

# Extreme Learning Machines for Intrusion Detection

Chi Cheng, Wee Peng Tay and Guang-Bin Huang

School of Electrical and Electronic Engineering

Nanyang Technological University

Nanyang Avenue, Singapore 639798

Email: {ch0004hi,wptay,egbhuang}@ntu.edu.sg

**Abstract**—We consider the problem of intrusion detection in a computer network, and investigate the use of extreme learning machines (ELMs) to classify and detect the intrusions. With increasing connectivity between networks, the risk of information systems to external attacks or intrusions has increased tremendously. Machine learning methods like support vector machines (SVMs) and neural networks have been widely used for intrusion detection. These methods generally suffer from long training times, require parameter tuning, or do not perform well in multi-class classification. We propose a basic ELM method based on random features, and a kernel based ELM method for classification. We compare our methods with commonly used SVM techniques in both binary and multi-class classifications. Simulation results show that the proposed basic ELM approach outperforms SVM in training and testing speed, while the proposed kernel based ELM achieves higher detection accuracy than SVM in multi-class classification case.

**Index Terms**—Extreme Learning Machines, Support Vector Machines, intrusion detection, random features.

## I. INTRODUCTION

Computer network security is an active and diverse research topic. Networks of all sizes are targets of attackers who seek to disrupt and disable network traffic. Websites such as Ebay, Yahoo, and Amazon.com have been compromised by Distributed Denial of Service (DDoS) attack in 2000 [1], even though they were believed to have strong defenses. A successful Denial of Service (DoS) attack could tremendously degrade network performance and result in monetary losses in millions of dollars. Thus, development of countermeasures to different attacks has been of great interest. Current countermeasures under development focus on detection of anomalies, their prevention, or a combination of both.

We consider the problem of intrusion detection in a computer network in this paper. Intrusion of a computer network refers to the unauthorized access or attempt to access a network so as to disrupt or damage the network, or to steal information from the network [2]. We propose an extreme machine learning (ELM) approach [3], [4], [5] to classify and detect network intrusions. 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 [6], and considered as a common benchmark for evaluating intrusion detection techniques [2], [7], [8], [9]. 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.

## A. Related Works

There are many nonparametric approaches for intrusion detection in network traffic, such as rule based network intrusion detection systems [10], clustering based techniques [11], [12], machine learning methods like support vector machines (SVM) and neural networks [2], [7]. Parametric methods that rely on statistical models and detection techniques have also been investigated in [13]. In this paper, our focus is on machine learning techniques, so we will not describe any parametric approaches in the sequel.

The most commonly used systems today are rule based detection systems. If the rules are not robust enough in describing the characteristic of the attack, then simple modifications to the attack can be made which will allow it to succeed. Rule based systems rely on humans to create, test and deploy the rules. Thus, it may take hours or days to generate a new rule for an attack. In [14], the authors introduce a new internet trace technique for generating frequent episode rules to characterize internet traffic events. Then these rules are used to distinguish anomalous sequences of TCP, UDP or ICMP connections from normal traffic. However, attackers can easily adapt their attack techniques to bypass those rules.

Clustering based techniques usually involves computation of pairwise distances for all data instances, which makes its computational complexity a bottleneck. Several clustering based techniques are effective only when the intrusions do not form significant clusters among themselves. For example, in distributed DoS attack, the network traffic can form a single cluster, with no abnormal cluster being detected. K-nearest Neighbor Classifier (KNNC) [11] has been proposed to categorize processes into normal or intrusive class. The KNNC calculates the similarity between each new process and every training process instance, and assumes that the processes belonging to the same class will cluster together in a vector space. It has good intrusion detection rates, but the detector is

computationally expensive for real-time implementation when the number of processes is large.

Spectral techniques have also been applied by researchers to network traffic analysis. Spectral techniques try to represent traffic data trace like packet arrival rate in frequency domain in order to find useful information. For example, [15] uses spectrum based methods to detect features with periodic patterns, such as bottlenecks in a link, the effects of the TCP windowing mechanism, and DoS attacks and traffic anomalies. However, accuracy of spectral methods degrades when periodicity in an attack is weak or nonexistent. Moreover, such methods are typically computationally expensive, and not suitable for high volume traffic analysis.

An intrusion detection system based on neural networks has been presented in [16]. While it exhibits good detection with low false alarm rates when tested with the DARPA 1999 IDS Evaluation data, using multilayer perceptrons requires relatively more processing time for intrusion detection.

SVM based classification approaches have enjoyed immense success and research interest over the last twenty years. However, the traditional SVM involves solving a quadratic program, which has computational complexity that is a quadratic order of the training sample size. The reference [2] compares the performance of a binary-class SVM method for differentiating between normal and intrusive traffic, with a neural network based approach. However, their algorithm only tests a very small subset of the DARPA 1999 IDS Evaluation data. When using a larger dataset, the time and storage complexity of their detection algorithm significantly increases. Moreover, the authors do not consider multi-class SVM techniques to classify the 22 different types of intrusions in the dataset. When SVM is adapted for multi-class classification, its performance degrades (cf. Section IV).

In this paper, we propose the use of ELM methods to classify binary and multi-class network traffic for intrusion detection. The performance of ELM in both binary-class and multi-class scenarios are investigated, and compared to SVM based classifiers. The advantages of ELM include its scalability and significant reductions in training time, as compared to SVM. Simulation results show that the proposed method can detect intrusions even in large datasets with short training and testing times. Most importantly, when using the basic ELM (cf. Section II-B2), our method can be implemented without any parameter tuning.

The rest of the paper is organized as follows. In section II, a brief introduction to binary-class and multi-class SVM is given. We also describe the ELM approach and highlight its differences with SVM. In Section III, 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 SVM-based techniques and ELM-based techniques are presented in section IV. In section V, we conclude and summarize our results.

## II. BACKGROUND

In this section, we briefly review both SVM and ELM approaches.

### A. SVM

1) *Binary-class SVM*: SVMs were introduced by Vapnik in [17]. Since then, SVM has been recognized as one of the best off-the-shelf supervised learning algorithms. SVMs classify data by determining a set of support vectors, which are members of the set of training inputs that outline a hyperplane in the feature space.

Given training data  $(\mathbf{x}_i, t_i)$ ,  $i = 1, \dots, N$ , where  $\mathbf{x}_i \in \mathbb{R}^n$ , and  $t_i \in \{1, -1\}$ , SVM solves the following problem

$$\begin{aligned} & \text{minimize} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i \\ & \text{such that} \quad y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \text{ for } i = 1, \dots, N, \\ & \quad \xi_i \geq 0, \text{ for } i = 1, \dots, N, \end{aligned} \quad (1)$$

where the training data  $\{\mathbf{x}_i : i = 1, \dots, N\}$  is mapped to a feature space by the function  $\phi$ , and  $C$  is a parameter that trades off the size of the separating margin with the training error.

Minimizing  $\|\mathbf{w}\|^2/2$  is equivalent to maximizing the distance  $2/\|\mathbf{w}\|^2$  between two different classes in the feature space. When the training data is not linearly separable, we use  $C \sum_{i=1}^N \xi_i$  to act as a penalty term to reduce the number of training errors. The parameter  $C$  controls the trade-off between maximizing the distance between two classes and minimizing the training error. When  $C$  is too large, SVM would tend to over-fit the training data; when  $C$  is too small, insufficient stress is placed on fitting the training data.

Based on the Karush-Kuhn-Tucker (KKT) theorem, we can develop the dual form of our problem (1) as

$$\begin{aligned} & \text{minimize} \quad \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N t_i t_j \alpha_i \alpha_j \phi(\mathbf{x}_i) \phi(\mathbf{x}_j) - \sum_{i=1}^N \alpha_i \\ & \text{such that} \quad \sum_{i=1}^N \alpha_i t_i = 0, \\ & \quad 0 \leq \alpha_i \leq C \text{ for } i = 1, \dots, N. \end{aligned} \quad (2)$$

By implementing the kernel trick, we are able to operate in a feature space without ever computing the coordinates of the data in that space like  $\phi(\mathbf{x}_i)$  and  $\phi(\mathbf{x}_j)$ , but rather by simply computing the inner products between the images of all pairs of data in the feature space  $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \phi(\mathbf{x}_j)$ . Moreover, kernel functions  $K(\mathbf{u}, \mathbf{v})$  can be user defined, which make SVM quite flexible for different kinds of applications. In this paper, the Gaussian radial basis function kernel is used for SVM, so  $K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma \|\mathbf{u} - \mathbf{v}\|^2)$ .

Then the output function of SVM can be written as

$$f(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^{N_s} \alpha_i t_i K(\mathbf{x}, \mathbf{x}_i) + b \right) \quad (3)$$

where  $N_s$  represents the number of support vectors  $\mathbf{x}_s$ .

2) *Multi-class SVM*: SVMs were initially designed for binary classification problems. However, some methods were proposed to extend it for multi-class classification. We describe two common approaches here [18]. The first approach is a one-against-one method, where several binary classifiers are constructed, with their result combined together. The second approach is a one-against-all method, where each class is trained against the aggregate of all the other classes. These multi-class SVM methods have been compared in [18], and it is shown that the one-against-one method is more suitable practically in terms of speed and accuracy.

Suppose that there are  $k$  classes. The one-against-one method constructs  $k(k-1)/2$  classifiers, where each is trained using data from two classes. If the training data comes from the  $i$ th and the  $j$ th classes, we solve the following binary classification problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} \|\mathbf{w}^{ij}\| + C \sum_{t: y_t \in \{i, j\}} \xi_t^{ij} \\ \text{such that} \quad & (\mathbf{w}^{ij})^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij}, \\ & \text{for all } (x_t, y_t) \text{ with } y_t = i, \\ & (\mathbf{w}^{ij})^T \phi(x_t) + b^{ij} \leq -1 + \xi_t^{ij}, \\ & \text{for all } (x_t, y_t) \text{ with } y_t = j, \\ & \xi_t^{ij} \geq 0 \text{ for all } t \text{ with } y_t \in \{i, j\}. \end{aligned} \quad (4)$$

The final class is determined by a voting strategy [19] known as ‘‘Max Wins’’. In the voting strategy for  $x$ , the vote for the  $i$ th class is incremented by one for every  $j$  such that  $\text{sgn}((\mathbf{w}^{ij})^T \phi(x) + b^{ij}) = 1$ . The final class is then taken to be the class with the maximum number of votes, where ties are broken arbitrarily.

## B. ELM

Traditionally, all the parameters in a feed-forward network are tuned, creating dependencies between different layers of parameters (weights and biases). Gradient descent based methods are commonly used in training a feed-forward neural network. However, gradient descent based learning methods are generally very slow due to improper learning steps, or may converge to a local minimum. In addition, many iterative learning steps may be required to obtain better learning performance.

A new learning paradigm known as ELM has been proposed in [3], [4], [5] for single-hidden layer feed-forward neural networks (SLFNs), which have either additive neurons or kernel based schemes. The essence of ELM is that, different from the traditional training of a feed-forward network, the hidden layer of a SLFN is not tuned. Rather, the parameters in the hidden layer are *randomly* chosen. One typical example is that of a SLFN with additive neurons. We randomly choose the input weights and the hidden neurons’ biases and choose the output weights of the hidden layer to minimize the training error. Input weights are the weights of the connections between inputs and hidden neurons. Output weights are the weights of the connections between hidden neurons and output nodes.

After the input weights and the hidden layer biases are chosen arbitrarily, SLFNs can be considered as a linear system, where the output weights can be analytically determined. The analysis in [3], [4], [5] shows that ELM tends to have good generalization performance and is very easy to implement.

1) *Function Approximation of SLFNs with Additive Neurons*: For  $N$  arbitrary distinct samples  $\{(\mathbf{x}_i, \mathbf{t}_i) : i = 1, \dots, N\}$ , where  $\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in \mathbb{R}^n$  and  $\mathbf{t}_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in \mathbb{R}^m$ , a SLFN with  $n$  inputs,  $m$  outputs,  $k$  hidden neurons and activation function  $g(x)$  is mathematically modeled as

$$\sum_{i=1}^k \beta_i g(\mathbf{w}_i^T \mathbf{x}_j + b_i) = \mathbf{o}_j, j = 1, \dots, N,$$

where  $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$  is the weight vector connecting the  $i$ th hidden neuron and the input neurons,  $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$  is the weight vector connecting the  $i$ th hidden neuron and the outputs, and  $b_i$  is the threshold of the  $i$ th hidden neuron.

The standard SLFN with  $k = N$  hidden neurons can approximate these  $N$  samples with zero error, i.e.,  $\sum_{j=1}^N \|\mathbf{o}_j - \mathbf{t}_j\| = 0$ , and there exist  $\{\beta_i, \mathbf{w}_i, b_i\}_{i=1}^k$  such that

$$\sum_{i=1}^k \beta_i g(\mathbf{w}_i^T \mathbf{x}_j + b_i) = \mathbf{t}_j, \text{ for } j = 1, \dots, N. \quad (5)$$

The  $N$  equations in (5) can be written compactly as

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

where

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_1^T \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_k^T \mathbf{x}_1 + b_k) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1^T \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_k^T \mathbf{x}_N + b_k) \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_k^T \end{bmatrix} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}$$

The matrix  $\mathbf{H}$  is called the hidden layer output matrix of the neural network; the  $i$ th column of  $\mathbf{H}$  is the  $i$ th hidden neuron output with respect to inputs  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ .

2) *Basic ELM*: In most applications, including intrusion detection in network traffic, the number of hidden neurons is much smaller than the number of distinct training samples, i.e.,  $k \ll N$ , and  $\mathbf{H}$  is a non-square matrix. There may not exist  $\{\mathbf{w}_i, b_i, \beta_i\}_{i=1}^k$  such that  $\mathbf{H}\beta = \mathbf{T}$ . In the ELM approach, the input weights  $\mathbf{w}_i$  and the hidden layer biases  $b_i$  of SLFNs are not tuned, but are assigned randomly and then fixed. This is equivalent to mapping the samples to a random feature space. Then, training a SLFN is equivalent to finding a least squares error solution  $\hat{\beta}$  of the linear system  $\mathbf{H}\beta = \mathbf{T}$ . The unique smallest norm least-squares solution of the linear system is

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}, \quad (6)$$

where  $\mathbf{H}^\dagger$  is the Moore-Penrose generalized inverse of hidden layer output matrix  $\mathbf{H}$ . In [5], it is shown that this method tends to reach good generalization performance. There are many ways of calculating the Moore-Penrose generalized inverse of a matrix such as the orthogonal projection method, iterative method and singular value decomposition [20]. In our case, the number of training samples is far bigger than the dimensionality of the feature space, and we add a positive value to the diagonal of  $\mathbf{H}^T \mathbf{H}$  so that a stable solution

$$\hat{\beta} = \left( \frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}, \quad (7)$$

where  $C$  is a positive constant is obtained. It has been shown numerically in [3] that the above solution has better generalization performance. Moreover, when the training dataset is very large, computational cost in (7) remains small as  $\mathbf{H}^T \mathbf{H}$  is a  $k \times k$  matrix.

The three main steps involved in the ELM algorithm can be summarized as follows

**ELM Algorithm:** Given a training set  $\{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \dots, N\}$ , activation function  $g(\mathbf{x})$ , and hidden neuron number  $k$ ,

- 1) Assign random input weights  $\mathbf{w}_i$  and biases  $b_i$ ,  $i = 1, \dots, N$ .
- 2) Calculate the hidden layer output matrix  $\mathbf{H}$ .
- 3) Calculate the output weights  $\beta$ :

$$\beta = \left( \frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}.$$

The output function is then given by  $f(\mathbf{x}) = g(\mathbf{x})\beta$ . Unlike iterative gradient descent based learning methods, ELM can handle non-differentiable activation functions, and does not have issues such as finding a suitable stopping criteria, learning rate, and learning epochs.

3) *Kernel Based ELM*: It should be noted that many nonlinear activation and kernel functions can be used in ELM. When the hidden layer feature mapping  $g(\mathbf{x})$  is unknown to users, one can define a kernel matrix  $\Omega_{ELM}$ , where the  $(i, j)$ th entry is  $K(\mathbf{x}_i, \mathbf{x}_j)$ , and  $K(\cdot, \cdot)$  is a positive semi-definite function. Then we can write the ELM output function as

$$f(\mathbf{x}) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left( \frac{\mathbf{I}}{C} + \Omega_{ELM} \right)^{-1} \mathbf{T}. \quad (8)$$

In this specific kernel implementation of ELM, the hidden layer feature mapping  $g(\mathbf{x})$  need not to be known to users, instead its corresponding kernel  $K(\cdot, \cdot)$  is given by users. In this paper, the Gaussian radial basis function kernel is used, so  $K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma \|\mathbf{u} - \mathbf{v}\|^2)$ .

### III. INTRUSION DETECTION USING ELM

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

TABLE I  
ATTACK TYPE

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

#### 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 [21].

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 privileges
- 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.

#### B. Intrusion Detection

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

- 1) Data pre-processing: a data processing script is used to convert the raw TCP/IP dump data into machine readable form.
- 2) Training phase: SVM 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.
- 3) Testing phase: SVM and ELM are used to predict the type of each data point in the testing dataset, and their performances are compared.

Both SVM and ELM can not 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

TABLE II  
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
service	network service on the destination	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port, 0 otherwise	discrete
wrong_fragment	number of “wrong” fragments	continuous
urgent	number of urgent packets	continuous

TABLE III  
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  
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

*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 [22].

LIBSVM [23] is used as the SVM implementation. After we transform the data to the format required by LIBSVM, scaling is implemented on the data. Then we use cross validation to find the best parameters  $C$  in (1) and (4), and  $\gamma$  in (3).

For implementing the ELM algorithm, MATLAB code from [24] is used. We transform the data to the required format with appropriate scaling. For basic ELM, we let the number of hidden neurons to be 400, and choose the value of  $C$  in (7) to be 2. For kernel based ELM, we use the same strategy used in SVM to find the best  $C$  and  $\gamma$  in (8).

The simulation of the two algorithms on all datasets are carried out using either MATLAB 2010a or gcc compiler

running on a machine with an Intel Core 2 Duo, 2.26GHz CPU and 4GB RAM.

## IV. SIMULATION RESULTS

### A. Binary classification

Gaussian radial basis kernel function is used in both SVM and kernel based ELM. It is known that the performance of SVM and kernel ELM closely depend on the combination of  $(C, \gamma)$ . To achieve good generalization performance, we choose the cost parameter  $C$  and kernel parameter  $\gamma$  appropriately. For each dataset, we choose  $C$  from  $[2^{-24}, 2^{-9}, 2^{-8}, \dots, 2^{25}]$  and  $\gamma$  from  $[2^{-24}, 2^{-9}, 2^{-8}, \dots, 2^{25}]$ . Therefore, for each dataset we try  $50 * 50 = 2500$  combinations. For basic ELM, we just choose the number of hidden neurons to be 400, and let the hidden neurons to be sigmoidal additive nodes. Because random values are assigned to some of the parameters in basic ELM, the output is not fixed. Hence, for basic ELM, we conduct 50 trials for each dataset and record its average testing accuracy and 95% confidence interval.

The datasets being tested are 2000, 4000, 8000 connection data chosen randomly from the dataset downloaded from the website [6]. We split them equally into training data and testing data. Then, we conduct a 10 fold cross validation on the training data and choose the parameter pair  $(C, \gamma)$  that reports the best cross validation rate. After that, we train the whole training set using the pair of  $(C, \gamma)$  and predict the testing data. After 50 trails for each dataset, average accuracy and its 95% confidence interval are recorded.

Simulation results including average testing accuracy and corresponding 95% confidence interval are given in Table V.

Figure 1 and Figure 2 show the time spent by SVM and ELMs when training and testing the same size of dataset. It can be seen that the training time and testing time spent by kernel based ELM increase sharply when the size of data increases. In comparison, basic ELM and SVM increase slowly when the number of data increases. Eventually, SVM starts consuming more time for both training and testing than basic ELM.

First, we compare basic ELM with SVM. In general, basic ELM always achieves comparable performance as SVM. As the dataset becomes larger, basic ELM outperforms SVM with faster learning speed. This means after training and deploying the basic ELM classifier, it can react to new observations much faster than a SVM classifier in real applications. It is obvious that low computational costs and fast response rate are critical for intrusion detection system. So when deploying intrusion detection system in high volume traffic environment, basic ELM would be a better choice compared to SVM. It should be noted that in order to obtain as good performance as possible for SVM, we have to spend a very long time tuning two parameters  $C$  and  $\gamma$ . In basic ELM, no parameters need to be tuned, and we only need to specify the number of hidden neurons. In [4], it has been investigated that the performance of ELM is not sensitive to the number of hidden neurons as long as it is large enough, and 1000 hidden neurons are good enough for many real world applications. Then, we compare kernel based ELM with SVM. Kernel based ELM achieves similar accuracy for all cases, as compared to SVM. However, kernel based ELM seems to take longer time for both training and testing. We have to keep in mind that C executable environment usually runs faster than MATLAB environment. We tested the dataset 4000/4000 using a SVM MATLAB package [25], it takes 17s for training, which is almost 10 times slower than SVM in C. Taking this into consideration, both basic ELM and kernel based ELM are faster than SVM.

### B. Multi-class classification

Binary classification can determine whether a connection is normal or not. But for network administrators, it is better to know what kind of attack the network experiences so that they can choose countermeasures respectively. So we use multi-class classification to determine which type of attack the abnormal connection belongs to.

The datasets used here are similar to the ones in binary cases, except that the labels are from 1 to 23 to represent normal data and 22 different attacks. SVM is extended for

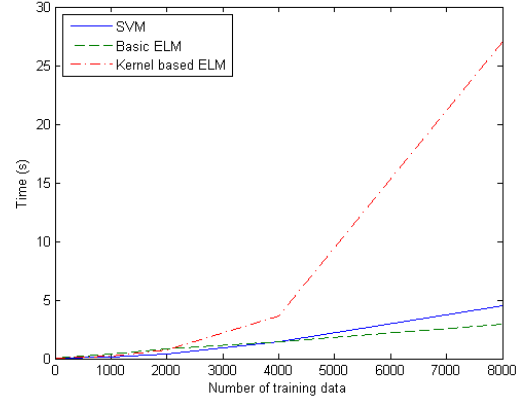


Fig. 1. Training time comparison

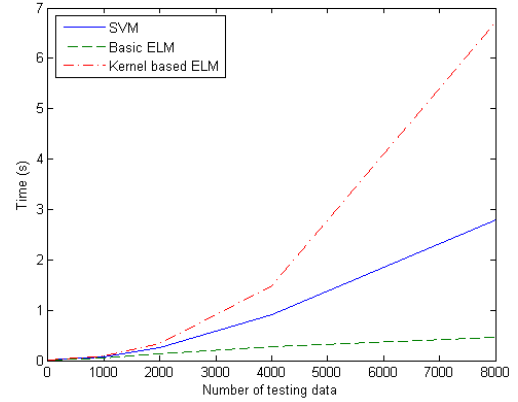


Fig. 2. Testing time comparison

multi-class classification using the "Max Win" voting strategy. Parameter choosing process, kernel function and number of hidden neurons are all the same as in binary classification case.

Simulation results including average testing accuracy and corresponding 95% confidence interval are given in Table VI.

A clear time consumption comparison can be seen from Figure 3 and Figure 4. From the results, we can conclude that basic ELM performs better than SVM in terms of speed. To increase accuracy, we can implement kernel based ELM. However, if we compare the results with the ones in binary classification cases, we can notice that basic ELM takes almost the same training time and testing time while SVM takes several times longer. Moreover, because of the performance degradation of SVM, kernel based ELM outperforms SVM in terms of speed without considering the fact that C program could run several times faster than MATLAB program. This shows that our proposed ELM methods have better scalability than SVM when classifying multi-class network traffic for intrusion detection.

TABLE V  
BINARY-CLASS PERFORMANCE COMPARISON RESULTS

Dataset Size	SVM		ELM		Kernel ELM	
Training/Testing	Accuracy (%)	95% Confidence Interval (%)	Accuracy (%)	95% Confidence Interval (%)	Accuracy (%)	95% Confidence Interval
1000/1000	99.15	99.12 - 99.17	99.33	99.15 - 99.51	99.12	99.05 - 99.25
2000/2000	99.43	99.40 - 99.45	99.07	98.90 - 99.24	99.27	99.25 - 98.28
4000/4000	99.77	99.76 - 98.78	99.58	99.50 - 99.66	99.63	99.61 - 99.65

TABLE VI  
MULTI-CLASS PERFORMANCE COMPARISON RESULTS

Dataset Size	SVM		ELM		Kernel ELM	
Training/Testing	Accuracy (%)	95% Confidence Interval (%)	Accuracy (%)	95% Confidence Interval (%)	Accuracy (%)	95% Confidence Interval
1000/1000	97.58	97.52 - 97.64	96.83	96.40 - 97.23	97.78	97.72 - 97.83
2000/2000	98.31	98.27 - 98.34	97.07	96.77 - 97.37	98.81	98.76 - 98.86
4000/4000	98.69	98.66 - 98.72	97.00	96.68 - 97.32	98.74	98.70 - 98.78

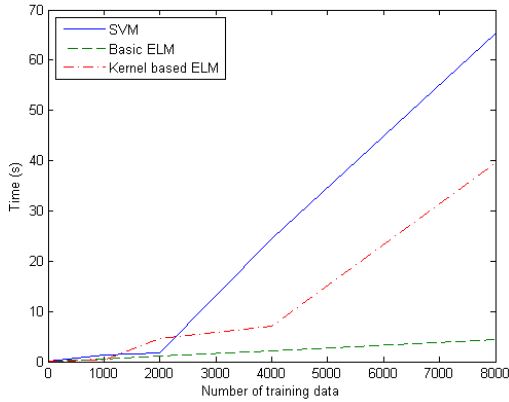


Fig. 3. Training time comparison

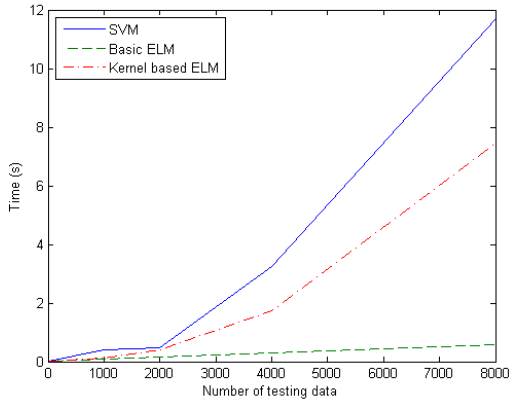


Fig. 4. Testing time comparison

## V. CONCLUSIONS

In this paper, we have proposed the use of both the basic and kernel based ELMs for intrusion detection in a computer network. For the basic ELM, one of the key advantages is that parameters in the learning process can be randomly assigned. This greatly reduces the time for training and provides good scalability. However, the basic ELM has slightly lower accuracy as compared to SVM techniques. To increase accuracy,

kernel based ELM can be implemented.

Whether to use SVM, basic ELM or kernel based 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, kernel based ELM would be preferred.

## REFERENCES

- [1] S. Bosworth and M. E. Kabay, *Computer Security Handbook*, 4th ed. Wiley, 2002.
- [2] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, May 2002, pp. 1702–1707.
- [3] G.-B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.
- [4] G.-B. Huang, H. M. Zhou, X. J. Ding, and R. Zhang, "Extreme learning machines for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2011.
- [5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Proceedings of IEEE International Joint Conference on Neural Networks*, vol. 2, 2004, pp. 985–990.
- [6] (1999) KDDCUP dataset. [Online]. Available: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>



- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 25, pp. 439–448, 2002.
- [12] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, 2001, pp. 5–8.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] W. W. Streilein, R. K. Cunningham, and S. E. Webster, "Improved detection 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.
- [17] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.
- [18] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.
- [19] J. H. Friedman, "Another approach to polychotomous classification," Department of Statistics, Stanford University, Tech. Rep., 1996.
- [20] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications*. John Wiley & Sons Inc, 1972.
- [21] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based 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.
- [22] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," *Bioinformatics*, vol. 1, no. 1, pp. 1–16, 2010.
- [23] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [24] (2004) ELM MATLAB code. [Online]. Available: [http://www.ntu.edu.sg/home/egbhuang/ELM\\_Codes.htm](http://www.ntu.edu.sg/home/egbhuang/ELM_Codes.htm)
- [25] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy, "SVM and kernel methods MATLAB toolbox," Perception Systmes et Information, INSA de Rouen, Rouen, France, 2005. [Online]. Available: <http://asi.insa-rouen.fr/enseignants/arakotom/toolbox/index.html>