Intrusion Detection of Multiple Attack Classes using a Deep Neural Net Ensemble

Simone A. Ludwig
North Dakota State University
Fargo, ND, USA
simone.ludwig@ndsu.edu

***Abstract*—An intrusion detection system (IDS) is a necessity to protect against network attacks. The system monitors the activity within a network of connected computers in order to analyze the activity for intrusive patterns. Should an 'attack' event happen, then the system has to respond accordingly. Different machine learning techniques have been proposed in the past roughly falling into two categories namely clustering algorithms and classification algorithms. In this paper, the IDS is designed with a neural network ensemble method to classify the different attacks. The neural network ensemble method comprises autoencoder, deep belief neural network, deep neural network, and an extreme learning machine. The NSL-KDD data set is used to measure the detection rate and false alarm rate of the implemented neural network ensemble method. The detection rate and false alarm rate are the two important measure for IDSs, however, several other measures are also reported on such as confusion matrix, classification accuracy, and AUC (area under curve).**

I. Introduction

Cybersecurity is a very important aspect when it comes to protect networks, computers, and data from attacks and unauthorized access. The term cybersecurity encompasses different technologies, processes and practices involved. There are different categories that include application security, information security, network security, disaster recovery, operational security and end-user education. One of the challenges that computing systems and network systems face is the evolving nature of threats. In the past, this challenge was dealt with by protecting the most crucial system components from the biggest known threats. However, this leaves the less important portions of a system unprotected and vulnerable to inevitable threats. Since this is not an approach to follow, new ways, methodologies, and technologies need to be invented in order to protect systems better [1].

Network based attacks have been increasing over the past several years both in terms of frequency and severity. One of the reasons is that more and more technologies are using communication networks, in particular wireless communication systems. Thus, network security is a high priority to protect against potential attacks, which is accomplished by monitoring the network traffic as well as making use of a defense system and mechanism. There are different attacks on communication network system and these are flooding, distributed denial-of-service, surfing, vulnerabilities, etc. Intrusion detection systems are systems that deal with the recognition of normal behavior on the network versus suspicious behavior on the network, in particular, with IDS detection intruders' actions that threaten the integrity of the computer system [2].

Today's system and data intrusions are quite sophisticated and thus require a multi-tiered approach [5]. This implies that companies that secure their networks often use several technologies to prevent cyberattacks and intrusions. There is a variety of tools and methodologies available, however, the two common elements to a secure network configuration is the firewall and the intrusion detection system.

There are two types of IDS namely host based and network based. The host based system sits on a particular host and watches for potential attacks. A network based system looks at the network traffic in real time in order to detect intrusion patters in the network [3]. A security model should employ both a host based and a network based solution since both have advantages and disadvantages. A drawback for the host based solution is that resources are taken away from the host in order to power the host based protection system. Furthermore, host based solutions are reactive and thus can only respond after an attack has actually occurred. Network based solutions are usually installed in the form of a hardware appliance, and thus do not need to use the system resources. This solution tends to be more costly, however, the installation process is easy.

The detection of an intrusion is based on the difference between normal operating and intrusion behavior and thus divides into anomaly detection and misuse detection [4]. Anomalies are detected by analyzing features of normal behavior on the network and identifying when anomalies occur. The advantages of anomaly detection is that unknown intrusion types can be detected, however, it might result in a high rate of false positives. On the other hand, misuse detection is to analyze attack behavior by establishing templates of attack characteristics, which is used to determine the attacks. Misuse detection has the characteristics of high accuracy and fast speed, however, the templates need to be updated very frequently otherwise this method is not effective.

The aim of this paper is to analyze and classify the NSL-KDD data set [6] to distinguish between normal and intrusion behavior. The NSL-KDD data set is a 'cleaned' data set of the KDD data set [7] created in 1999. The NSL-KDD data set consists of selected records from the KDD99 data set by removing redundancy and duplicates. The approach chosen in this paper is based on a deep learning ensemble method whereby related deep learning models are implemented and evaluated.

The paper continues as follows. Section II describes related work in the area of intrusion detection systems. In Section III, the proposed approach is described. Section IV contains the experiments conducted as well as the results and findings. The conclusion and future work is given in Section V.

II. Related Work

Many different systems have been built to monitor data flow in networks in order to prevent temporary and permanent damages caused by unauthorized access. Unfortunately, these systems cannot detect all types of intrusions as attack permutations are occurring over time. Machine learning algorithms have been applied to classify normal and anomalous behavior on the network.

K-means and K-nearest neighbor algorithms were used in [8], [9] to perform the classification task. In this approach a centroid function is used to choose the average and closest grouping of new instances in order to group similar training examples together.

Another approach used for IDS are support vector machines (SVM). SVM divide the dimensional space with a smaller dimensional hyperplane [10], [11]. In [12], SVM was used to automate feature selection, which is a preprocessing step that usually is applied in the area of data mining, thus improving the classification rate. The so-called CSV-ISVM algorithm was proposed in [13]. This algorithm uses incremental SVM in order to select candidate support vectors and shows advantages in real-time network intrusion detection.

Threshold-based anomaly detection also known as signature matching has been widely applied to model network traffic. For example, [14] discussed that traditional network-based profile models are not sufficient enough to satisfy user profiles in the environment nowadays. Thus, a genetic algorithm approach was used to find signatures of pattern detection rules via permutations of parent signatures. Another approach [15] proposed a core-plus-module framework (STAT) that is based on state transition analysis technique in order to tailor the design of an IDS to specific traffic types and environments. Other research compared the genetic algorithm approach with other approaches such as Naive bayes and K-nearest neighbor [10].

Deep neural networks (DNN) have seen quite an uptake in recent years. In particular, significant breakthroughs have been achieved for tasks such as image recognition, speech recognition, text recognition, and language translation. Deep learning methods encompasses many different neural network models such as deep belief neural networks, convolutional neural networks, autoencoders, recurrent neural networks, etc. All these deep neural network approaches have been developed each serving a different purpose. For example, deep belief networks (DBN) [16] was applied to image, text and voice learning tasks. A DBN is formed by stacking several restricted boltzmann machines (RBM) [17] that serve as multiple processing layers in order to learn the representation and features inherent in the data with multiple levels of abstraction.

Deep learning methods have been applied to the intrusion detection as follows. Several DBN approaches have been applied to the intrusion detection task. For example, a DBN approach is proposed in [18], where a two-layer RBM is used to train the network in an unsupervised fashion followed by a feedforward layer whereby backpropagation is used to train this layer in a supervised fashion. The authors report on the classification accuracy based on the test set comparing their approach with SVM, and a hybrid version of DBN with SVM.

In [19], the DBN algorithm was implemented and applied on the NSL-KDD data set. The measures used were classification accuracy, TP (true positives), FP (false positives), TN (true negatives), and FN (false negatives).

A hybrid approach based on autoencoder and DBN is implemented in [20]. The autoencoder learning method is used to reduce the dimensionality of the data, which allows to convert high-dimensional data to a low-dimensional transformation with nonlinear mapping, and thus extracting the main features of the data. After this first step, DBN learning will be applied to detect anomalies. The DBN consists of multilayer RBMs followed by a feedforward layer. First, the RBMs are trained using an unsupervised approach followed by supervised training with the feedforward layer. The measures reported are TPR (true positive rate), FPR (false positive rate), accuracy and CPU time.

An improved version of DBM is proposed in [21]. Since the fine-tuning method of DBN is very time-consuming and suffers from the possibility of only reaching a local optimum, ELM (Extreme Learning Machines) was applied during the training process in order to improve the accuracy as well as the efficiency. The improved DBN was compared with the normal DBN and achieved an improvement of 0.6% in the detection rate by roughly reducing the execution time by half. Besides the detection rate and the execution time no other measures were reported on.

An accelerated DNN is proposed in [22]. A parallel version of the DNN is used to accelerate the training phase, which is a very time consuming task. The training phase consists of several forward passes and backward passes. The input is applied to the input nodes and each layer computes the output on a layer-by-layer basis; this step is the forward pass. The backward pass first calculates the error between the actual output and the desired output and then backpropagates this error by adjusting the weights in each layer as to reduce the result during the next iteration. Since gradient calculations are involved during the backpropagation process and depending on the depth of the network, long training times are inevitable.

A deep learning approach for flow-based anomaly detection in a Software Defined Networking (SDN) environment is introduced in [23]. The DNN model is built using just six basic features, ones that can be easily obtained in an SDN environment. Different learning rates were experimented with resulting in the best accuracy when the learning rate is set to 0.0001.

It is important to note that some of the deep learning methods listed as related work were applied to the KDD data set and others to the NSL-KDD data set.

III. Proposed Approach

Ensemble learning is an approach where several classifiers are trained and their results are fused together in order to separate the different classes. Ensemble techniques are also known as multiple classifier systems, or just ensemble systems. In this paper, several deep neural network approaches are used and their results are fused together in order to distinguish between normal and attack behavior of a network. Ensemble approaches have lead to very promising results, usually achieving a higher classification accuracy than single classifier approaches alone.

A. Basic Concepts of Ensemble Learning

The concept of ensemble learning was first introduced in 1979 [27], which proposed using an ensemble system in a divide-and-conquer fashion, whereby the feature space was partitioned using two or more classifiers. More than 10 years later, another ensemble system was introduced showing that the generalization performance of similar neural network configurations can be improved using ensembles by introducing the variance reduction property [28]. However, research in [29] placed ensemble systems at the center of machine learning research. This was achieved by proving that a strong classifier in the probably approximately correct sense can be generated by combining weak classifiers through a procedure called boosting.

The following paragraphs describe the details of ensemble learning.

Definition 1 Let $\Omega = \{\omega_1, \omega_2, ..., \omega_M\}$ be a set of class labels, and a function $D : \mathbb{R}^n \rightarrow \Omega$ is called a classifier with the feature vector $X = (X_1, X_2, ..., X_n) \in \mathbb{R}^n$.

Definition 2 Let $h_1, h_2, ..., h_M, h_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, ..., M$ be discriminator functions that correspond to the class labels $\omega_1, \omega_2, ..., \omega_M$, respectively. Then, the classifier D belonging to this discriminator function is:

$$D(X) = \omega_{j*} \Leftrightarrow h_{j*}(X) = \max_{j=1}^{M}(h_j(X)) \tag{1}$$

for all $X \in \mathbb{R}^n$.

Definition 3 Let $D_1, D_2, ..., D_L$ be classifiers and the majority voting ensemble classifier $D_{maj} : \mathbb{R}^n \rightarrow \Omega$ is obtained from these classifiers:

$$\begin{aligned} D_{maj}(X) = \omega_{i*} \Leftrightarrow & |\{j : D_j(X) = \omega_{i*}, j = 1, ..., M\}| \\ & = \max_{i=1}^{M} |\{j : D_j(X) = \omega_i, j = 1, ..., M\}| \end{aligned} \tag{2}$$

Definition 4 Let $D_1, D_2, ..., D_L$ be classifiers and $\beta = (\beta_1, \beta_2, ..., \beta_L) \in \mathbb{R}^L$ be a weight vector assigned to the classifiers. Then, the weighted majority voting ensemble classifier $D_{wmaj} : \mathbb{R} \rightarrow \Omega$ is defined by:

$$D_{wmaj}(X) = \omega_{i*} \Leftrightarrow \sum_{\substack{j=1 \\ D_j(X)=\omega_{i*}}}^{L} \beta_j = \max_{i=1}^{M} \left(\sum_{\substack{j=1 \\ D_j(X)=\omega_i}}^{L} \beta_j \right) \tag{3}$$

B. Proposed Ensemble Deep Neural Network Classifiers

1) Autoencoder (AE): Autoencoders are composed of an input, a hidden and an output layer. The output received by the output layer is a reconstruction of the input after the input has been ‘squished’ through a smaller hidden layer. Thus, autoencoders offer dimensionality reduction and basically a compression similar to PCA (principle component analysis). The features that are extracted via the hidden layer can be used to train a feedforward layer, thus, removing the output layer from the autoencoder, i.e., use the hidden layer as input features for the classification or another autoencoder. The network is being trained by training each autoencoder using unsupervised data followed by the fine-tuning step whereby the last layer is trained using supervised data.

The autoencoder architecture that was used for the experiments follows the implementation in [22] with two autoencoders of size 20 and 10, respectively, followed by a fully connected layer of size 5.

2) Deep Belief Neural Network (DBN): The main idea of a DBN is to train a sequence of RBMs by defining the probability distributions over the hidden layer in order to estimate the probability of the generating visible layer. This is done by learning certain parameters via random sampling to learn the model. The cascade of RBMs allows the hidden vectors of one RBM to be the input for the next RBM and so on.

The following general rules apply:

- If the number of hidden units in the top layer is above a certain threshold, then the performance converges to a certain accuracy.
- The performance tends to increase with the training of each RBM.
- The performance decreases as the number of layers increases.

The stacking of RBM layers can be seen as a feature extraction method. The training of the RBM layers is done without the labels, and thus is unsupervised training. The last layer of the network is a fully connected layer and is trained with labeled data.

The DBN architecture that was used for the experiments consisted of 2 RBM layers with 20 and 15 nodes, respectively, followed by a 15-node fully connected layer.

3) Deep Neural Network (DNN): A DNN consists of an input, several hidden layers, and an output layer. The network is trained using backpropagation in order to minimize the error between the actual output and the desired output.

For the experiments, a network with two hidden layers of size 25 and 20 was implemented. Standard backpropagation has two shortcomings, the first is that a fixed learning rate is used, and the second that the search can get stuck in local minima. Thus, the Adam optimizer was introduced [31]. Adam (Adaptive Moment Estimation) uses separate learning rates for each weight as well as an exponentially decaying average of previous gradients that contributes to a better training algorithm.

4) Extreme Learning Machine RBM (ELM): Extreme learning machine is a learning algorithm that is used to train a single hidden layer neural network [32]. The input weights and hidden biases are randomly generated. The output weights are calculated by the regularized least square method, thus, resulting in a simple deterministic solution. Since there are no iterations and/or parameter tuning involved as in backpropagation based neural networks, the method is very fast. Moreover, the regularized least squares computations of the ELM are much faster than solving the quadratic programming problem as is the case in SVM. Several studies have shown that ELM is much more efficient at the same time achieving a higher generalization performance than NN or SVM [33]. Thus, ELM has become a significant research topic in particular in the machine learning domain.

The ELM architecture that was used for the experiments follows the implementation in [21]. The network structure used is a 110-90-50-25 layer structure trained with a maximum number of iterations set to 300.

IV. Experiments and Results

A. NSL-KDD Data set

The history behind the NSL-KDD data set is the following. In 1998, the MIT Lincoln Lab held a DARPA-sponsored IDS event which simulated an attack scenario to the Air-Force base with an repeat event one year later [24]. During these events improvements were suggested by the computer security community. The DARPA data set [25] consists of host and network data files recorded during a seven week time period. The first two weeks were attack-free whereas the remaining weeks contained also attack data. In order to make it easier for the data mining community to apply machine learning techniques another data set, the KDD99 data set, was created by preprocessing the data and extracting the relevant features. The output classes are divided into 5 categories namely DOS (denial of service), probe, R2L (Root to local), U2R (user to root), and normal. The KDD99 data is still in use today and has been extensively studied. However, several researchers have pointed out various shortcomings [26]:

- Imbalance data set; 80% is attack data.
- U2R and R2L attacks are rare.
- Duplicate records in both training and testing data set.

Thus, these shortcomings were alleviated with the introduction of the NSL-KDD data set. The NSL-KDD data set contains 41 features that are either continuous or discrete. The features of the data set are grouped into four categories:

- Basic features that are derived from the packet headers without inspecting the payload information.
- Content features for which domain knowledge is used to assess the payload of the original TCP packets.
- Time-based traffic features that are extracted to capture the properties during a 2-second time window.
- Host-based traffic features that are extracted to assess attacks that span intervals of longer than 2-second time periods.

The outcome is given as either normal or a specific attack type. The simulated attacks fall into one of the following categories:

- Denial of Service (DoS): this is an attack that occupies either a computing or memory resource so that no other requests can be serviced.
- Probing: the attacker scans the network to gather information in order to exploit the systems; an example is port scanning.
- Remote to Local (R2L): attacker sends a packet to the network by exploiting some vulnerability in order to gain local access; an example is password guessing.
- User to root (U2R): the attacker accesses a normal user account and exploits vulnerability to gain root access to the system; an example is a buffer overflow attack.

There are different attack types that map to the different attack classes; these are outlined in Table I.

TABLE I
MAPPING OF ATTACK TYPES TO ATTACK CLASSES

Attack class	Attack types
DoS	back, land, neptune, pod, smurf, teardrop, mailbomb, apache2, processtable, udpstorm
Probe	ipsweep, nmap, portsweep, satan, mscan, saint
R2L	ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster, sendmail, named, snmpgetattack, snmpguess, xlock, xsnoop, worm
U2R	buffer_overflow, loadmodule, perl, rootkit, httptunnel, ps, sqlattack, xterm

Table II shows the number of training and testing records and their distribution. There are $125,973$ records in the training data set, and $22,543$ records in the testing data set.

TABLE II
DESTRIBUTION OF TRAINING AND TESTING RECORDS

	Normal	DoS	Probe	U2R	R2L	Total
Train	67,343	45,927	11,656	52	995	125,973
Test	9,711	7,458	2,421	200	2,754	22,543

B. Evaluation Measures

The performance measures used to evaluate the ensemble classifier are the following:

- Confusion matrix: contains the number of actual and predicted classifications achieved by the classifier.
- False positives (FP): defines the number of detected attacks that are actual normal behavior.
- False negatives (FN): are the wrong predictions whereby instances that are attacks are classified as normal.
- True positive (TP): instances that are correctly classified as normal.
- True negatives (TN): attack instances that are correctly classified.

- Accuracy or True positive rate (TPR): percentage of correct predictions compared to all predictions.
- Area Under Curve (AUC): describes the curve between TPR and FPR and the area under the curve; FPR is calculated as
$$\frac{FP}{TN+FN}. \tag{4}$$
- False alarm rate: is calculated as
$$\frac{FP}{TN}. \tag{5}$$
- Detection rate: is calculated as
$$\frac{TN-FN}{TN}. \tag{6}$$
- Precision (P): is calculated as
$$\frac{TP}{TP+FP}. \tag{7}$$
- Recall (R): is the proportion of instances belonging to the positive class that are correctly predicted as positive and calculated as
$$\frac{TP}{TP+FN}. \tag{8}$$
- F1-score: also known as F-score or F-measure considers both precision and recall to compute the score; it is computed as
$$2\times\frac{P\times R}{P+R}. \tag{9}$$

C. Results

The following are the results of the run achieved during the training phase. Table III shows the confusion matrix whereby $13,313$ and $11,769$ are correctly classified as normal and attacks, respectively.

TABLE III
CONFUSION MATRIX OF TRAINING PHASE

	normal	attack
normal	13313	15
attack	36	11769

The different metric scores during the training phase are listed in Table IV. The main measure for IDSs are the false alarm rate, which should be low, and the detection rate, which should be high. Results of 0.11% and 99.69% are achieved, respectively. Other values of importance are the classification accuracy and AUC with both achieving 99.79%.

TABLE IV
VARIOUS METRIC SCORES FOR TRAINING PHASE

Accuracy	0.997971
AUC	0.997912
False alarm rate	0.001125
Detection rate	0.996950
F1 score	0.997838

TABLE V
PRECISION, RECALL, F1-SCORE AND SUPPORT FOR TRAINING PHASE

	Precision	Recall	F1-score	Support
0.00	1.00	1.00	1.00	13328
1.00	1.00	1.00	1.00	11805
avg / total	1.00	1.00	1.00	25133

Precision, recall, F1-score and support results are given in Table V.

The following are the results of the run achieved during the testing phase when the trained model is applied to the test data. Table VI displays the confusion matrix. $8,282$ and $12,570$ records were correctly classified as normal and attack, respectively, with only $1,692$ records being misclassified.

TABLE VI
CONFUSION MATRIX OF TESTING PHASE

	normal	attack
normal	8282	1429
attack	263	12570

A false alarm rate of 14.72%, and a detection rate of 97.95% were achieved on the test data set as shown in Table VII. A classification accuracy of 92.50% was achieved with an AUC of 91.62%. The F1-score resulted in 93.69%.

TABLE VII
VARIOUS METRIC SCORES FOR TESTING PHASE

Accuracy	0.924947
AUC	0.916177
False alarm rate	0.147153
Detection rate	0.979506
F1 score	0.936941

Precision, recall, F1-score and support results are given in Table VIII.

TABLE VIII
PRECISION, RECALL, F1-SCORE AND SUPPORT FOR TESTING PHASE

	Precision	Recall	F1-score	Support
0.00	0.97	0.85	0.91	9711
1.00	0.90	0.98	0.94	12833
avg / total	0.93	0.92	0.92	22544

D. Comparison with Results of Related Work

In order to compare the results obtained by our proposed ensemble method, different related work applying deep neural network approaches to the NSL-KDD data set have been used. However, please note that related approaches mostly reported only on one or a few of the measures. Thus, Table IX is rather sparse with many measures missing from the comparison approaches.

The comparison approaches are DNN [30], DBN [18], Autoencoder DNN [22], ELM-DBN [21], and DNN2 [23].

In terms of classification accuracy, only DBN, autoencoder and DNN2 have published results. These are 97.45% achieved by the DBN approach, and 91.7% achieved by DNN2. For the autoencoder DNN, listed as separate values of 96.5% for the normal category and 97.5% for the attack category. With regards to this measure our proposed ensemble method falls short with achieving only 92.49% compared to DBN and autoencoder. However, in the area of IDSs the most important measures are the detection rate and false alarm rate. None of the comparison approach has considered the false alarm rate, thus it cannot be compared. The detection rate on the other hand was only measured by the ELM-DBM approach achieving 91.8% whereas our method achieved 97.95%. In terms of precision, recall, and F-measure the only approaches that can be compared with are DNN and DNN2. The DNN approach achieved values of 96%, 64% and 77%, and the DNN2 method obtained 83%, 75%, and 72%, respectively. Please note that the results of DNN were obtained from the figures given in the paper. Our proposed method achieved values of 93%, 92% and 92% for precision, recall and f-measure, respectively. Overall, our method obtained better results with the exception of the classification accuracy.

V. Conclusion

IDSs are systems that are designed to defend against network attacks. Different methods have been proposed in the past and many of these systems implement a data mining approach. The data mining approaches can be classified into clustering and classification approaches. In this paper, a classification model using deep neural networks was studied. In particular, the NSL-KDD data set was investigated using a deep neural network ensemble technique. The ensemble technique comprised different deep neural network techniques such as autoencoder, deep belief neural network, deep neural network, and an extreme learning machine. The most important measures for IDSs are the detection rate and false alarm rate. The detection rate is the fraction of the difference between the attack instances that are correctly classified and the instances that are falsely classified as normal, and attack instances that are correctly classified. The false alarm rate is the fraction of detected attacks that are actual normal and attack instances that are correctly classified. Other measures considered are classification accuracy, AUC, precision, recall, and F-measure.

The results of the proposed ensemble method was compared with related deep learning methods that were applied on the NSL-KDD data set. The comparison approaches were DNN, DBN, autoencoder DNN, and ELM-DBN. Unfortunately, one shortcoming of the related work methods is that most of them only reported on one or a few of the measures. In particular, the relevant measures of false alarm rate and detection rate were hardly measured.

In terms of classification accuracy, DBN and autoencoder achieved approximately 5% higher accuracy value than our proposed ensemble method. However, in terms of the other more relevant measures our proposed ensemble method outperforms the other measures quite significantly. For example, for the detection rate our method achieved 97.95% whereas ELM-DBM achieved only 91.8%. Furthermore, the precision, recall, and F-measure are also significantly higher than the comparison approaches DNN and DNN2. Values of 93%, 92% and 92% were obtained by our method, and 96%, 64% and 77% were achieved by the DNN method, and 83%, 75%, and 72% were obtained by DNN2. In summary, our proposed ensemble method obtained better results than the comparison approach in terms of the important measures of IDSs with the exception of the classification accuracy.

Future research will include more deep learning methods such as clustering methods called deep embedded clustering, deep self-organizing maps, etc. Also, other machine learning methods can be used and added to the ensemble. It would also be interesting to investigate the execution time when run on a CPU versus a GPU, in particular with regards to big data problems.

References

[1] Cybersecurity, http://whatis.techtarget.com/definition/cybersecurity, 2017.

[2] W. Stallings, Network security essentials: applications and standards, 5th edition, Pearson, 2013.

[3] K. Scarfone and P. Mell, Guide to Intrusion Detection and Prevention Systems Recommendations (IDPS), National Institute of Standards and Technology, NIST Spec. Publ. 800-97, 2007.

[4] B. C. Rhodes, J. A. Mahaffey, J. D. Cannady, Multiple self-organizing maps for intrusion detection, 23rd national information systems security conference, 2000.

[5] Top Free Network-Based Intrusion Detection Systems (IDS) for the Enterprise, https://www.upguard.com/articles/top-free-network-based-intrusion-detection-systems-ids-for-the-enterprise, 2015.

[6] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, A Detailed Analysis of the KDD CUP 99 Data Set, IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[7] W. Lee, S. J. Stolfo, A framework for constructing features and models for intrusion detection systems, ACM Transactions on Information and System Security 3:227-261, 2000.

[8] I. Chairunnisa, Lukas, and H. D. Widiputra. Clustering base intrusion detection for network profiling using k-means, ecm and k-nearest neighbor algorithms. In Konferensi Nasional Sistem dan Informatika, 2009.

[9] S. Zanero and S. M. Savaresi. Unsupervised learning techniques for an intrusion detection system. In SAC '04: Proceedings of the 2004 ACM symposium on Applied computing, pages 412-419, New York, NY, USA, 2004.

[10] A. Ali, A. Saleh, and T. Ramdan. Multilayer perceptrons networks for an intelligent adaptive intrusion detection system. International Journal of Computer Science and Network Security, 10(2), 2010.

[11] N. Gornitz, M. Kloft, K. Rieck, and U. Brefeld. Active learning for network intrusion detection. In 2nd ACM workshop on security and artificial intelligence, pages 47-54, 2009.

[12] M. Kloft, U. Brefeld, P. Dussel, C. Gehl, and P. Laskov. Automatic feature selection for anomaly detection. In AISEC 2008, pages 71-76, 2008.

[13] R. Chitrakar and C. Huang, Selection of candidate support vectors in incremental SVM for network intrusion detection, Computers & Security, vol 45, pp. 231-241, 2014.

[14] F. Giroire, J. Chandrashekar, G. Iannaccone, K. Papagiannaki, E. M. Schooler, and N. Taft. The cubicle vs. the coffee shop: Behavioral modes in enterprise end-users. In Proceedings of the 2008 Passive and Active Measurement Conference, pages 202-211, Springer, 2008.

[15] M. Pillai, J. Eloff, and H. Venter. An approach to implement a network intrusion detection system using genetic algorithms. In Proceedings of South African Institute of Computer Scientists and Information Technologists, pages 221-228, Western Cape, South Africa, 2004.

[16] G. E. Hinton, S. Osindero, and Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural computation, vol. 18, pp. 1527-1554, 2006.

TABLE IX
COMPARISON OF PROPOSED APPROACH WITH OTHER APPROACHES BASED ON THE AVAILABLE MEASURES

	Accuracy (%)	Detect rate (%)	Precision (%)	Recall (%)	F-measure (%)
DNN [30]	-	-	96.0 (est. from figure)	64.0 (est. from figure)	77.0 (est. from figure)
DBN [18]	97.5 (40% training set)	-	-	-	-
Autoencoder [22]	96.5 (normal) 97.5 (attack)	-	-	-	-
ELM-DBN [21]	-	91.8	-	-	-
DNN2 [23]	91.7	-	83.0	75.0	74.0
Proposed approach	92.5	97.9	93.0	92.0	92.0

[17] R. Salakhutdinov and G. E. Hinton, Deep boltzmann machines, International conference on artificial intelligence and statistics, 2009.

[18] M. Z. Alom, V. Bontupalli and T. M. Taha, Intrusion detection using deep belief networks, 2015 National Aerospace and Electronics Conference (NAECON), Dayton, OH, 2015.

[19] K. Alrawashdeh and C. Purdy, Toward an Online Anomaly Intrusion Detection System Based on Deep Learning, 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Anaheim, CA, 2016.

[20] Y. Li, R. Ma, R. Jiao, A Hybrid Malicious Code Detection Method based on Deep Learning, International Journal of Security and Its Applications, vol. 9, no. 5, 2015.

[21] Y. Liu and X. Zhang, Intrusion Detection Based on IDBM, 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, Auckland, 2016.

[22] S. Potluri and C. Diedrich, Accelerated deep neural networks for enhanced Intrusion Detection System, 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, 2016.

[23] T. A. Tang, L. Mhamdi, D. McLernon, S. A. Raza Zaidi, M. Ghogho, Deep learning approach for Network Intrusion Detection in Software Defined Networking, 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 2016.

[24] A. Ozgur, H. Erdem, A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015 (Version 1), PeerJ Preprints, 2016.

[25] DARPA Intrusion Detection Data Sets, https://www.ll.mit.edu/ideval/data/, 1998.

[26] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, Proceedings of the 2010 IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, DC, USA, 2010.

[27] B. V. Dasarathy and B. V. Sheela, Composite classifier system design: concepts and methodology, Proceedings of the IEEE, vol. 67, no. 5, pp. 708-713, 1979.

[28] L. K. Hansen and P. Salamon, Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993-1001, 1990.

[29] R. E. Schapire, The Strength of Weak Learnability, Machine Learning, vol. 5, no. 2, pp. 197-227, 1990.

[30] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, A Deep Learning Approach for Network Intrusion Detection System. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies, Brussels, Belgium, 2016.

[31] D. P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2014.

[32] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing, vol. 70, no. 13, pp. 489-501, 2006.

[33] G.-B. Huang, L. Chen, and C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks, vol. 17, no. 4, pp. 879-892, 2006.