Intrusion Detection Based on Self-adaptive Differential Evolutionary Extreme Learning Machine

Junhua Ku
Department of information engineering
Hainan institute of science and technology
Haikou, China
E-mail: kujunhua@163.com

Dawei Yun
Department of information engineering
Hainan institute of science and technology
Haikou, China
E-mail: cogemm@163.com

Bing Zheng
Department of information engineering
Hainan institute of science and technology
Haikou, China
E-mail: zhbahn@vip.qq.com

***Abstract*—Nowadays with the rapid development of network-based services and users of the internet in everyday life, intrusion detection becomes a promising area of research in the domain of security. Intrusion detection system (IDS) can detect the intrusions of someone who is not authorized to the present computer system automatically, so intrusion detection system has emerged as an essential component and an important technique for network security.**

Extreme learning machine (ELM) is an interested area of research for detecting possible intrusions and attacks. In this paper, we propose an improved learning algorithm named self-adaptive differential evolution extreme learning machine (SADE-ELM) for classifying and detecting the intrusions. We compare our methods with commonly used ELM, DE-ELM techniques in classifications. Simulation results show that the proposed SADE-ELM approach achieves higher detection accuracy in classification case.

Keywords-Extreme learning machines; Differential evolution extreme learning machines; Self-adaptive differential evolution extreme learning machines; Intrusion detection; Network security

I. INTRODUCTION

Intrusion into computer networks and systems is a major threat in today's network centric world. Few most prevalent intrusion attacks include Denial-of-Service (DoS) attacks, Distributed- Denial-of-Service (DDoS) at-tacks, probing based attacks and account takeover attacks. Intrusion detection identifies computer attacks by observing various records processed on the network. Intrusion detection models are classified into two variants, misuse detection and anomaly detection systems. Misuse detection can discover intrusions based on a known pattern also known as signatures [1]. Anomaly detection can identify the malicious activities by observing the deviation from normal network traffic pattern [2]. Hence anomaly detection can identify new anomalies. The difficulty with the current developmental techniques is the high false positive rate and low false negative rate.

Recently, a new fast learning neural algorithm for SLFNs, named extreme learning machine (ELM) [3,4], was devel-oped to improve the efficiency of SLFNs. Different from the conventional learning algorithms for neural networks (such as BP algorithms[5]), which may face difficulties in manually tuning control parameters (learning rate, learn-ing epochs, etc.) and/or local minima, ELM is fully auto-matically implemented without iterative tuning, and in theory, no intervention is required from users. Further-more, It was popular for its fast training speed by means of utilizing random hidden node parameters and calculating the output weights with least square algorithm [6-10]. However, in ELM, the number of hidden nodes is assigned a priori, the hidden node parameters are randomly chosen and they remain unchanged during the training phase. Many non-optimal nodes may exist and contribute less in minimizing the cost function. Moreover, in [11] Huang et al. pointed out that ELM tends to require more hidden nodes than conventional tuning-based algorithms [12,13] in many cases.

Differential evolution (DE) [14] which is a simple but powerful population-based stochastic direct searching technique is a frequently used method for selecting the network parameters [15–17]. In [15], DE is directly adopt-ed as a training algorithm for feed forward networks where all the network parameters are encoded into one population vector and the error function between the network approximate output and the expected output is used as the fitness function to evaluate all the populations. However, Subudhi and Jena [16] have pointed out that using the DE approach alone for the network training may yield a slow convergence speed. Therefore, in [17], a new algorithm named evolutionary extreme learning machine (DE-ELM)

based on DE and ELM has been developed for SLFNs. Using the DE method to optimize the network input parameters and the ELM algorithm to calculate the network output weights, DE-ELM has shown several promising features. It not only ensures a more compact network size than ELM, but also has better generalization performance.

However, in the above DE based neural network training algorithms, the trial vector generation strategies and the control parameters in DE have to be manually chosen. For example, the control parameters in DE-ELM are manually selected according to an empirical suggestion and the simple random generation method is adopted to produce the trial vector. As pointed out by many researchers, the performance of the DE algorithm highly depends on the chosen trial vector generation strategy and the control parameters, and inappropriate choices of strategies and control parameters may result in premature convergence or stagnation. Therefore, we propose a novel learning algorithm named self-adaptive evolutionary extreme learning machine (SaDE-ELM) for SLFNs. In SaDE-ELM, the hidden node learning parameters are optimized by the self-adaptive differential evolution algorithm. We verify our approach using the data originated from the 1998 DARPA Intrusion Detection Evaluation Program 1999, which is adopted in the Data Mining and Knowledge Discovery (KDD) competition [18], and considered as a common benchmark for evaluating intrusion detection techniques [19-21]. In this benchmark, there are four types of attacks, Denial of Service (DoS) attack, user to root attack, remote to user attack and probing attack. A denial of service attack is an attempt to make a computer resource unavailable or respond slowly to its legitimate users. User to root attack basically tries to exploit vulnerability to gain root access to the system. Remote to user attack is that attackers remotely exploit vulnerability of a machine to gain local access as a user. Probing are attacks that are trying to access computers, computer systems, networks or applications for weakness. In the following, we first review common methods for intrusion detection and classification.

The rest of the paper is organized as follows. In section II, a brief introduction to ELM and SaDE are given. In Section III, we introduce model of proposed SaDE-ELM algorithm in detail. In Section IV, we present the dataset we use in our numerical studies and our intrusion detection approach. Experiments for detecting intrusion in network traffic data and performance comparisons between ELM-based techniques and DE-ELM-based techniques are presented in section. In section V, we conclude and summarize our results.

II. BACKGROUND

As a novel training algorithm for SLFNs, ELM is very efficient and effective. In this section, we will give a brief review of ELM. In this section, we briefly review ELM and SaDE-ELM approach for ELM is the foundation of SaDE-ELM.

A. Extreme learning machine (ELM)

For N arbitrary distinct samples $(\mathrm{x}_j, \mathrm{t}_j)$, where $\mathrm{x}_j = [x_{j1}, x_{j2}, \cdots, x_{jn}]^T \in \mathbb{R}^n$, $\mathrm{t}_j = [t_{j1}, t_{j2}, \cdots, t_{jm}]^T \in \mathbb{R}^m$,

SLFNs with L hidden nodes and activation function $g(x)$ are

$$\sum_{i=1}^{L} \beta_i \, \mathrm{g}_i(\mathrm{x}_j) = \sum_{i=1}^{L} \beta_i \, \mathrm{g}_i(\mathrm{w}_i \cdot \mathrm{x}_j + b_j) = \mathrm{o}_j \qquad (j = 1, 2, ..., N) \tag{1}$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, b_i is the threshold of the ith hidden node, $\mathbf{w}_i \cdot \mathbf{x}_j$ denotes the inner product of w_i and x_j, $g(x)$ is activation function and Sigmoid, Sine, Hardlim and other functions are commonly used. The output nodes are chosen linear in this paper, and $\mathrm{o}_j = [o_{j1}, o_{j2}, ..., o_{jm}]^T$ is the jth output vector of the SLFNs [22].

The SLFNs with L hidden nodes and activation function g(x) can approximate these N samples with zero error. It means $\sum_{j=1}^{L} \left\| \mathrm{o}_j - \mathrm{t}_j \right\| = 0$ and there exist β_i, $\mathbf{w}_i$ and b_i such that

$$\sum_{i=1}^{L} \beta_i \, \mathrm{g}_i(\mathrm{x}_j) = \sum_{i=1}^{L} \beta_i \, \mathrm{g}_i(\mathrm{w}_i \cdot \mathrm{x}_j + b_j) = \mathrm{t}_j \qquad (j = 1, 2, ..., N) \tag{2}$$

The equation above can be expressed compactly as follows:

$$\mathbf{H}\beta = \mathbf{T} \tag{3}$$

Where

$$\mathrm{H}(\mathrm{w}_1, \mathrm{w}_2, \cdots, \mathrm{w}_L, b_1, b_2, \cdots, b_L, \mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_L)$$

$$= [h_{ij}] = \begin{bmatrix} \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_1 + b_1) & \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_1 + b_2) & \cdots & \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_1 + b_L) \\ \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_2 + b_1) & \mathrm{g}(\mathrm{w}_2 \cdot \mathrm{x}_2 + b_2) & \cdots & \mathrm{g}(\mathrm{w}_L \cdot \mathrm{x}_2 + b_L) \\ \vdots & \vdots & & \vdots \\ \mathrm{g}(\mathrm{w}_1 \cdot \mathrm{x}_N + b_1) & \mathrm{g}(\mathrm{w}_2 \cdot \mathrm{x}_N + b_2) & \cdots & \mathrm{g}(\mathrm{w}_L \cdot \mathrm{x}_N + b_L) \end{bmatrix}_{N \times L}$$

$$\beta = \begin{pmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{L1} & \beta_{L2} & \cdots & \beta_{Lm} \end{pmatrix} \quad \text{and} \quad \mathrm{T} = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1m} \\ t_{21} & t_{22} & \cdots & t_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ t_{N1} & t_{N2} & \cdots & t_{Nm} \end{pmatrix}$$

The matric H is called the hidden layer output matrix of the neural network and the ith column of H is the ith hidden node output with respect to inputs x1 , x2, ..., xN .

By simply randomly choosing hidden nodes and then adjusting the output weights, single hidden layer feedforward networks (SLFNs) work as universal approximators with any bounded non-linear piecewise

continuous functions for additive nodes [23]. ELM algorithm claims that the hidden node parameters can be randomly assigned [3,4], then the system equation becomes a linear model and the network output weights can be analytically determined by finding a least-square solution of this linear system as follow

$$\widehat{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{4}$$

Where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix **H**. Then the output function of ELM can be modeled as follows.

$$\phi(\xi) = \gamma(\xi)\circledR = \gamma(\xi)\mathbf{H}\square\mathbf{T} \tag{5}$$

Moreover, it should be noted that many nonlinear activation and kernel functions can be used in ELM.

B. Self-adaptive differential evolution

Differential evolution (DE), proposed by Storn and Price in 1995, is a simple yet powerful evolutionary algorithm (EA) [24]. There are three parameters in DE algorithm, which are the population size *NP*, mutation scaling factor *F* and crossover rate *CR*. *NP* is a problem-dependent parameter, while *F* and *CR* are very sensitive to the performance at different stages of evolution. To overcome the limitations of choosing the parameters in DE, Brest et al. [25] proposed a parameter adaptation technique to choose the mutation scaling factor *F* and crossover rate *CR* namely SADE-ELM algorithm which performs better than the basic DE algorithm. In general, SADE algorithm is composed of three main steps: mutation, crossover, and selection [26].

We consider the following optimization problem:

Minimize $f(x_i)$, $x_i \in R_D$

where $x_i = [x_{i1}, x_{i2}, \cdots, x_{iD}]^T, i = 1, 2, \cdots, NP$ is a target vector of D decision variables. During the mutation operation, mutant vector v_i is generated by mutation strategy in the current population:

$$v_i = x_{r1} + F \cdot (x_{r2} - x_{r3}) \tag{6}$$

where $r1, r2, r3$ are mutually exclusive integers randomly chosen in the range [1,*NP*], and $r1 \neq r2 \neq r3 \neq i$.

Following mutation, trial vector u_i is generated between x_i and v_i during crossover operation where the most widely used operator is the binomial crossover performed as follows:

$$u_{ij} = \begin{cases} v_{ij}, & if\ (\text{rndreal}(0,1) < \text{CR} \ \text{ or } \ j = j_{\text{rand}}), \\ x_{ij}, & otherwise \end{cases} \tag{7}$$

Where j_{rand} is a integer randomly chosen in the range [1, *D*], and rndreal(0, 1) is a real number randomly generated in (0, 1). Finally, to keep the population size constant during the evolution, the selection operation is used to determine whether the trial or the target vector survives to the next generation according to one-to-one selection:

$$x_i = \begin{cases} u_{i,} & if\ (f(u_i)\, f(x_i)) \\ x_i, & otherwise \end{cases} \tag{8}$$

Where $f(x)$ is the optimized objective function. During the evolution, *F* and *CR* are adaptively tuned to improve the performance of DE for each individual

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \cdot F_u & if\ (rand_2 < \tau_1) \\ F_{i,G} & \text{otherwise} \end{cases} \tag{9}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & if\ (rand_4 < \tau_2) \\ CR_{i,G} & \text{otherwise} \end{cases} \tag{10}$$

Where $F_{i;G+1}$ and $CR_{i;G+1}$ are the mutation scaling factor and crossover rate for *i* individual in *G* generation respectively, *randj*=1;2;3;4 are randomly chosen from (0, 1), τ_1 and τ_2 both valued 0.1 which is used to control the generation of *F* and *CR*, *Fl* valued 0.1 and *Fu* is valued 0.9. In the first generation, *F* and *CR* are initialized to 0.5.

III. MODEL OF PROPOSED SADE-ELM ALGORITHM

Since the ELM generates the input weights and hidden biases arbitrarily which are the basic of calculating the output weights, it may not reach the optimal result in classification or regression. Thus, a hybrid approach integrated self-adaptive differential evolution algorithm and extreme learning machine namely SADE-ELM algorithm to optimize the input weights and hidden biases is able to obtain better generalization performance than ELM algorithm [17].

In SaDE-ELM, we proposed SaDE-ELM for SLFNs by incorporating the self-adaptive differential evolution algorithm [25] to optimize the network input weights and hidden node biases and the extreme learning machine to derive the network output weights.

Given a set of training data and L hidden nodes with an activation function g(·), we summarize the SaDE-ELM algorithm in the following steps.

Step 1. Initialization

A set of *NP* vectors where each one includes all the network hidden node parameters are initialized as the populations of the first generation

$$\theta_{k,G} = \left[\mathrm{w}_{1,k,G}^T, \cdots, \mathrm{w}_{L,k,G}^T, b_{1,k,G}^T, \cdots, b_{L,k,G}^T\right] \tag{11}$$

where w_j and b_j (j = 1, … , L) are randomly generated, *G* represents the generation and k = 1, 2,…, *NP*.

Step 2. Calculations of output weights and RMSE

Calculate the network output weight matrix and root mean square error (RMSE) with respect to each population vector with the following equations, respectively.

$$\beta_{k,G} = \mathrm{H}^{\dagger}_{k,G}\mathrm{T} \tag{13}$$

$$\mathrm{RMSE}_{k,G} = \sqrt{\frac{\sum_{i=1}^{N}\left\|\sum_{j=1}^{L}\beta_j g(\mathrm{w}_{j,k,G},b_{j,k,G},x_i)-t_i\right\|}{m\times N}} \tag{14}$$

Then use the value of RMSE to calculate the new best population vector $\theta_{k,G+1}$ with the following equation.

$$\theta_{k,G+1} = \begin{cases} u_{k,G+1} & if\ (\mathrm{RMSE}_{\theta_{k,G}}\text{-}\mathrm{RMSE}_{u_{k,G+1}})>\varepsilon\cdot\mathrm{RMSE}_{\theta_{k,G}} \\ u_{k,G+1} & if\ \left|\mathrm{RMSE}_{\theta_{k,G}}\text{-}\mathrm{RMSE}_{u_{k,G+1}}\right|<\varepsilon\cdot\mathrm{RMSE}_{\theta_{k,G}} \\ & and\left\|\beta_{u_{k,G+1}}\right\|<\left\|\beta_{\theta,G}\right\| \\ \theta_{i,G} & \text{otherwise} \end{cases} \tag{15}$$

where ε is the preset small positive tolerance rate. In the first generation, the population vector with the best RMSE is stored as θbest,1 and RMSEθbest,1 .

All the trial vectors uk,G+1 generated at the (G+1)th generation are evaluated using equation(11) .The norm of the output weight $\|\beta\|$ is added as one more criteria for the trial vector selection as pointed out in [23] that the neural networks tend to have better generalization performance with smaller weights.

The three operations mutation, crossover and selection are repeated until the preset goal is met or the maximum learning iterations are completed. At last we calculate the output weigh $\beta = [\beta_{i1}\ \beta_{i2}\ \cdots\ \beta_{iL}]^T$ with equation $\beta = \mathbf{H}^{\dagger}\mathbf{T}$.

IV. INTRUSION DETECTION USING SADE-ELM

In this section, we describe the dataset that we use for our numerical studies, and our SaDE-ELM approach to classification of intrusions in the data.

A. Dataset Description

The dataset we use is from the 1998 DAPRA intrusion detection program. During the evaluation program, an environment was set up in Lincoln Labs to record 9 weeks of raw TCP/IP dump data for a network simulating a typical U.S. air force LAN. Then the LAN was operated under a real environment and blasted with multiple attacks. After that, 7 weeks of raw tcpdump data was processed into millions of connection records. Finally, 41 quantitative and qualitative features were extracted using data mining techniques. The detail of the feature extraction can be found in [27].

Four main categories of attacks were simulated:

1) DoS: denial of service attack

2) R2L: unauthorized access from a remote machine

3) U2R: unauthorized access to local root previledges

4) Probing: surveillance and other probing

In the intrusion detection simulation, the dataset was labeled with 22 attack types falling into the four categories shown in Table I. The feature list and its descriptions are in Tables II, III and IV.

TABLE I. TABLE I ATTACK TYPE

Denial of Service	User to Root	Remote to User	Probing
Back	Perl	FTP Write	IP Sweep
Neptune	Buffer Overflow	Guess Password	Nmap
Land Teardrop	Load Module	Imap	Port Sweep
Ping of Death	Rootkit	Multihop	Satan
Smurf		Phf	
		Spy	
		Warezclient	
		Warezmaster	

B. Intrusion Detection System using SaDE-ELM

Our SaDE-ELM intrusion detection method has the following steps. We also use a ELM method to classify the data to provide a comparison benchmark.

Step 1. Data pre-processing: a data processing script is used to convert the raw TCP/IP dump data into machine readable form.

Step 2. Training phase: SaDE-ELM and ELM are trained on normal data and different types of attacks. For the binary classification case, the data has 41 features and falls into 2 classes: normal and attack; for the multi-class classification case, the data has 41 features and falls into 23 classes: normal and 22 types of attack. The model is trained in a large program which can test immediately after the training completed. According to SaDE-ELM theory that has been introduced above, we can summarize the following steps.

For N arbitrary distinct samples (xi,ti), i = 1,... N,and hidden nodes and activation function g(x):

2.1) A set of NP individual parameter vectors $\theta_{k,G}$ (k = 1, 2 . . . NP), where each one includes all the network hidden node parameters are initialized as the populations of the first generation;

2.2) In the case of g(x) and L are invariable run the three operations including mutation, crossover and selection to produce the new population, and the process is repeated until the stop condition is completed.

2.3) Changing the type of g(x) and increase the number of hidden nodes L gradually from one to find the most suitable g(x) and L to construct an optimal forecasting model with the best testing accuracy;

2.4) Calculating the output matrix according to Eq.(4);

2.5) Calculating the output weights $\beta = \mathrm{H}^{\dagger}\mathrm{H}$, where T = [t1, . . . tN] and $\mathbf{H}^{\dagger} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}$.

Step 3. Testing phase: ELM, DE-ELM and SaDE-ELM are used to predict the type of each data point in the testing dataset, and their performances are compared.

Both ELM and SaDE-ELM cannot process symbolic data, so the following method is used to convert symbolic data into continuous data without affecting the performance. As can be seen from the feature description table, there are several symbolic features in the dataset. For features like land, logged in, root shell, is host login and is guest login that take values 0 or 1, so we can handle these features in the same way as continuous features. Other features like protocol, service and flag have more than 2 different values. For example, there are three different values in feature protocol TCP, UDP and ICMP. We represent these three category attributes TCP, UDP, ICMP using (0,0,1), (0,1,0) and (1,0,0). The same method is applied to encode the features service and flag. Experiments have shown that if the number of values in an attribute is not too large, this coding is more stable than using a single number. The simulation of the three algorithms on all datasets are carried out using MATLAB 2013a on a machine with an Intel Core 2 Duo, 2.26GHz CPU and 4GB RAM.

V. SIMULATION RESULTS

The datasets being tested are 2000, 4000, 8000 connection data chosen randomly from the dataset downloaded from the website [18]. We split them equally into training data and testing data. Simulation results including average testing accuracy and corresponding 95% confidence interval are given in Table IV.

In order to test the relationship between SaDE-ELM and the number of hidden layer, according to the different number of hidden layer nodes, we made classification tests using ELM, DE-ELM and SaDE-ELM respectively. Simulation results are given in Table V.

Figure1 and Fig .2 show the time spent by ELM, DE-ELM and SaDE-ELM when training and testing the same size of dataset. It can be seen that the training time and testing time spent by SaDE-ELM increase sharply when the size of data increases. In comparison, ELM and DE-ELM increase slowly when the number of data increases. Eventually, DE-ELM starts consuming more time for both training and testing than ELM.

A clear time consumption comparison can be seen from Fig .1and Fig .2. From the results, we can conclude that ELM performs better than DE-ELM and SaDE-ELM in terms of speed. To increase accuracy, we can implement SaDE-ELM. This shows that our proposed SaDE-ELM methods have better scalability than ELM and DE-ELM when classifying network traffic for intrusion detection.

TABLE II. TABLEII BASIC FEATURES OF INDIVIDUAL TCP CONNECTIONS

feature name	description	type
Duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol	discrete discrete
service	network service on the destination	continuous continuous
src_bytes	number of data bytes from source to destination	discrete discrete
dst_bytes	number of data bytes from destination to source	continuous continuous
flag	normal or error status of the connection	
land	1 if connection is from/to the same host/port, 0 otherwise	
wrong_fragment	number of “wrong” fragments	
urgent	number of urgent packets	

TABLE III. TABLEIII CONTENT FEATURES WITHIN A CONNECTION SUGGESTED BY DOMAIN KNOWLEDGE

feature name	description	type
hot	number of “hot” indicators	continuous
num failed logins	number of failed login attempts	continuous
logged in	1 if successfully logged in, 0 otherwise	discrete
num compromised	number of “compromised” conditions	continuous
root shell	1 if root shell is obtained, 0 otherwise	discrete
su attempted	1 if “su root” command attempted, 0 otherwise	discrete
num root	number of “root” accesses	continuous
num file creations	number of file creation operations	continuous
num shells	number of shell prompts	continuous
num access files	number of operations on access control files	continuous
num outbound cmds	number of outbound commands in an ftp session	continuous
is hot login	1 if the login belongs to the “hot” list, 0 otherwise	discrete
is guest login	1 if the login is a “guest”login, 0 otherwise	discrete

TABLE IV. TABLE IV TRAFFIC FEATURES COMPUTED USING A TWO-SECOND TIME WINDOW

feature name	description	type
count	number of connections to the same host as the current connection in the past two seconds	continuous
serror rate	% of connections that have “SYN” errors	continuous
rerror rate	% of connections that have “REJ” errors	continuous
same srv rate	% of connections to the same service	continuous
diff srv rate	% of connections to different services	continuous
srv count	number of connections to the same service as the current connection in the past two seconds	continuous
srv serror rate	% of connections that have “SYN” errors	continuous
srv rerror rate	% of connections that have “REJ” errors	continuous
srv diff host rate	% of connections to different hosts	continuous

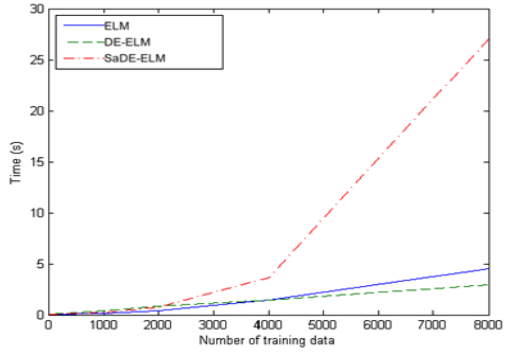

Figure 1. Training time comparison

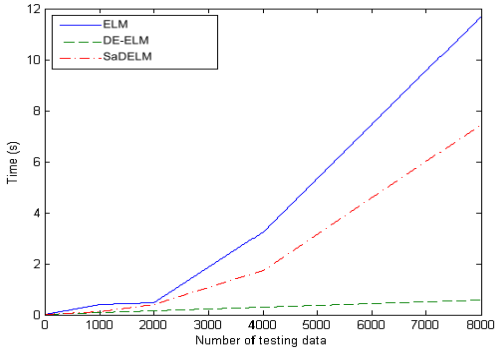

Figure 2. Testing time comparison

TABLE V. TABLE V. PERFORMANCE COMPARISON RESULTS

Dataset Size	ELM		DE-ELM		SaDE-ELM	
Training/Testing	Accuracy (%)	95% Confidence Interval (%)	Accuracy (%)	95% Confidence Interval (%)	Accuracy (%)	95% Confidence Interval
1000/1000	99.32	99.08 - 99.47	99.33	99.15 - 99.51	99.55	99.05 - 99.65
2000/2000	99.10	98.82 - 99.23	99.24	98.90 - 99.44	99.47	99.25 - 98.58
4000/4000	99.07	98.79 - 9.28	99.18	99.01 - 99.26	99.35	99.11 - 99.65

VI. CONCLUSION

In this paper, we have made a comparison by the use of ELM, DE-ELM and SaDE-ELM for intrusion detection in a computer network. For the SaDE-ELM, By incorporating the self-adaptive differential evolution algorithm to optimize the network hidden node parameters and employing the extreme learning machine to derived the network output weights. Obviously, the proposed SaDE-ELM can obtain higher accuracy.

Whether to use ELM, DE-ELM or SaDE-ELM in implementing an intrusion detection system depends on the type of intrusion likely to occur. For example in a DDoS attack, the attacker usually controls thousands of agents to send a large number of TCP SYN packets to a victim’s server port. When the port is actively listening for connection requests, the victim would respond by sending back ACK packets. However, the victim will not get further responses and keep the connections half-open, which would eventually quickly consume all the memory allocated for pending connections. The victim’s server would then no longer be able to process new requests from normal clients. If we can correctly detect more than 90% of the attack connections and drop these, we can effectively prevent the DDoS attacker from overwhelming the server. For DDoS attack detection, basic ELM with sigmoid additive neurons would be a good choice since it has significantly shorter training times compared to other techniques. On the other hand, attacks like user to root attack exploit the victim’s vulnerability to gain root access and may not create as many connections as DDoS attack. Each connection by a

successful attack however provides root access to the system. Therefore, in this case, detection accuracy matters more than speed. To detect this kind of attack, SaDE-ELM would be preferred.

ACKNOWLEDGMENT

This research was supported by science research project of Education Department of Hainan province (Hnky2017ZD-20).

REFERENCES

[1] Ilgun, K., Kemmerer, R.A., Porras, P.A., 1995. State transition analysis: a rule-based intrusion detection approach. IEEE Trans. Software Eng. 21 (3), 181–199.

[2] Ikram S T, Cherukuri A K. Improving Accuracy of Intrusion Detection Model Using PCA and optimized SVM[J]. CIT. Journal of Computing and Information Technology, 2016, 24(2): 133-148.

[3] Huang GB, Zhu QY, Siew CK. Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), vol 2, no 25–29, pp 985–990.

[4] Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501.

[5] Espana-Boquera S, Zamora-Mart ńez F, Castro-Bleda M J, et al. Efficient BP algorithms for general feedforward neural networks[C]//International Work-Conference on the Interplay Between Natural and Artificial Computation. Springer Berlin Heidelberg, 2007: 327-336.

[6] G. Thatte, U. Mitra, and J. Heidemann, "Parametric methods for anomaly detection in aggregate traffic," IEEE/ACM Transactions on Networking, vol. 19, no. 2, pp. 512–525, April 2011.

[7] M. Qin and K. Hwang, "Frequent episode rules for internet anomaly detection," in Proceedings of the Network Computing and Applications, Third IEEE International Symposium. Washington, DC, USA: IEEE Computer Society, 2004, pp. 161–168.

[8] X. He, C. Papadopoulos, J. Heidemann, U. Mitra, and U. Riaz, "Remote detection of bottleneck links using spectral and statistical methods," Computer Networks, vol. 53, pp. 279–298, February 2009.

[9] W. W. Streilein, R. K. Cunningham, and S. E. Webster, "Improved detec- tion of low-profile probe and denial-of-service attacks," in Proceedings of the 2001 Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection, June 2001.

[10] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, pp. 273–297, 1995.

[11] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," International Journal of Machine Leaning and Cybernetics, vol. 2, no. 2, pp. 107–122, 2011.

[12] G. Tandon, "Weighting versus pruning in rule validation for detecting network and host anomalies," in In Proceedings of the 13th ACM SIGKDD international. ACM Press, 2007.

[13] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," Computers & Security, vol. 25, pp. 439–448, 2002.

[14] Storn R, Price K (2004) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359

[15] Ilonen J, Kamarainen JI, Lampinen J (2003) Differential evolution training algorithm for feedforward neural networks. Neural Process Lett 17:93–105

[16] Subudhi B, Jena D (2008) Differential evolution and levenberg marquardt trained neural network scheme for nonlinear system identification. Neural Process Lett 27:285–296.

[17] Zhu Q-Y, Qin A-K, Suganthan P-N, Huang G-B (2005) Evolutionary extreme learning machine. Pattern Recog 38(10):1759–1763

[18] (1999)KDDCUPdataset. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[19] S. Mukkamala and A. Sung, "Detecting denial of service attacks using support vector machines," in Proceedings of the 12th IEEE International Conference on Fuzzy Systems, 2003.

[20] M. Luo, L. Wang, H. Zhang, and J. Chen, "A research on intrusion detection based on unsupervised clustering and support vector machine," in Information and Communications Security, ser. Lecture Notes in Computer Science, S. Qing, D. Gollmann, and J. Zhou, Eds. Springer Berlin / Heidelberg, 2003, vol. 2836, pp. 325–336.

[21] D. Kim and J. Park, "Network-based intrusion detection with support vector machines," in Information Networking, ser. Lecture Notes in Computer Science, H.-K. Kahng, Ed. Springer Berlin / Heidelberg, 2003, vol. 2662, pp. 747–756.

[22] Lin, Y., Lv, F., Zhu S., Yang, M., Cour, T., Yu, K., Cao, L., Huang, T.S.: Large-scale image classification: fast feature extraction and SVM training. In: Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1689-1696 (2011).

[23] Huang G-B, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw17(4):879-892

[24] R. Storn, K. Price, Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 11 (1997) 341-359.

[25] J. Brest, S. Greiner, B. Bo˘skovi´c, M. Mernik, V. ˇZumer, Self-adapting control parameters in differential evolution: A comprehensive study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation, 10 (2006) 646-657.

[26] J. Wu, Z. H. Cai, Attribute Weighting via Differential Evolution Algorithm for Attribute Weighted Naive Bayes (WNB), Journal of Computational Information Systems, 7 (2011) 1672-1679.

[27] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Costbased modeling for fraud and intrusion detection: results from the JAM project," in Proceedings of DARPA Information Survivability Conference and Exposition, vol. 2, January 2002, pp. 130‑144..