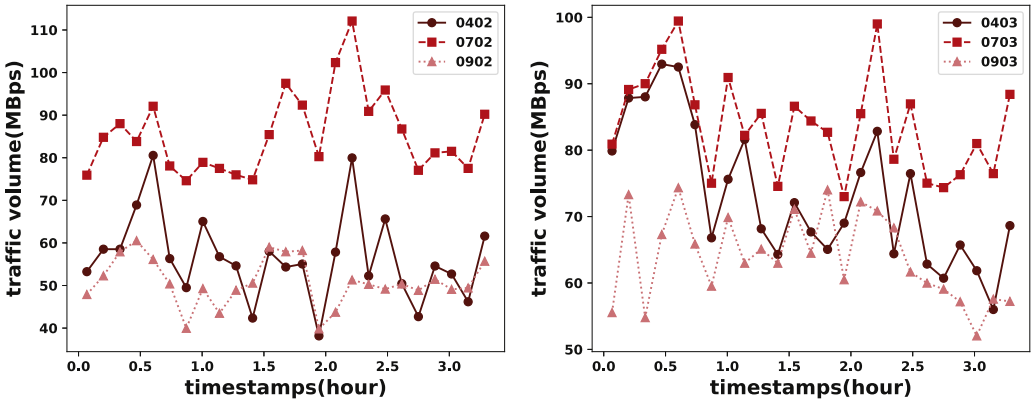
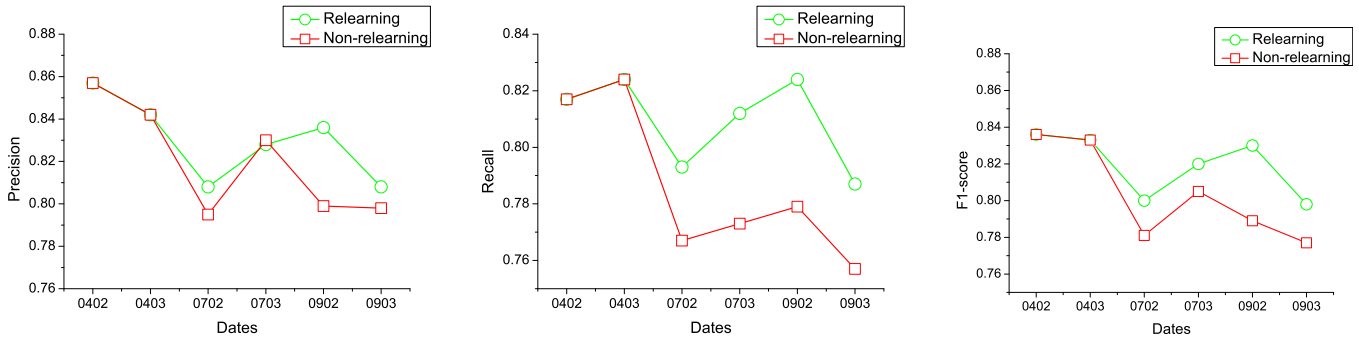


Fig. 8. The traffic volumes statistics corresponding to relearning experiment under MAWILab dataset.



(a) Precision (b) Recall (c) F1-score

Fig. 9. Evaluation indicators for relearning under MAWILab dataset (200 features, 300,000 packets, gp=0.65, p=0.8). This data set comes from different months. 0402 represents the detection effect on April 2.

Table 13
Evaluation indicators under MAWILab dataset (200 features, 300,000 packets, gp = 0.65, p = 0.8).

Compared methods	Precision	Recall	F1-score
SVM	0.598	0.651	0.623
RBM	0.708	0.714	0.711
SAE	0.759	0.711	0.734
IF	0.756	0.724	0.739
GMM	0.645	0.668	0.657
kitsune	0.871	0.870	0.870
HELAD	0.901	0.880	0.891

Table 14
Number of labels in the IDS 2017 data set.

DataSet	Abnormal	Normal
300,000	226,618	73,382

Table 15
Evaluation indicators under IDS 2017 dataset (200 features, 300,000 packets).

Compared methods	Precision	Recall	F1-score
SVM	0.797	0.444	0.768
RBM	1.000	0.433	0.604
SAE	1.000	0.961	0.980
IF	0.997	0.971	0.983
GMM	0.786	0.694	0.737
kitsune	0.940	0.998	0.968
HELAD (gp=0.65, p=0.80)	0.941	0.998	0.969
HELAD (gp=0.60, p=0.85)	0.988	0.982	0.985

in the 300,000 data packet, with a Precision of 0.871. Our HELAD algorithm has improved detection effect compared to the kitsune algorithm. The average performances of different methods are displayed in Table 13. As we can see from the results, our HELAD model achieves significantly better results compared with other machine learning based approaches.

To verify the robustness of the algorithm, we experiment on IDS 2017 data set. Our experiments on the IDS 2017 data set are based on partition 3. Table 14 shows the number of labels in the IDS 2017 data set. The experimental results are shown in Table 15. SVM's Precision and Recall are not high. The comprehensive evaluation index F1 value is only 0.768. The Precision of RBM can reach 1.000, but the corresponding Recall is very low. This leads to the fact that the comprehensive evaluation index F1 value is not as good as SVM. Both SAE and IF performed better in the IDS 2017 data set. This also shows that SAE and IF perform differently for different environments (data sets). Next is the GMM algorithm, and the overall performance is not good. The Kitsune algorithm still performs very well. Our HELAD algorithm uses a contrasting approach. The first group (gp=0.65, p=0.80) uses the gp value and the p value trained directly on the MAWILab data set, and the second group (gp=0.60, p=0.85) uses the newly trained gp value and p value according to the IDS 2017 data set. Experiments show that the comprehensive performance of the second group is better than other algorithms, and the comprehensive evaluation index F1 can reach 0.985. Comparative experiments verify the environmental adaptability of our algorithm.

5.3.4. How does our algorithm compare to the state-of-the-art homogeneous ensemble learning algorithms

Next, we compare our algorithm with the anomaly detection algorithm for homogeneous ensemble learning. GradientTreeBoost [56] integrates the gradient tree through the boost ensemble method. BaggingCR [57] integrates the conjunctive rule (CR) classifier through the bagging ensemble method. At the same time, we have added bagging integration of various basic classifiers SVM, k-NearestNeighbor (KNN), Multilayer Perceptron (MLP). Then, considering the advantages of Adaboost, we also add the Adaboost ensemble method using Logistic Regression as the base classifier. For the sake of fairness, we all use the same features for training. The experimental data set is derived from the reselected 300,000 data sets from IDS2017. This data set is called IDS2017-other. The purpose of re-selecting the data set is to make the experimental data more representative. Table 16 shows that our HELAD algorithm has higher F1 value and AUC, as well as lower FPR.

In order to measure the performance difference between the classification algorithms we use, Quade test and Quade post hoc test are used to weight how much our model and other state of the arts algorithms deviate from each other [62]. These tests can find differences between classifiers [63]. The $\widehat{F}$ value is an intermediate process for calculating the p value, and the larger the $\widehat{F}$ value, the smaller the p value. Based on the knowledge of hypothesis testing, we make two hypotheses HA and HB. HA means that there are no performance differences among the classifiers. HB indicates that there are performance differences among the classifiers. Table 17 reveals the result of Quade test with all classifiers, indicating that the performance of the classifiers is significantly different ($p < 0.05$) in terms of Precision, Recall, F1-score, FPR, AUC. If the result of Quade test is highly significant, the null hypothesis HA (the performance of all classifiers is similar) could be rejected and hypothesis HB should be accepted. Table 18 shows the results of Quade post hoc test in terms of AUC, indicating that the difference between BaggingCR and HELAD is most significant. Because the lower the p-value, the greater the relative significant. This significant suggests that if our model continues to ensemble

Table 16
Performance comparison between HELAD and other ensemble learning models.

Compared methods	Precision	Recall	F1-score	False Positive Rate	AUC
HELAD (gp=0.60, p=0.85)	0.9958	0.9958	0.9958	0.0215	0.9986
GradientTreeBoost	0.9573	0.9663	0.9618	0.0610	0.9657
BaggingCR	0.9340	0.9310	0.9320	0.0610	0.9370
AdaboostLogistic	0.9680	0.9199	0.9433	0.0432	0.9391
BaggingSVM	0.9646	0.9213	0.9425	0.0480	0.9591
BaggingKNN	0.9685	0.9495	0.9589	0.0438	0.9662
BaggingMLP	0.9649	0.9201	0.9419	0.0474	0.9539

Table 17
The result of Quade test with all ensemble learning models.

	Precision	Recall	F1-score	False Positive Rate	AUC
$\widehat{F}$	17.6667	5.3636	6.7778	7.7500	7.7500
p value	0.0014	0.0302	0.0174	0.0125	0.0125

Table 18
The p value of post hoc Quade test for AUC.

	HELAD	GradientTreeBoost	BaggingCR	AdaboostLogistic	BaggingSVM	BaggingKNN
GradientTreeBoost	0.2666	–	–	–	–	–
BaggingCR	0.0015	0.0052	–	–	–	–
AdaboostLogistic	0.0037	0.0151	0.3938	–	–	–
BaggingSVM	0.0104	0.0498	0.1158	0.3938	–	–
BaggingKNN	0.1767	0.7698	0.0073	0.0222	0.0758	–
BaggingMLP	0.0330	0.1767	0.0330	0.1158	0.3938	0.2666

BaggingCR, our model will perform better statistically. This provides an expandable space for our model.

6. Discussion

In this section, we will discuss some of the details of our models and experimental methods.

Q1: Whether e^{y_0} is needed in the final discriminant formula?

The first question, in HELAD model, the abnormal score calculated by Autoencoder and the original feature are stitched into a new vector x_L, which is used as an input to the LSTM. LSTM training uses the anomaly score and is affected by this abnormal score when forecasting. However, our discriminant formula also considers the abnormal score calculated by Autoencoder and the predicted value of LSTM. In other words, we want to determine whether e^{y_0} is needed in the final discriminant formula. We do the following analysis. We refer to the anomaly score as the LSTM input as the ASA (anomaly score A) and the ASB (anomaly score B) as the discriminant's anomaly score. The ASA is designed to train LSTM so that LSTM can predict abnormal scores based on past data. The ASB (that is y_0 in $pe^{y_0} + (1-p)e^{y_1+y_2+y_3}$) is to calculate the abnormal score of latest network packet and is one of the judgment conditions for ensemble learning. The functions of the two abnormal scores are different, so there is no redundancy. Table 19 shows the necessity of e^{y_0}, because our HELAD model works better with considering ASB.

Q2: Whether the hold-out algorithm we use is over-fitting, or is there a better evaluation method?

For second question, the hold-out data set partitioning method we use is consistent with the processing of time series data. Moreover, the same experimental method we use in all experiments is fair. To further demonstrate the effectiveness of our algorithm, we have added a set of experiments, which use Forward Chaining Cross-Validation [60] based on IDS2017-other. Forward Chaining Cross-Validation uses the front part of the data set as the training set and the latter part as the test set. And there are multiple split points, which can achieve the purpose of multiple training based on different partition. Finally, the errors on each partition are averaged to calculate a robust estimate of the model error. We define it as error-seeking method. We use the method which finds the average of the various indicators to replace the error-seeking method in the original paper. The two methods are equivalent. We set up three sets of data, and the number of training sets and test sets are: A (150,000 ‖150, 000), B(150, 000‖150, 000), C(450, 000‖150, 000). Then we average the experimental results of three groups. Table 20 shows that there is a small drop in Precision, Recall, F1-score, but our algorithm is still performing well. FPR is still the lowest.

Q3: Whether LSTM performs better in continuous attack detection while Autoencoder performs better in sudden attack?

For the third question, we assume that Autoencoder gains the profile of normal network traffic as one of the base learners, and provides learned RMSE as the label needed to train the LSTM, while LSTM works well for continuous attacks. In this section we prove this hypothesis with experimental results.

We re-extract 400,000 network packets from IDS2017 for experiments to analyze sudden attacks and continuous attacks. We are

Table 19
Whether e^{y_0} is needed in the final discriminant formula (200 features, gp = 0.65, p = 0.8, IDS 2017 dataset-other).

Method	Precision	Recall	F1-score
HELAD (with e^{y_0})	0.996	0.996	0.996
HELAD (without e^{y_0})	0.871	0.913	0.891

Table 20
Verify the problem of overfitting (200 features, gp = 0.65, p = 0.8, IDS 2017 dataset-other).

Compared methods	Precision	Recall	F1-score	FPR
HELAD	0.971	0.972	0.971	0.020
SVM	0.814	0.696	0.728	0.132
IF	0.907	0.610	0.678	0.234
RBM	0.874	0.875	0.869	0.411
SAE	0.987	0.887	0.933	0.091
GMM	0.800	0.682	0.709	0.154

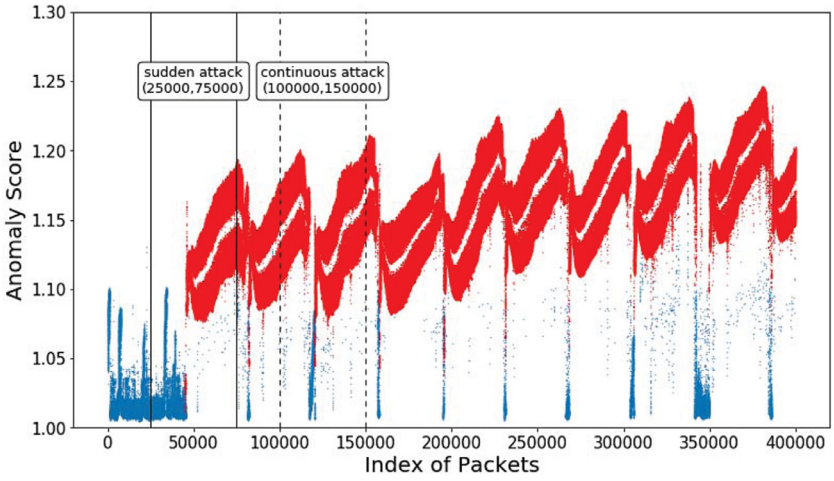


Fig. 10. Selection of corresponding areas of sudden attacks and continuous attacks.

Table 21
The experimental indicator of sudden attack (200 features, gp = 0.65, p = 0.8).

Compared methods	Precision	Recall	F1-score	FPR
LSTM	0.975	0.952	0.964	0.033
Autoencoder	0.982	0.982	0.982	0.025
LSTM+Autoencoder	0.986	0.993	0.989	0.019

Table 22
The experimental indicator of continuous attack (200 features, gp = 0.65, p = 0.8).

Compared methods	Precision	Recall	F1-score	FPR
Autoencoder	0.994	0.979	0.986	0.075
LSTM	0.996	0.996	0.996	0.043
LSTM+Autoencoder	0.998	0.997	0.998	0.020

using Wednesday's data, which contains different types of DoS attacks (DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye). The time period of the selected sudden attack is 25,000-75,000, that is, the period during which the number of attacks gradually increases. The duration of the continuous attack is 100,000-150,000, and the attack frequency is very high. The red dots in Fig. 10 represent abnormal packets, and the blue dots represent normal packets. Fig. 10 shows selection of corresponding areas of sudden attacks and continuous attacks. The X axis represents the network packet sequence number, and the Y axis represents the RMSE value of each network packet. The data from 25,000 to 75,000 between the two vertical solid lines represent sudden attacks. The data from 100,000 to 150,000 between the two vertical dotted lines represent continuous attacks. A comparison of Tables 21 and 22 shows that LSTM does perform better in continuous attack detection while Autoencoder performs better in sudden attack. Finally, our HELAD model works best and illustrates the need for our ensemble approach.

7. Conclusion

The network environment is increasingly complex, and as such, the form of attack is ever-changing. Many of the existing machine learning related anomaly detection models published previously are evaluated using data such as KDD, leading to the emergence of this problem that it is not practical in real-life environments. By introducing the idea of organic integration of various deep learning techniques, the HELAD model can better combine LSTM classifier and Autoencoder classifier. This provides a new idea for the application of heterogeneous ensemble learning in the field of anomaly detection. The advantage is to make it adaptable in the real environment. Then, latest raw packet data is used in our experiments, which provides a verification idea for the proof of the future anomaly detection algorithm. Next, in order to prevent the degradation of the anomaly detection model we design, we introduce a module for relearning. Experimental results compare Gaussian distribution, SVM, homogeneous ensemble learning and the latest Kitsune algorithm, which shows the superiority of our algorithm.

Declaration of Competing interest

The authors declare that they have no conflicts of interest to this work.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under grant no. 2018YFB1800205. And we also thank for the support of National Engineering Lab for Next Generation Internet Technologies (no. NGIT2019004).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2019.107049.

References

[1] D.E. Denning, An intrusion-detection model, IEEE Trans. Softw. Eng. SE-13 (2) (1987) 222–232.
[2] R. Sommer, V. Paxson, Outside the closed world: on using machine learning for network intrusion detection, IEEE Symposium on Security and Privacy. IEEE Computer Society (2010).
[3] P. Casas, S. Vaton, L. Fillatre, et al., Optimal volume anomaly detection and isolation in large-scale IP networks using coarse-grained measurements, Comput. Netw. 54 (11) (2010) 1750–1766.
[4] A. Abraham, C. Grosan, C. Martinvide, Evolutionary design of intrusion detection programs, Int. J. Netw. Secur. 4 (3) (2007) 328–339.
[5] P. Mishra, V. Varadharajan, U. Tupakula, E.S. Pilli, A detailed investigation and analysis of using machine learning techniques for intrusion detection, IEEE Communications Surveys & Tutorials 21 (1) (2019) 686–728. Firstquarter.
[6] J. Carr, Snort: Open source network intrusion prevention, 2007.
[7] K.E. Smith, P. Williams, K.J. Bryan, M. Solomon, M. Ble, R. Haber, Shepard interpolation neural networks with k-means: ashallow learning method for time series classification, 2018 International Joint Conference on Neural Networks (IJCNN) (2018).
[8] L. Shao, D. Wu, X. Li, Learning deep and wide: a spectral method for learning deep networks, IEEE Trans. Neural Netw. Learn.Syst. 25 (12) (2014) 2303–2308.
[9] J. Jin, G. Lai, G. Lai, GPU-Accelerated parallel algorithms for linear rankSVM, J. Supercomput. 71 (11) (2015) 4141–4171.

[10] E. Kijsipongse, A. Piyatumrong, S. U-Ruekolan, A hybrid GPU cluster and volunteer computing platform for scalable deep learning, J. Supercomput. 74 (7) (2018) 3236–3263.
[11] A.A. Aburomman, M.B.I. Reaz, A survey of intrusion detection systems based on ensemble and hybrid classifiers, Comput. Secur. 65 (2017) 135–152.
[12] G. Folino, P. Sabatino, Ensemble based collaborative and distributed intrusion detection systems: a survey, J. Netw. Comput. Appl. 66 (2016) 1–16.
[13] J.Z. Lei, A.A. Ghorbani, Improved competitive learning neural networks for network intrusion and fraud detection, Neurocomputing 75 (1) (2012) 135–145.
[14] A.A. Aburomman, M.B.I. Reaz, A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems, Inf. Sci. 414 (2017) 225–246.
[15] X. Jing, Z. Yan, W. Pedrycz, Security data collection and data analytics in the internet: A survey, IEEE Communications Surveys & Tutorials 21 (1) (2019) 586–618. Firstquarter.
[16] W. Lee, D. Xiang, Information-theoretic measures for anomaly detection, IEEE Symp. Secur. Privacy (2001) 130–143.
[17] M. Yu, A nonparametric adaptive CUSUM method and its application in network anomaly detection, Int. J. Adv. Comput.Technol. 4 (1) (2012) 280–288.
[18] B. Krishnamurthy, Z.Y. Subhabrata Sen, Y. Chen, Sketch-based change detection: methods, evaluation, and applications, in: In Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC '03), ACM, New York, NY, USA, 2003. 234–247.
[19] J.D. Brutlag, Aberrant behavior detection in time series for network service monitoring, in Proceedings of Usenix Conference on System Administration (2000) 139–146.
[20] A. Patcha, J.M. Park, An overview of anomaly detection techniques: existing solutions and latest technological trends, Comput. Netw. 51 (12) (2007) 3448–3470.
[21] T.T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tutor. 10 (4) (2008) 56–76.
[22] A.L. Buczak, E. Guven, A survey of data mining and machine learning methods for cyber security intrusion detection, IEEE Commun. Surv. Tutor. 18 (2) (2016) 1153–1176.
[23] M.S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, A.P. Sheth, Machine learning for internet of things data analysis: a survey, Digital Commun. Netw. 4 (3) (2017) 161–175.
[24] Ian goodfellow and yoshua bengio and aaron courville. deep learning, MIT Press (2016).
[25] N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Trans. Emerging Top.Comput. Intell. 2 (1) (2018) 41–50.
[26] F.A. Khan, A. Gumaei, A. Derhab, A. Hussain, TSDL: a two-stage deep learning model for efficient network intrusion detection, IEEE Access 7 (2019) 30373–30385.
[27] Y. Mirsky, T. Doitshman, Y. Elovici, et al., Kitsune: an ensemble of autoencoders for online network intrusion detection, Netw. Distrib. Syst. Secur. Symp. (2018).
[28] M. Du, F. Li, G. Zheng, et al., Deeplog: anomaly detection and diagnosis from system logs through deep learning, in: ACM Sigsac Conference on Computer and Communications Security, ACM, 2017, pp. 1285–1298.
[29] W. Wang, et al., HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection, IEEE Access 6 (2018) 1792–1806.
[30] F. Jiang, et al., Deep learning based multi-channel intelligent attack detection for data security, IEEE Transactions on Sustainable Computing (2018). 1–1.
[31] A.A. Aburomman, M.B.I. Reaz, A survey of intrusion detection systems based on ensemble and hybrid classifiers, Comput. Secur. 65 (2017) 135–152.
[32] J. Vanerio, P. Casas, Ensemble-learning approaches for network security and anomaly detection, Proc. Workshop Big Data Anal.Mach. Learn. Data Commun. Netw. (2017) 1–6.
[33] G. Folino, P. Sabatino, Ensemble based collaborative and distributed intrusion detection systems: a survey, J. Netw. Comput. Appl. 66 (2016) 1–16.
[34] A.J. Malik, W. Shahzad, F.A. Khan, Binary PSO and random forests algorithm for probe attacks detection in a network, 2011 IEEE congress on evolutionary computation (CEC) (2011) 662–668.
[35] V. Bukhtoyarov, V. Zhukov, Ensemble-distributed approach in classification problem solution for intrusion detection systems, in: Intelligent data engineering and automated learning (IDEAL), Springer, 2014, pp. 255–265.
[36] S. Masarat, H. Taheri, S. Sharifian, A novel framework, based on fuzzy ensemble of classifiers for intrusion detection systems, 2014 4th international conference on computer and knowledge engineering (ICCKE), IEEE, 2014.
[37] V. Rijn, N. Jan, et al., The online performance estimation framework: heterogeneous ensemble learning for data streams, Mach. Learn. (2018) 149–176.
[38] G.E. Hinton, A practical guide to training restricted Boltzmann machines, Momentum 9 (1) (2012) 599–619.
[39] M. Welling, G. Hinton, A new learning algorithm for mean field Boltzmann machines, in: International Conference on Artificial Neural Networks, Springer, 2002.
[40] G.E. Hinton, S. Osindero, Yee whye teh:a fast learning algorithm for deep belief nets, Neural Comput. (2006) 1527–1554.
[41] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2011) 1527–1554.
[42] G.E. Hinton, R.S. Zemel, Autoencoders, Minimum Description Length and Helmholtz Free Energy, in: International Conference on Neural Information Processing Systems, Morgan Kaufmann Publishers Inc, 1993, pp. 3–10.
[43] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, 1986, 399–421.
[44] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: continualprediction with LSTM, Neural Comput. (2014) 2451–2471.
[45] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. (1997) 1735–1780.
[46] R. Fontugne, P. Borgnat, P. Abry, K. Fukuda, MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking, ACM CoNEXT 2010, Philadelphia, PA, 2010.
[47] R. Fontugne, et al., Scaling in internet traffic: a 14 year and 3 day longitudinal study, with multiscale analyses and random projections, IEEE/ACM Trans. Netw. 25 (4) (2017) 2152–2165.
[48] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: IEEE International Conference On Data Mining, ICDM08, IEEE, 2008, pp. 413–422.
[49] D. Reynolds, Gaussian mixture models, Encycl. Biometrics (2015) 827–832.
[50] S.K. Sahu, S.K. Jena, A multiclass SVM classification approach for intrusion detection, Int. Conf. Distrib. Comput. Internet Technol. (2016).
[51] B. Yan, G. Han, Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system, IEEE Access (2018) 41238–41248.
[52] U. Fiore, F. Palmieri, A. Castiglione, A.D. Santis, Network anomaly detection with the restricted boltzmann machine, Neurocomputing (2013) 13–23.
[53] S. Papadimitriou, C. Faloutsos, A. Brockwell, Adaptive, hands-off stream mining, 29th Int. Conf. Very Large Data Bases (2003).
[54] D. Ippoliti, C. Jiang, Z. Ding, X. Zhou, Online adaptive anomaly detection for augmented network flows, ACM Trans. Auton. Adapt. Syst. 11 (3) (2016) 1–28. 2016.
[55] E. Viegas, A. Santin, A. Bessani, N. Neves, Bigflow: real-time and reliable anomaly-based intrusion detection for high-speed networks, Future Gener. Comput. Syst. 93 (3) (2019) 473–485.
[56] B.A. Tama, K.H. Rhee, An in-depth experimental study of anomaly detection using gradient boosted machine, Neural Comput. Appl. 31 (4) (2019) 955–965.
[57] B.A. Tama, M. Comuzzi, K. Rhee, TSE-IDS: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system, IEEE Access 7 (2019) 94497–94507.
[58] B. Hang, R.M. Hu, W. Shi, An enhanced SYN cookie defence method for TCP DDos attack, JNW 6 (2011) 1206–1213.
[59] L.J. Tashman, Out-of-sample tests of forecasting accuracy: an analysis and review, Int. J. Forecast. 16 (2000) 437–450.
[60] C. Bergmeir, J.M. Bentez, On the use of cross-validation for time series predictor evaluation, Inf. Sci. 191 (2012) 192–213.
[61] Z.H. Zhou, Ensemble Methods - Foundations and Algorithms, Taylor & Francis, 2012.
[62] W.J. Conove, Practical Nonparametric Statistics 3rd edition, John Wiley and Sons, Michigan, 1999.
[63] S. García, et al., Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Inf. Sci. (2010) 2044–2064.

Ying Zhong received master degree at the College of Information Science and Engineering, Hunan University, China. He is working towards the Ph.D. degree at the Institute for Network Sciences and Cyberspace at Tsinghua University. His research interests include the machine learning, big data, network security, data analysis and mining algorithms and their parallel implementation.

Wenqi Chen is now working for an undergraduate degreee in computer science and technology at Tsinghua University. His research interests include Cyberspace security and network intrusion detection.

Zhiliang Wang received the B.E., M.E. and Ph.D. degrees in computer science from Tsinghua University, China in 2001, 2003 and 2006 respectively. Currently he is an Associate Professor in the Institute for Network Sciences and Cyberspace at Tsinghua University. His research interests include formal methods and protocol testing, next generation Internet, network measurement.

Yifan Chen is now a junior student of Beijing University of Posts and Telecommunications, Beijing, P.R.China. His research interests include Computer Network and Intrusion Detection System.

Kai Wang is now working for an undergraduate degreee at University of Electronic Science and Technology of China. His research interests include Cyberspace security and network intrusion detection.

Yahui Li received her B.S. degree from the College of Software from Jilin University, China in 2015. She is now pursuing her Ph.D. at Tsinghua University. Her research concerns network verification, network testing and formal methods.

Xia Yin received her B.E., M.E. and Ph.D. degrees in computer science from Tsinghua University in 1995, 1997 and 2000, respectively. She is a Full Professor at the Department of Computer Science and Technology at Tsinghua University. Her research interests include future Internet architectures, formal methods, protocol testing and largescale Internet routing.

Xingang Shi received his B.S. degree from Tsinghua University and his Ph.D. degree from The Chinese University of Hong Kong. He is now working at the Institute for Network Sciences and Cyberspace at Tsinghua University. His research interests include network measurement and routing protocols.

Jiahai Yang received his M.S. and Ph.D. degrees in computer science from Tsinghua University, Beijing, P.R.China, in 1992 and 2003, respectively. He is now a professor of Tsinghua University. Jiahais research interests include Internet architecture and its protocols, IP routing technology, network measurement, network management, cloud computing, big data, etc.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor at Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyberphysical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 690 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.