sciendo

JAISCR, 2019, Vol. 9, No. 3, pp. 177– 188
10.2478/jaiscr-2019-0002

APPLYING A NEURAL NETWORK ENSEMBLE TO INTRUSION DETECTION

Simone A. Ludwig

Department of Computer Science, North Dakota State University, Fargo, ND, USA

E-mail: simone.ludwig@ndsu.edu

Submitted: 27st July 2018; Accepted: 19th November 2018

Abstract

An intrusion detection system (IDS) is an important feature to employ in order to protect a system against network attacks. An IDS monitors the activity within a network of connected computers as to analyze the activity of intrusive patterns. In the event of an 'attack', the system has to respond appropriately. Different machine learning techniques have been applied in the past. These techniques fall either into the clustering or the classification category. In this paper, the classification method is used whereby a neural network ensemble method is employed to classify the different types of attacks. The neural network ensemble method consists of an autoencoder, a deep belief neural network, a deep neural network, and an extreme learning machine. The data used for the investigation is the NSL-KDD data set. In particular, the detection rate and false alarm rate among other measures (confusion matrix, classification accuracy, and AUC) of the implemented neural network ensemble are evaluated.

Keywords: Ensemble learning, Deep Neural Networks, NSL-KDD data set

1 Introduction

Cyber security is becoming more and more important when it comes to protecting networks, computers, and data from attacks and unauthorized access. The term cyber security encompasses many things such as different technologies, processes, and practices. The different categories include application security, information security, network security, disaster recovery, operational security and end-user education. One of the challenges of computing systems and network systems 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 is not good enough since it leaves the less important portions of a system unprotected and vulnerable to possible threats. Thus, new ways, methodologies, and technologies need to be designed and invented in order to protect systems better [1].

Network-based attacks have been increasing over the past years; both in terms of frequency and severity. One reason is that more and more technologies use communication networks, in particular, wireless communication systems. Therefore, network security has to be a high priority to protect against potential attacks. This is accomplished by monitoring the network traffic as well as the usage of a defense system. There are different attacks on communication network systems, which are: flooding, distributed denial-of-service, surfing, vulnerabilities, etc. Intrusion detection systems are systems that are designed to deal with the recognition of normal behavior on the network versus abnormal behavior on the network, in particular, when

actions are recognized that threaten the integrity of the computer system [2].

Today's system and data intrusions are quite sophisticated. Thus, these systems require a multi-tiered approach [3], which implies that companies that secure their networks often use several technologies to prevent cyber attacks and intrusions. There is a variety of tools and methodologies available, however, the two fundamental elements to a secure network configuration is the firewall and the intrusion detection system.

IDS (intrusion detection system) are either host-based or network-based. The host-based system sits on a particular host and watches for potential attacks whereas a network-based system looks at the network traffic in real time in order to detect intrusive patterns in the network [4]. Ideally, a security model should employ both a host-based and a network-based solution since both of these have advantages and disadvantages. One drawback for a host-based solution is that resources are taken away from the host in order to enable the host-based protection system. In addition, host-based solutions are reactive, and thus, can only respond after an attack has actually occurred, which is undesirable. On the other side, 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 much easier compared to the host-based solution.

An intrusion is detected by observing the difference between normal operating and intrusion behavior, and thus, divides into anomaly detection and misuse detection [5]. Anomalies are detected by analyzing features of normal behavior on the network and identifying anomalies. The advantage of anomaly detection is that unknown intrusion types can be detected, however, the process might result in a high rate of false positives. On the other hand, misuse detection analyzes attack behavior by establishing templates of attack characteristics, which is then 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 would not be effective.

Past research suggests that information available in the network is sufficient, and therefore, IDSs are preferred [6, 7, 8, 9]. Examples of network-based systems, which have been commercial successes are Suircate [10], Snort [11], and Bro IDS [12]. As reported by the commercial systems, the usefulness of IDSs is limited due to poor quality alerts since unfortunately perfect detection is impossible [13]. For example, Snort deployed at a large financial institution, has reported 411,947 alerts per day [14]. Managing so many alerts by hand is completely infeasible, thus the need for ever improved IDS development.

The aim of this paper is to improve the classification accuracy of IDSs. In particular, this paper analyzes and classifies the NSL-KDD data set [15] to distinguish between normal and the different types of intrusion behavior and is an extension of the work published in [16]. The previous paper only analyzed normal vs. intrusive behavior whereas this paper also classifies the different attack classes. The classification in this paper is based on a deep learning ensemble method whereby related deep learning models are run on the data set and the weighted outcome is evaluated, thus, employing an ensemble method.

The paper is arranged as follows. Section 2 describes related work in the area of intrusion detection systems. In Section 3, the proposed approach is described. Section 4 contains the experiments conducted as well as the results and findings. The conclusion and future work is given in Section 5.

2 Related Work

In order to prevent temporary and permanent damages caused by unauthorized access, a multitude of different systems have been built to monitor data flow in networks. Unfortunately, none of these systems can detect all types of intrusions since the attack permutations are occurring over time. Thus, in an attempt to overcome this, machine learning algorithms have been applied to classify normal and anomalous behavior on the network.

Related work in the area can be summarized as follows. In [17, 18], K-means and K-nearest neighbor algorithms were used to perform the classification task. The approach works by using a centroid

function to choose the average and closest grouping of new instances in order to group similar training examples together.

Another classification approach used for IDSs are support vector machines (SVM). SVM divides the dimensional space into a smaller dimensional hyperplane [19, 20]. In [21], SVM was used to automate feature selection. Feature selection is a preprocessing step that is usually applied in the area of data mining, thus, improving the classification rate. The so-called CSV-ISVM algorithm was proposed in [22] and uses incremental SVM in order to select candidate support vectors showing the advantages in real-time network intrusion detection.

Threshold-based anomaly detection, also known as signature matching, has been widely applied to model network traffic as discussed in [23]. The authors argue that traditional network-based profile models are not sufficient enough to satisfy user profiles in the environment. Therefore, a genetic algorithm approach was used to find signatures of pattern detection rules via permutations of parent signatures. Another approach proposed a core-plus-module framework (STAT) that is based on the state transition analysis technique [24]. This is done in order to tailor the design of an IDS to specific traffic types and environments. Other research compared a genetic algorithm approach with other approaches such as Naive Bayes and K-nearest neighbor as provided in [19].

Applications of deep neural networks (DNN) have seen quite an uptake in recent years, in particular, since significant breakthroughs have been achieved for tasks such as image recognition, speech recognition, text recognition, and language translation. Deep learning 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) [25] was applied to image, text, and voice learning tasks whereby a DBN is formed by stacking several restricted Boltzmann machines (RBM) [26]. These RBMs 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 intrusion detection as well. Several different DBN architectures have been applied to the intrusion detection task. For example, a DBN approach is proposed in [27] where a two-layer RBM is used to train the network in an unsupervised fashion. This is followed by a feedforward layer whereby back-propagation 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.

A DBN algorithm was implemented and applied on the NSL-KDD data set [28]. 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 was implemented in [29]. The autoencoder learning method is used to reduce the dimensionality of the data. This dimensionality reduction allows to convert high-dimensional data to a low-dimensional transformation with nonlinear mapping, and thus extracts the main features of the data. After this first step, DBN learning is applied to detect anomalies. The DBN consists of multilayer RBMs followed by a feedforward layer. First, the RBMs are trained using an unsupervised approach, which is then 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 [30]. 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. This approach is used 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. However, besides the detection rate and the execution time no other measures were reported on.

In [31], an accelerated DNN is proposed. 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 to reduce the error during the next iteration. Since gradient calculations are involved during the backpropagation process, and given the depth of the network long training times are unavoidable.

A deep learning approach for flow-based anomaly detection in a Software Defined Networking (SDN) environment is introduced in [32]. The DNN model is built using six basic features, the ones that can be easily obtained in an SDN environment, and different learning rates were experimented with. The results were very promising with the best accuracy achieved when the learning rate was 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 [33] and others to the NSL-KDD data set.

3 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.

3.1 Basic Concepts of Ensemble Learning

The concept of ensemble learning was first introduced in 1979 [34], which used 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 to introducing the variance reduction property [35]. However, research in [36] 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

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

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

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

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

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

3.2 Proposed Ensemble Deep Neural Network Classifiers

3.2.1 Autoencoder (AE)

Autoencoders are composed of one input, one hidden and one 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. This process offers dimensionality reduction and thus 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. This effectively removes the output layer from the autoencoder, so that the hidden layer can be used as input features for the classification or another autoencoder. The network is being trained by 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 is identical to the implementation in [31]. The architecture consists of two autoencoders of size 20 and 10, respectively, followed by a fully connected layer of size 5.

3.2.2 Deep Belief Neural Network (DBN)

A DBN trains 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 achieved by learning certain parameters via random sampling in order to learn the model. The cascade of RBMs allows the hidden vectors of one RBM to be the input for the next RBM etc.

The following rules are applied:

- 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 is effectively a feature extraction method. The training of the RBM layers is done without the labels (unsupervised training). The last layer of the network is a fully connected layer and is trained with labeled data (supervised training).

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.

3.2.3 Deep Neural Network (DNN)

A DNN consists of an input, several hidden layers, and an output layer and 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 used [38]. Adam (Adaptive Moment Estimation) uses separate learning rates for each weight as well as an exponentially decaying average of previous gradients, which leads to better results.

3.2.4 Extreme Learning Machine RBM (ELM)

Extreme learning machine is a learning algorithm that is used to train a single hidden layer neural network [39]. The input weights and hidden biases are randomly generated and 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 than standard NN and SVM and at the same time achieves higher generalization performance [40].

The ELM architecture that was used for the experiments follows the implementation as given in [30]. The network structure used is a 110-90-50-25 layer architecture trained with a maximum number of iterations of 300.

4 Experiments and Results

4.1 NSL-KDD Data set

The MIT Lincoln Lab held a DARPA-sponsored IDS event which simulated an attack scenario to the Air-Force base with a repeat event one year later [41] in 1998. Improvements were suggested by the computer security community dur-

ing these events. The DARPA data set [42] 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. This was done 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. Several researchers have pointed out various shortcomings [43]. These are the following:

- Imbalanced 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 of the network traffic is given as either normal or a specific attack type. The simulated attack types 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: an attacker scans the network to gather information in order to exploit the systems; an example is port scanning.
- Remote to Local (R2L): an 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): an 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 1.

Table 1. 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 2 shows the number of training and testing records and their distribution based on the type of network traffic (normal or attack type). There are $125,973$ records in the training data set, and $22,543$ records in the testing data set.

For the visualization of the data, PCA (principal component analysis) is used. PCA is a statistical procedure that uses an orthogonal transformation to convert the data points of correlated variables into a set of values of uncorrelated variables (principal components). The first principal component has the

Table 2. 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

largest possible variance and each of the other components has the second, third, etc. highest variance. Thus, the resulting vectors present an uncorrelated orthogonal basis set.

Figure 1 shows the result of the PCA applied to the training data set. The Figure shows the data separated into the different attack classes as well as the normal class using PCA, i.e., all 5 classes are shown.

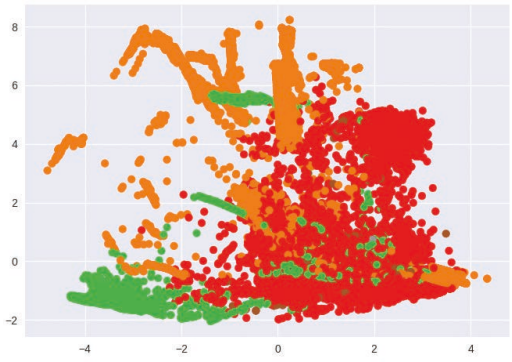

Figure 1. Four attack types

4.2 Evaluation Measures

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

- 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 actually normal behavior.
- False negatives (FN): are the wrong predictions whereby instances that are attacted 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

$$2\times\frac{P\times R}{P+R}. \tag{9}$$

4.3 Results

The experiments from the previous binary classification where the outcomes normal and attack were investigated achieved the following results [16]:

- Accuracy = 92.50%
- AUC = 91.62%

– False alarm rate = 14.72%
– Detection rate = 97.95%
– F1 score = 93.70%

In addition, in the same paper [16] a comparison was done with DNN [37], DBN [27], Autoencoder DNN [31], ELM-DBN [30], and DNN2 [32]. From the results, we saw that our proposed method achieved values of 93%, 92% and 92% for precision, recall and f-measure, respectively and overall obtained better results with the exception of the classification accuracy.

The following are the results of the run achieved during the testing phase. Table 4.3 shows the confusion matrix whereby 9,391 and 6,680 are correctly classified as normal and DoS, respectively.

Table 3. Confusion matrix - normal vs. DoS

	normal	DoS
normal	9,391	320
DoS	778	6,680

The different metric scores are listed in Table 4. The main measure for IDSs is the false alarm rate, which should be low, and the detection rate, which should be high. Results of 3.30% and 89.57% are achieved, respectively. Other values of importance are the classification accuracy and AUC with values of 93.60% and 93.14%, respectively.

Table 4. Various metric scores - normal vs. DoS

Accuracy	0.9360480
AUC	0.9313650
False alarm rate	0.0329523
Detection rate	0.8956820
F1 score	0.9240560

Precision, recall, F1-score and support results are given in Table 5.

Table 5. Precision, recall, F1-score and support - normal vs. DoS

	Precision	Recall	F1-score	Support
0.0	0.92	0.97	0.94	9,711
1.0	0.95	0.90	0.92	7,458
avg/total	0.94	0.94	0.94	17,169

Table 3 displays the confusion matrix showing 8,807 and 2,253 records were correctly classified as normal and probe, respectively, with 1,072 records being misclassified.

Table 6. Confusion matrix - normal vs. Probe

	normal	probe
normal	8,807	904
probe	168	2,253

A false alarm rate of 9.31% and a detection rate of 93.06% were achieved on the test data set as shown in Table 4. A classification accuracy of 91.16% was achieved with an AUC of 91.88%. The F1-score resulted in 80.78%.

Table 7. Various metric scores - normal vs. Probe

Accuracy	0.9116390
AUC	0.9187580
False alarm rate	0.0930903
Detection rate	0.9306070
F1 score	0.8078160

Precision, recall, F1-score and support results are given in Table 8.

Table 8. Precision, recall, F1-score and support - normal vs. Probe

	Precision	Recall	F1-score	Support
0.0	0.98	0.91	0.94	9,711
1.0	0.71	0.93	0.81	2,421
avg/total	0.93	0.91	0.92	12,132

Table 3 shows the confusion matrix whereby 9,694 and 892 records are correctly classified as normal and R2L, respectively.

Table 9. Confusion matrix - normal vs. R2L

	normal	R2L
normal	9,694	17
R2L	1,862	892

The different metric scores are listed in Table 10. Results of 0.17% and 32.39% are achieved for false alarm rate and detection rate, respectively.

Other values of importance are the classification accuracy and AUC with both achieving 89.93%.

Table 10. Various metric scores - normal vs. R2L

Accuracy	0.8492580
AUC	0.6610710
False alarm rate	0.0017506
Detection rate	0.3238930
F1 score	0.4870320

Precision, recall, F1-score and support results are given in Table 11.

Table 11. Precision, recall, F1-score and support - normal vs. R2L

	Precision	Recall	F1-score	Support
0.0	0.84	1.00	0.91	9,711
1.0	0.98	0.32	0.49	2,754
avg/total	0.87	0.85	0.82	12,465

Table 12 displays the confusion matrix showing that 9,697 and 44 records were correctly classified as normal and U2R, respectively, with 170 records being misclassified.

Table 12. Confusion matrix - normal vs. U2R

	normal	U2R
normal	9,697	14
U2R	156	44

A false alarm rate of 0.14%, and a detection rate of 22.00% were achieved on the test data set as shown in Table 13. A classification accuracy of 98.28% was achieved with an AUC of 60.93%. The F1-score resulted in 34.11%.

Table 13. Various metric scores - normal vs. U2R

Accuracy	0.9828470
AUC	0.6092790
False alarm rate	0.0014417
Detection rate	0.2200000
F1 score	0.3410850

Precision, recall, F1-score and support results are given in Table 14.

Table 14. Precision, recall, F1-score and support - normal vs. U2R

	Precision	Recall	F1-score	Support
0.0	0.98	1.00	0.99	9,711
1.0	0.76	0.22	0.34	200
avg/total	0.98	0.98	0.98	9,911

Figure 2 shows the summary of the results in a graph. Again, we can see the results obtained for the five measures and the four different types of attacks. As can be seen, the accuracy achieved for all attack classes are fairly high ranging between 85.93% to 98.28%. However, given the unbalanced nature of having far less samples for attack classes R2L and U2R, the AUC and other measures for the same are rather low. However, the false alarm rate for R2L and U2R attacks are very low with 0.17% and 0.14%, respectively.

The class imbalance issue is a well-known problem [44]. Different imbalance solutions to this problem fall into two major categories: sampling based approaches and cost function based approaches. The idea behind cost function based approaches is that false negatives are scored higher in terms of the cost function than false positives. The sampling based approaches consists of undersampling, oversampling and hybrid methods. The idea behind oversampling and undersampling is that either samples are removed from the majority class or are added to the minority class, respectively. The hybrid method represents a mix between under- and oversampling.

5 Conclusion

Different methods for IDSs have been proposed in the past and many of these systems implement a data mining approach whereby the data mining approaches can be classified into clustering and classification approaches. In this paper, a classification model using deep neural networks was investigated. In particular, the NSL-KDD data set was used applying a deep neural network ensemble technique. The ensemble technique comprised of different deep neural network architectures such as an autoencoder, a deep belief neural network, a deep neural network, and an extreme learning machine. The most important measures to evaluate in

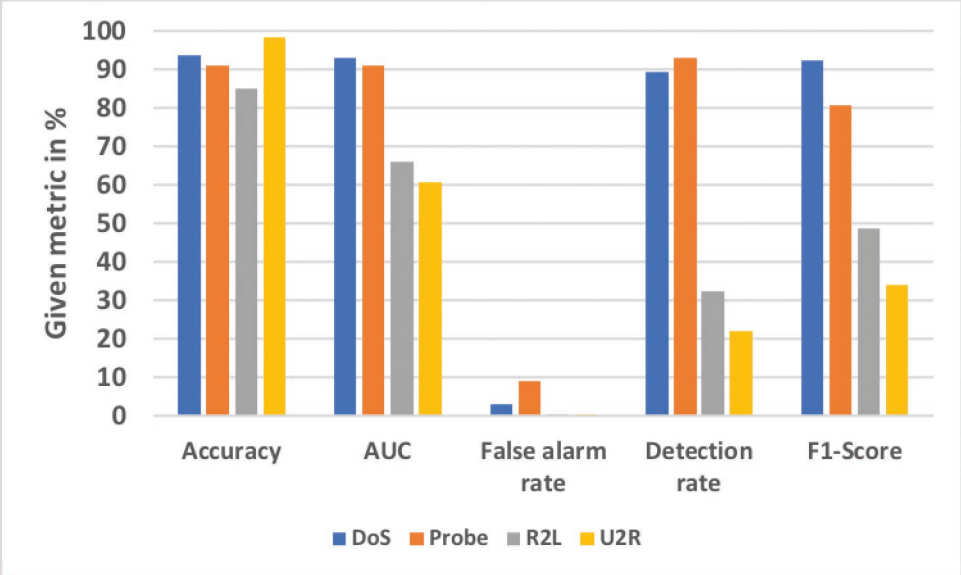

Figure 2. Overall performance results

this area 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 normal and attack instances that are correctly classified. Other measures considered are classification accuracy, AUC, precision, recall, and F-measure.

The results revealed that the accuracy achieved for all attack classes are fairly high ranging between 85.93% and 98.28%. In addition, the AUC, detection rate and F1-score are high for the DoS and Probe attack classes. However, the results for the attack classes R2L and U2R are rather low. The reason for this is the class inbalance given that there are only 200 and 2,754 samples in the test set for R2L and U2R, respectively, compared to the overall total of 22,543 samples. However, in terms of false alarm rate, which is an important feature for IDSs, R2L and U2R achieve good results with values of 0.17% and 0.14%, respectively.

References

[1] Cyber security, http://whatis.techtarget.com/definition/cybersecurity, last retrieved in 2018.

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

[3] 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, last retrieved in 2018.

[4] 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.

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

[6] P. O. Kane, S. Sezer, K. McLaughlin, Obfuscation: the hidden malware, IEEE Security & Privacy 9 (5), 41-47, 2011.

[7] G. Gu, P. Porras, V. Yegneswaran, M. Fong, W. Lee, Bothunter: Detecting malware infection through ids-driven dialog correlation, in: Proceedings of 16th USENIX Security Symposium, USENIX Association, 2007.

[8] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al., Botminer: Clustering analysis of network trace for protocol-and structure-independent botnet detection., in: USENIX Security Symposium, pp. 139-154, 2008.

[9] G. Gu, J. Zhang, W. Lee, Botsniffer: Detecting botnet command and control channels in network trace, in: Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08), 2008.

[10] V. Julien, Suricata ids, Tech. rep., Open Information Security Foundation (OISF), available online: http://suricata-ids.org/download/, last retrieved in 2018.

[11] M. Roesch, Snort: Lightweight intrusion detection for networks., in: LISA, pp. 229-238, 1999.

[12] V. Paxson, Bro: a system for detecting network intruders in real-time, Computer networks 31 (23), 2435-2463, 1999.

[13] D. M. Chess, S. R. White, Undetectable computer viruses, in: Virus Bulletin, pp. 107-115, 2000.

[14] R. Vaarandi, K. Podins, Network ids alert classification with frequent itemset mining and data clustering, in: Network and Service Management (CNSM), 2010 International Conference on, IEEE, pp. 451-456, 2010.

[15] 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.

[16] S. A. Ludwig, Intrusion Detection of Multiple Attack Classes using a Deep Neural Net Ensemble, IEEE Symposium Series on Computational Intelligence (SSCI), Honolulu, HI, USA, October 2017.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

[24] 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, pp. 221-228, Western Cape, South Africa, 2004.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] 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.

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

[39] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing, vol. 70, no. 1-3, pp. 489-501, 2006.

[40] 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.

[41] 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.

[42] DARPA Intrusion Detection Data Set, 1998.

[43] 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.

[44] N. V. Chawla, N. Japkowicz, A. Kotcz, Editorial: Special Issue on Learning from Imbalanced Data Sets, SIGKDD Explor. Newsl., vol. 6, no. 1, pp. 1-6, 2014.

Simone A. Ludwig is a Professor of Computer Science at North Dakota State University, USA. Prior to joining NDSU she worked at the University of Saskatchewan (Canada), Concordia University (Canada), Cardiff University (UK) and Brunel University (UK). Dr. Ludwig received her Ph.D. degree and M.Sc. degree with distinction from Brunel University in 2004 and 2000, respectively. Before starting her academic career she worked several years in the software industry. Dr. Ludwig's research interests lie in the area of computational intelligence including swarm intelligence, evolutionary computation, neural networks, and fuzzy reasoning. Example application areas are data mining (including big data), image processing, intrusion detection, cryptography, and cloud computing.