Evaluating Effectiveness of Shallow and Deep Networks to Intrusion Detection System

Vinayakumar R*, Soman KP* and Prabaharan Poornachandran†
*Centre for Computational Engineering and Networking (CEN), Amrita School of Engineering, Coimbatore
†Center for Cyber Security Systems and Networks, Amrita School of Engineering, Amritapuri
Amrita Vishwa Vidyapeetham, Amrita University, India
Email: vinayakumarr77@gmail.com

***Abstract*—Network intrusion detection system (NIDS) is a tool used to detect and classify the network breaches dynamically in information and communication technologies (ICT) systems in both academia and industries. Adopting a new and existing machine learning classifiers to NIDS has been a significant area in security research due to the fact that the enhancement in detection rate and accuracy is of important in large volume of security audit data including diverse and dynamic characteristics of attacks. This paper evaluates the effectiveness of various shallow and deep networks to NIDS. The shallow and deep networks are trained and evaluated on the KDDCup '99' and NSL-KDD data sets in both binary and multi-class classification settings. The deep networks are performed well in comparison to the shallow networks in most of the experiment configurations. The main reason to this might be a deep network passes information through several layers to learn the underlying hidden patterns of normal and attack network connection records and finally aggregates these learned features of each layer together to effectively distinguish the normal and various attacks of network connection records. Additionally, deep networks have not only performed well in detecting and classifying the known attacks additionally in unknown attacks too. To achieve an acceptable detection rate, we used various configurations of network settings and its parameters in deep networks. All the various configurations of deep network are run up to 1000 epochs in training with a learning rate in the range [0.01-0.5] to effectively capture the time varying patterns of normal and various attacks.**

***Index Terms*—Shallow and Deep networks.**

I. Introduction

Information and communication technology (ICT) systems are essential for today's rapidly growing powerful technologies. At the same time, ICT system has been encountered by various attacks. These attacks are diverse and gradually advancing. To attack these various threats, both academia and industries have adopted various solutions to their ICT systems. This is categorized in to 2 types (1) firewalls, encryption and decryption techniques of cryptography, software updates and many others are belongs to static methods (2) anomaly and intrusion detection are belongs to dynamic methods. Static approaches serve as an initial shelter to the ICT systems. In dynamic analysis, intrusion detection have become an efficient method and largely adopted in commercial solutions systems. Due to these reasons, intrusion detection has been widely studied area in security research since from 1980's, a seminal work by John Anderson on the "Computer Security threat monitoring and surveillance" [1]. IDS monitors the computer or networks to detect the malicious events, otherwise they may perform illegal actions to break computers or network. Primarily, based on the network behaviors intrusion detection is classified in to host based IDS (HIDS) and network based IDS (NIDS). HIDS rely on the information of log files including sensors logs, system logs, software logs, file systems, disk resources and others in each system. NIDS inspect each packet contents in traffic flows. Mostly organizations use a hybrid of both the NIDS and HIDS. This paper gives its focus to NIDS.

Analysis of network traffic flows is done using misuse detection, anomaly detection and state full protocol analysis. Misuse detection uses predefined signatures and filters to detect the known attacks. It relies on human inputs to constantly update their signature database. This method is accurate in finding the known attacks but in the case of unknown attacks completely ineffective. Anomaly detection uses heuristic mechanism to find the unknown malicious activities. In most of the scenarios, anomaly detection produces high false positive rate. To combat high false positive rates, most organization used the combination of both the misuse and anomaly detection in their commercial solution systems. State full protocol analysis is most powerful in comparison to the aforementioned detection methods due to the fact that state full protocol analysis acts on the network layer, application layer and transport layer. This uses the predefined vendor's specification settings to detect the deviations of appropriate protocols and applications.

Mostly, the existing NIDS of commercial systems in market are based on statistical measures or threshold computing mechanisms. Both of these mechanisms uses feature sets such as packet length, inter-arrival time, flow size and other network traffic parameters as features to effectively model them within a specific time-window. These commercial based solutions are in-effective for present day attacks. Self-learning system is one of the effective methods to deal with the present day attacks. This uses supervised, semi-supervised and unsupervised mechanisms of machine learning to learn the patterns of various normal and malicious activities with a large corpus of normal and attack connection records. Though various machine learning based solution found, the applicability in commercial system is in early stages [2]. The existing machine learning based solutions outputs a high false positive rate icluding a high computational cost [3]. The primary reason for this is machine learning classifiers learn the characteristic of simple TCP/IP features locally. Deep learning is complex subnet of machine learning that learns hierarchical feature representations and hidden sequential relationships through by passing the TCP/IP information on several layers. Deep learning have achieved a significant results in long standing artificial intelligence (AI) tasks in the field of image processing, speech recognition, natural language processing and many others. Additionally, these performances have been transformed to most important cyber security tasks [4], [5], [6]. This work towards analyzing an effectiveness of various shallow and deep networks for NIDS with the openly available network based intrusion data sets such as KDDCup '99' and NSL-KDD .

Towards this end, the rest of the paper is structured as follows: section II review the related work published on intrusion detection. Section III describes the necessary notion of deep networks. In section IV, reports the evaluation of experimental results and at the end, feature work direction of this research and conclusion is placed in section V.

II. Related Work

This section discusses the important published results of various machine learning based solutions to intrusion detection briefly till date. The openly available sufficient labeled data for intrusion detection is very less. This might be one of the major reasons towards not having an efficient machine learning based solution for NIDS. A few data sets exist such as KDDCup '99' [1] and NSL-KDD [2]. However, each has its own shortcomings. The major shortcoming of both of these data sets is that the connection records may not be a real representative of network traffic TCP/IP connection records. Though, both KDDCup '99' and NSL-KDD data sets have been used in assessing the effectiveness of various existing and newly introduced machine learning classifiers. KDDCup '99' is the most well-known data sets for various tasks in Cyber security. This was initially generated and used in KDDCup '98' and followed by KDDcup '99' challenges. The detailed evaluation results and its methods of these both competitions were published by [7]. Most of the submitted entries in both the competitions have used variants of decision tree. In KDDCup '99' challenge, the first 3 winning entries had very less difference in their performance. A large difference in accuracy was found between 17th and 18th entries. After this challenge, KDDCup '99' data set is used in various NIDS system research studies. Most of them have used only the 10% data and others used mixed data sets. Moreover, after the KDDCup '98' and KDDCup '99' challenge, both KDDCup '98' and '99' data sets have used in examining the effectiveness of machine learning methods with feature engineering mechanisms such as dimensionality reduction [8] and other research studies have used custom built-in data sets [9]. In beginning stages, [10] used PNrule method that composed of P-rules to predict the existence of the class and N-rules to predict the non-existence of the class. This has enhanced detection rate in attack categories except to the user-to-root (U2R) category in comparison to the KDDCup '99' winning entries. The importance of each feature was studied by [11] and they also examined the significance of minimal feature sets for identifying each attack categories. In [12] discussed the effectiveness of random forest classifier in misuse detection, anomaly detection and hybrid of misuse detection and anomaly detection. They were claimed that their misuse approach resulted in better detection rate compared to KDDCup '99' challenge winning entries and anomaly detection outperformed other published unsupervised anomaly detection methods. Additionally, they reported that their hybrid system enhances the attack detection rate by taking the advantage from the misuse and anomaly detection methods. In [13] discussed the effectiveness of Adaboost classifier to NIDS by using the KDDCup '99' data set. They considered weak classifiers are decision stumps. For each categorical and continuous feature, they constructed decision rules and by combining both of them results a strong classifier. They claimed that their algorithm includes low computational complexity and less error rates with showing the experimental results on test bed framework. In [14] discussed the applicability of shared nearest neighbor (SNN) classifier to intrusion detection. They claimed that their method as best one by reporting a higher detection rate. This was not acceptable due to the fact that they used the custom-built-in data set. In [15] represented a particular class of a network connection as root node and other leaf nodes to represent the feature of its connection. In [16] discussed the Naive Bayes network to intrusion detection in two scenarios; one was classifying the network connection as either normal or abnormal and in second scenario they classified an attack to particular category. In some of the cases Bayesian net had performed well in comparison to the winning entries. In [17] discussed the nonparametric density estimation approach to NIDS with considering only the normal data at training phase. They were able to achieve closer detection rate to winning entries of KDDCup '99' challenge. In [18] discussed the various ensemble learning approach to NIDS. [19] used genetic algorithm with the aim to model the temporal and spatial information to identify complex and diverse anomalous behavior. In [20] discussed kernel based mechanisms to real time-IDS. They also reported that online-feature extraction mechanism with least square support vector machine classifier performed well in comparison to the off-line intrusion detection system. In [21] discussed the multi-layer perceptron (MLP) for NIDS using off-line mechanism. In [4], [5], [6] discussed the taxonomy and survey of shallow and deep networks in detail.

III. Background

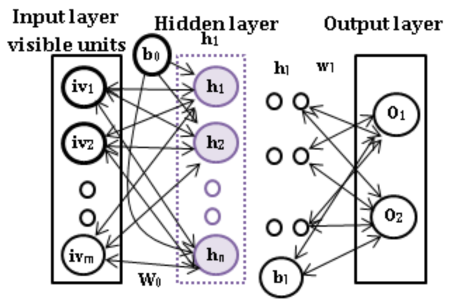
This section discusses the concepts of deep belief networks mathematically.

A. Deep Belief Networks (DBN)

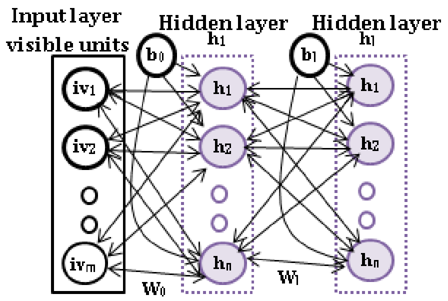
Deep belief network (DBN) or deep networks is a generative architecture, appeared as a descendant network of traditional

[1]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[2]http://www.unb.ca/cic/research/datasets/nsl.html



(a)



(b)

Fig. 1: (a) Deep belief network (DBN) architecture (b) RBM architecture

artificial neural network (ANN). It contains an input layer, one or more hidden layers (with one layer of DBN is same as FFN) and an output layer, as shown in Fig. 1a. Both the input and hidden layer have one or more neurons, typically termed as units in mathematical term. The initialization of appropriate value for parameters of DBN is often considered as difficult one. Random initialization mechanism might generate low generalization by learning poor local minima. Moreover, a network with more than one hidden layer might consume more time for its convergence. To avoid random initialization, [22] introduced an unsupervised learning mechanism such as RBM, as shown in Fig. 1b to learn the compact feature vectors greedily by passing an input vector through one or more RBM hidden layers during training phase. An RBM is defined using an energy function as follows,

$$R(iv, h) = -h^T iv - bi^T iv - bh^T h \tag{1}$$

where iv denotes the visible unit of an input layer, h is the hidden layer units, w denotes the weight matrix, bi, bh are bias terms between input layer and visible units and hidden layer and non-visible units. The conditional distribution from one layer to the next one is defined as below.

$$P(iv_i = 1|h) = \sigma(bi_i + \sum\nolimits_k w_{ki} h_k) \tag{2}$$

$$P(h_k = 1|iv) = \sigma(bh_k + \sum\nolimits_i w_{ki} h_i) \tag{3}$$

where, $\sigma(x) = \frac{1}{1+e^{-z}}$.

Training phase of DBN has two steps: (1) pre-training (2) reconstruction. In pre-training the given training samples without class labels are propagated stochastically across RBM layers. The top level RBM layer is typically called as an associative memory. Each RBM layer has learned features to represent data in the previous layers. Each hidden units in hidden layer follows conditional distribution and generates feature vectors in the form of binary. And again the binary feature vectors are propagated in reverse direction i.e. from the hidden units in hidden layer to input visible units in input visible layer to reconstruct the same training samples. This procedure is followed iteratively for all the training samples. The update of values in parameter are mathematically formulated as follows

$$\Delta w_{ki} = \eta((iv_i h_k)_{input\,samples} - (iv_i h_k)_{reconstruction\,samples}) \tag{4}$$

$$\Delta bi_i = \eta((iv_i)_{input\,samples} - (iv_i)_{reconstruction\,samples}) \tag{5}$$

$$\Delta bh_i = \eta((iv_i)_{input\,samples} - (iv_i)_{reconstruction\,samples}) \tag{6}$$

where, η denotes the learning rate.

IV. Methodology and Results

For DBN implementation we used GPU enabled TensorFlow [23] in single NVidia GK110BGL Tesla k40, the python scikit-learn library [24] is used for implementing classical machine learning classifiers such as LR, NB, KNN, DT, AB, RF, and SVM. For ELM implementation, we used hpelm toolbox [25].

A. Network Intrusion Detection System (NIDS) Datasets

The choice of publically available data sets for NIDS is very less for security researchers. KDDCup '99' and NSL-KDD are the most-well known and openly available data sets for evaluating the effectiveness of various existing and newly introduced machine learning algorithms. This can't be used in developing real time IDS due to the reason that both of these data sets doesn't shows the real time network traffic connection records. A connection record characterizes the communications session associated in the form of packets between hosts. In addition, these data sets have other harsh criticisms but still both of these used in many NIDS research studies. In this paper, we use these data sets to evaluate the effectiveness of various classical and deep learning algorithms. This section provides the detailed information of these data sets.

1) DARAPA/KDDCup '99' dataset: At initially, NIDS data set collection was started by Defense Advanced Research Projects Agency (DARPA) Intrusion Detection Evaluation Group (presently the Cyber Systems and Technology Group) in air force base local area network (LAN), MIT Lincoln Laboratory with open-handed funding support from Defense

TABLE I: Description of KDDCup '99' and NSL-KDD 10% datasets

Attack Category	**Data instances 10% data**			
	KDD		**NSL-KDD**	
	Train	Test	Train	Test
Normal	97278	60593	67343	9710
DOS	391458	229853	45927	7458
Probe	4107	4166	11656	2422
R2L	1126	16189	995	2887
U2R	52	228	52	67
Total	**494021**	**311029**	**125973**	**22544**

Advanced Research Projects Agency (DARPA ITO) and Air Force Research Laboratory (AFRL/SNHS). They named that the collected data as DARPA. They used totally around 1000's of UNIX machines in MIT Lincoln laboratory for real time simulation of network traffic. These 1000's machines were accessed by 100's users at a time. The real time simulation was run continuously for 9 weeks to acquire the raw TCP dump data; 7 weeks for training and two weeks for testing. The attacks are made to UNIX machines in MIT Lincoln laboratory from external location of air force LAN using Windows NT and UNIX workstations machines. They conducted 32 distinct attacks and 7 distinct attack scenarios to collect various 300 attacks. For the purpose of evaluation, the raw data set was distributed in KDDCup '98' challenge. In next year, the raw data set was converted to connection records using the Mining Audit data for automated models for Intrusion Detection (MADMAID) framework and distributed in KDDCup '99' challenge at fifth International Conference on Knowledge Discovery and Data Mining. The primary task of the challenge was to classify the network connection records as either normal or an attack to one particular category ('dos', 'probe', 'r2l', 'u2r'). The statistics of these network connection records are placed in Table I.

2) NSL-KDD: To alleviate the issues existing in KDDCup '99', [26] introduced NSL-KDD. NSL-KDD is a refined version of KDDCup '99' intrusion data. The data set was made by removing the redundant connection records and in addition the invalid records, specifically 136,489 and 136,497 connection records are removed from KDDCup '99'. As a result, it protects machine learning algorithms not to be biased towards more frequent records in training phase. The data set is worked completely well for the task of misuse detection in comparison to the KDDCup '99'. Yet, this also suffers in exhibiting the live network traffic profile characteristics. A detailed statistics of NSL-KDD is placed in Table I.

B. Evaluation Results for Binary Classification

Initially, the feature sets of each connection records are preprocessed and normalized. To classify the connection record as either normal or attack, we used various classical machine learning classifiers for KDDCup '99'. The performance of them is known by confusion matrix. The confusion matrix provides class-specific metrics such as overall accuracy, precision, recall, f-measure, true-positive-rate (TPR) and false-positive-rate (FPR). LR classifier performance related to the overall accuracy statistical measure is low in comparison to other classifiers such as NB, KNN, DT, AD, RF, and SVM. Moreover, the accuracy has a marginal difference. For NSL-KDD, both LR and NB performance is low in comparison to other classifiers such as, KNN, DT, AD, RF, and SVM. The detailed results is reported in Table II

The above employed classical algorithms are shallow in nature. In our next experiment configuration, we adopted deep networks to know whether any improvement in performance. These deep networks pass the connection records to more than one hidden layers. Each hidden layer has a non-linear function. As a result deep networks learns the non-linear or highly varying patterns in connection records of training data set to distinguish them as normal or attack.

C. Network Topologies

Generally, the deep networks are parameterized, so to find optimal parameters related to the number of units and hidden layer three trails of experiments conducted on the following deep network topologies; (1) MLP 1 layer with 60 neurons (2) MLP 2 layer with 90 neurons (3) MLP 3 layer with 120 neurons (4) MLP 4 layer with 150 neurons (5) DBN 1 layer with 200 neurons (6) DBN 2 layer with 250 neurons (7) DBN 3 layer with 300 neurons (8) DBN 4 layer with 350 neurons.

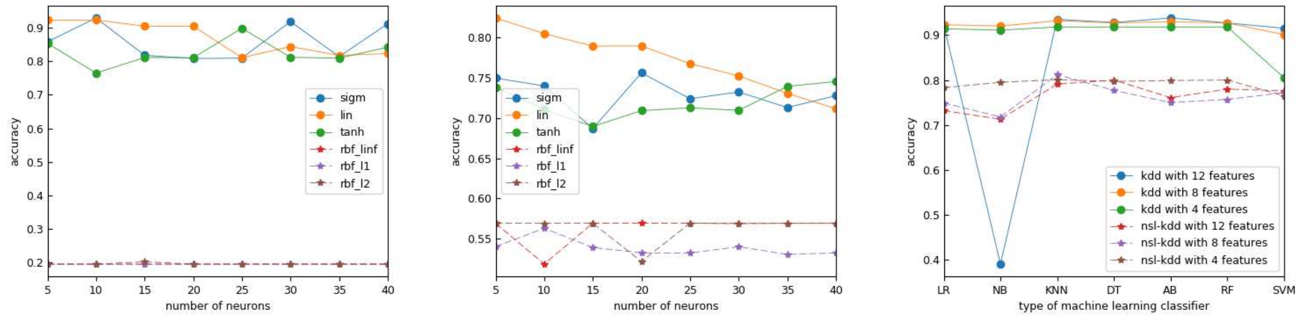
Initially, two trails of experiments are done on the moderately sized MLP network and DBN with 1 layer containing 100 units. An input layer has 41 neurons that are fully connected to the hidden layer neurons. An output layer has two neurons that provide the output such as whether the connection record as either normal or attack. All trails of experiments are run till 500 epochs with learning rate in the range [0.01-0.5]. Experiments with lower learning rate have performed better in classifying the connection record as either normal or attack. They required more number of epochs (> 2000) for attack connection records. Otherwise, most of them are classified as normal. By considering the training time and attack detection rate, the learning rate is set as 0.1 for the rest of the experiments. Moreover, based on the knowledge on experiments performance, 500 epochs are not sufficient to train attack connection records.

In our next experiment setting, learning rate is fixed as 0.1. All four networks in both MLP and DBN have performed well in distinguishing the connection record as either normal or attack. Moreover, DBN architectures have outperformed MLP networks. More complex networks have attained good classification performance in both the networks as shown in Table III. Moreover, the networks have really performed well after 500 epochs.

In next experiment setting, to categorize an attack to corresponding category we adopted the above network topologies. 3 trails of experiments are run till 1000 epochs with a fixed learning rate 0.1. Both MLP and DBN network have required different number of epochs for learning each of the

TABLE II: Detailed results of various classical machine learning classifiers

Algorithm	KDDCup '99'				NSL-KDD			
	Accuracy	Precision	Recall	F-score	Accuracy	Precision	Recall	F-score
LR	0.848	0.989	0.821	0.897	0.720	0.620	0.905	0.736
NB	0.929	0.988	0.923	0.955	0.728	0.622	0.935	0.747
KNN	0.929	0.998	0.913	0.954	0.788	0.677	0.971	0.798
DT	0.929	0.999	0.913	0.954	0.796	0.687	0.969	0.804
AB	0.925	0.995	0.911	0.951	0.774	0.662	0.970	0.787
RF	0.927	0.999	0.910	0.952	0.779	0.667	0.975	0.792
SVM	0.924	0.996	0.909	0.950	0.772	0.666	0.947	0.782



(a) (b) (c)

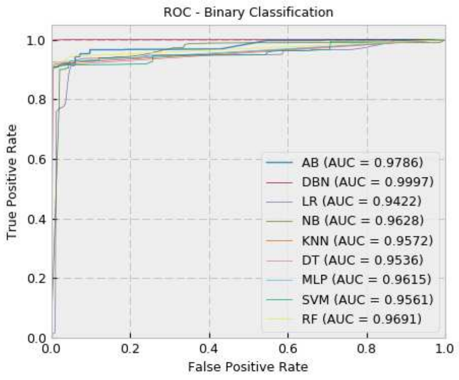
Fig. 2: Attack detection rate of (a) KDDCup '99' (b) NSL-KDD, (c) performance of minimal feature sets

attack and normal connection records. Moreover, the complex network topologies in both MLP and DBN have required large epochs to obtain an acceptable classification performance and at last obtained higher classification performance as well. The network started over fitting, once it learns the particular attack type. This point onwards the network has enhanced in remembering the training connection records. As a result, the generalization performance continuously lessens for that specific attack connection records. Rare attacks such as u2r and r2L have required large epochs to learn them in training. Additionally, learning complete representation of them is possible by adding few more input representations for training data. Both u2r and r2L attacks have not at all learnt by very simple such as 1 layer of MLP and DBN networks. 4 layer of MLP and DBN network attained greater performance with considering the training time and classification performance.

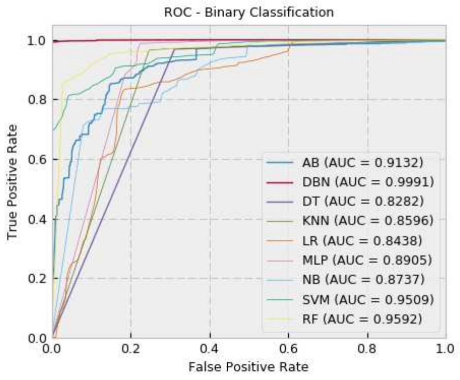
ELM has the following type of neurons or activation functions; lin, $sigm$, $tanh$, rbf_l1, rbf_l2 and rbf_linf. lin, $sigm$ and $tanh$ are dot product neurons that apply the corresponding activation function to the output (no function, $sigmoid$ function and $tanh$ function). rbf_l1, rbf_l2 and rbf_linf are Radial Basis Function (RBF) neurons that compute the corresponding distance (Manhattan, Euclidean or Chebyshev) between the input and the center point stored inside the neuron. They apply RBF activation function $\exp(-\beta * dis\tan ce)$. Here, exp is exponential function, β is kernel width. To find the optimal parameters related to number of hidden neurons and type of neuron in ELM, two trails of experiments done on varying number of neurons in range [5-40]. The best performed trail on each parameters of number of hidden neurons and type of neuron of KDDCup '99' and NSL-KDD NIDS data sets is displayed in Fig. 2a and Fig. 2b. During training, along with the training samples such as connection records, the number of hidden neurons with the type of neurons was given. Next, using the input samples feature matrix is constructed and input weights w and biases b generated randomly. Based on input weights matrix, the biases, the type of neuron or activation function and the feature matrix, hidden layer output matrix H is constructed. At last, by estimating the Moore-Penrose inverse of the hidden layer output matrix we found the output weights β. During testing, we passed the testing samples and extracted the feature matrix with its corresponding class label. Next, based on the random generated weights w and biases b , the hidden layer output matrix H is formed. At last, in order to get the predicted values, the target scores is calculated and compared with the expected labels to obtain the overall accuracy. In both the data sets, rbf_l1, rbf_l2 and rbf_linf have not at all performed well in comparison to other type of neurons over the number of neurons in range [5-40]. And lin as type of neuron has achieved highest accuracy in terms of detecting the connection record as either normal or an attack in both the datasets. ELM with lin as type of neurons has achieved highest accuracy with 10 neurons in KDDCup '99' and 5 neurons in NSL-KDD. ELM has performed well in comparison to the SVM in multi-class classification with accuracy 0.929 (SVM- 0.926) for KDDCup '99' and accuracy 0.776 (SVM- 0.755) for NSL-KDD.

TABLE III: Summary of test results of various MLP and DBN networks

Network	**Number of**	**KDDCup '99'**				**NSL-KDD**			
Layers	**Neurons**	**Accuracy**	**Precision**	**Recall**	**F-score**	**Accuracy**	**Precision**	**Recall**	**F-score**
MLP1	[60]	0.924	0.996	0.908	0.950	0.799	0.717	0.879	0.790
MLP2	[90,90]	0.934	0.995	0.922	0.958	0.811	0.701	0.879	0.817
MLP3	[120,120,120]	0.938	0.988	0.934	0.960	0.861	0.766	0.977	0.859
MLP4	[150,150,150,150]	0.939	1.000	0.924	0.960	0.866	0.768	0.988	0.864
DBN1	[200]	0.924	0.996	0.909	0.950	0.885	0.821	0.939	0.876
DBN2	[250,250]	0.929	0.998	0.913	0.954	0.896	0.814	0.984	0.891
DBN3	[300,300,300]	0.937	0.999	0.923	0.960	0.914	0.838	0.992	0.909
DBN4	[350,350,350,350]	0.997	1.000	0.997	0.998	0.973	0.944	0.996	0.969



(a)



(b)

Fig. 3: ROC curve (TPR vs FPR) for (a) KDDCup '99' (b) NSL-KDD

To get an intuitive understanding related to the classification performance of various classical machine learning and other neural network algorithms, we plotted Receiver operating characteristic (ROC) curves, as shown in Fig. 3a for KDDCup '99' and Fig. 3b for NSL-KDD. For KDDCup '99', the DBN has performed well by showing $AUC = 0.9997$ including the consistent true positive rate (TPR) and false positive rate (FPR) than the other approaches. The other mechanisms are comparable to DBN related to AUC measure. Also, DBN has achieved highest $AUC = 0.9991$ including the consistent TPR and FPR in comparison to the other approaches.

D. *Minimal Feature Sets*

Some features in KDDDCup '99' challenge data set contain redundant statistics and in some cases these statistics are low in comparison to real time network traffic. These issues may hinder the classification performance. To alleviate this, feature set reduction of KDDCup '99' challenge data set is studied in the perspective of data mining by [27], [28]. They reported 3 types of minimal feature sets such as 4-feature sets, 8-feature sets and 12-feature sets. Following this, we apply the classical machine learning classifiers and deep networks such as MLP and DBN to evaluate the performance of them in classifying the connection record in to either 'normal' or an 'attack' and classifying those attack to its categories. For KDDCup '99', the performance of classical machine learning algorithms is good with 8-feature set in comparison to the 4-feature set and 12-feature set. The performance of various classical machine learning classifiers are shown in Fig. 2c. Moreover, the statistical measure such as accuracy was comparable in all 3 types of minimal feature sets concerned with classical machine learning classifiers. Moreover, the performance related to accuracy of all 3 feature sets is same in comparison to the full feature sets in most of the classical machine learning algorithms. In the case of deep networks, both MLP and DBN performed well with 8-feature sets, as displayed in Table IV. But, the accuracy of these minimal feature sets is low in comparison to the full feature sets. The performance of deep networks for minimal feature sets of NSL-KDD is displayed in Table IV. DBN with 8 minimal feature sets has attained highest accuracy in comparision to the other feature sets and other deep networks.

E. *Evaluation Results for Multi-class Classification*

A connection record that classified as attack, is further categorized into different groups by employing various classical machine classifiers on KDDCup '99' and NSL-KDD. SVM has performed well in comparison to the other classical machine learning classifiers such as LR, NB, KNN, DT, AB and RF. Moreover, ELM has outperformed SVM in both KDDCup '99' and NSL-KDD. And, the performance of these shallow networks is low in comparison to the deep networks as shown in Table V, Table V, Table VI and Table VI.

TABLE IV: KDDCup '99' and NSL-KDD attack detection rate with minimal feature sets such as 4, 8 and 12

Minimal Features	NORMAL		DOS		PROBE		U2R		R2L		ACC
	TPR	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	
KDDCup '99'											
MLP4	0.871	0.079	0.927	0.170	0.553	0.011	0.0	0.0	0.0	0.0	0.886
MLP8	0.995	0.092	0.938	0.015	0.712	0.002	0.0	0.0	0.01	0.0	0.921
MLP12	0.996	0.081	0.941	0.040	0.794	0.002	0.0	0.0	0.001	0.0	0.923
DBN4	1.0	0.081	0.938	0.099	0.0	0.0	0.0	0.0	0.0	0.0	0.912
DBN8	0.998	0.081	0.939	0.052	0.654	0.002	0.0	0.0	0.018	0.0	0.921
DBN12	0.999	0.743	0.248	0.095	0.0002	0.0	0.0	0.0	0.0	0.0	0.381
NSL-KDD											
MLP12	0.760	0.168	0.901	0.318	0.470	0.012	0.0	0.0	0.0	0.0	0.684
MLP8	0.999	0.530	0.55	0.044	0.464	0.0024	0.0	0.0	0.0	0.0	0.667
MLP4	0.871	0.079	0.927	0.17	0.553	0.011	0.0	0.0	0.0	0.0	0.885
DBN12	0.996	0.095	0.781	0.024	0.849	0.077	0.447	0.001	0.529	0.019	0.686
DBN8	1.0	0.065	0.807	0.043	0.773	0.108	0.358	0.008	0.387	0.003	0.799
DBN4	0.999	0.003	0.928	0.059	0.842	0.109	0.388	0.003	0.194	0.001	0.715

TABLE V: Detailed results for KDDCup '99' and NSL-KDD using classical machine learning classifiers

Algorithm	NORMAL		DOS		PROBE		U2R		R2L		ACC
	TPR	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	
KDDCup '99'											
LR	0.969	0.185	0.841	0.07	0.073	0.002	0.0	0.0	0.0	0.0	0.833
NB	0.711	0.077	0.848	0.023	0.963	0.076	0.8	0.053	0.152	0.001	0.803
KNN	0.994	0.087	0.939	0.012	0.693	0.004	0.229	0.0	0.063	0.001	0.922
DT	0.995	0.085	0.941	0.004	0.745	0.002	0.343	0.002	0.125	0.0003	0.926
AB	0.99	0.625	0.383	0.03	0.116	0.002	0.0	0.0	0.003	0.0	0.487
RF	0.995	0.093	0.940	0.004	0.753	0.002	0.271	0.0	0.001	0.0	0.922
SVM	0.987	0.061	0.939	0.004	0.928	0.026	0.0	0.0	0.147	0.0001	0.926
ELM	0.997	0.073	0.940	0.005	0.809	0.009	0.157	0.001	0.172	0.002	0.929
NSL-KDD											
LR	0.926	0.468	0.641	0.122	0.171	0.020	0.0	0.0	0.0	0.0	0.635
NB	0.269	0.083	0.762	0.165	0.539	0.02	0.806	0.258	0.365	0.104	0.478
KNN	0.976	0.333	0.73	0.031	0.589	0.038	0.179	0.0001	0.084	0.016	0.741
DT	0.971	0.334	0.756	0.012	0.715	0.032	0.328	0.002	0.01	0.019	0.753
AB	0.936	0.37	0.796	0.146	0.11	0.009	0.0	0.0	0.0	0.0	0.685
RF	0.975	0.408	0.749	0.013	0.618	0.02	0.254	0.0003	0.002	0.0002	0.741
SVM	0.977	0.242	0.697	0.014	0.913	0.108	0.0	0.0	0.011	0.0009	0.755
ELM	0.974	0.232	0.774	0.015	0.854	0.091	0.0	0.0	0.023	0.001	0.776

V. Conclusion

This paper examines the effectiveness of shallow and deep networks to NIDS task by modeling the connection records of transmission control protocol / internet protocol (TCP/IP) information. In most of the scenario, deep networks performed well in comparison to the shallow networks in distinguishing the connection records as either normal or an attack and additionally in categorizing an attack to corresponding attack categories too. The primary reason to that is, a deep network passes the information through the several layers and non-linearity in each layer facilitates to learn the distinguishable patterns between normal and attack connection records. This work provides the detailed experiment results by exploiting the deep network to NIDS. As a result, it remains as a baseline work towards applying the other deep network to NIDS system. Though it performed well in identifying the attacks in comparison to the other shallow networks, experimental analysis results of deep networks on real time NIDS data set is essential. Because KDDCup '99' data set is one of the very few publically avilable one, but they are terribly outdated and not representative for today's network traffic.

References

[1] J. P. Anderson *et al.*, "Computer security threat monitoring and surveillance," Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, Tech. Rep., 1980.

[2] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23, pp. 2435–2463, 1999.

[3] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok, "Toward cost-sensitive modeling for intrusion detection and response," *Journal of computer security*, vol. 10, no. 1-2, pp. 5–22, 2002.

[4] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *Advanced Cloud and Big Data (CBD), 2014 Second International Conference on*. IEEE, 2014, pp. 247–252.

TABLE VI: Detailed results for KDDCup '99' and NSL-KDD using deep networks

Network	NORMAL		DOS		PROBE		U2R		R2L		ACC
Layers	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	**TPR**	**FPR**	
KDDCup '99'											
MLP 1	0.989	0.091	0.939	0.015	0.771	0.003	0.0	0.0	0.001	0.0	0.919
MLP 2	0.981	0.095	0.938	0.019	0.674	0.003	0.0	0.0	0.001	0.0001	0.917
MLP 3	0.983	0.144	0.882	0.013	0.785	0.006	0.0	0.0	0.0003	0.0	0.876
MLP 4	0.984	0.134	0.899	0.021	0.669	0.003	0.0	0.0	0.0	0.0	0.887
DBN 1	0.988	0.078	0.941	0.051	0.616	0.002	0.0	0.0	0.134	0.002	0.922
DBN 2	0.988	0.076	0.939	0.022	0.709	0.003	0.0	0.001	0.332	0.003	0.928
DBN 3	0.979	0.071	0.959	0.023	0.708	0.003	0.0	0.0	0.001	0.001	0.933
DBN 4	0.953	0.082	0.937	0.127	0.105	0.003	0.0	0.0	0.0	0.0	0.902
NSL-KDD											
MLP 1	0.975	0.093	0.768	0.031	0.775	0.128	0.0	0.0	0.214	0.027	0.789
MLP 2	0.976	0.149	0.777	0.058	0.817	0.065	0.0	0.0	0.343	0.007	0.812
MLP 3	0.972	0.254	0.758	0.025	0.785	0.089	0.0	0.0	0.035	0.001	0.764
MLP 4	0.975	0.392	0.658	0.018	0.733	0.056	0.0	0.0	0.0	0.0	0.721
DBN 1	0.998	0.084	0.828	0.073	0.839	0.114	0.0	0.0	0.002	0.004	0.793
DBN 2	0.974	0.233	0.778	0.015	0.857	0.091	0.0	0.0	0.029	0.001	0.776
DBN 3	0.982	0.092	0.776	0.014	0.922	0.111	0.0	0.0	0.261	0.029	0.817
DBN 4	0.979	0.256	0.778	0.014	0.809	0.058	0.0	0.0	0.089	0.015	0.793

[5] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *arXiv preprint arXiv:1701.02145*, 2017.

[6] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Aerospace and Electronics Conference (NAECON), 2015 National*. IEEE, 2015, pp. 339–344.

[7] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, "Analysis and results of the 1999 darpa off-line intrusion detection evaluation," in *Recent Advances in Intrusion Detection*. Springer, 2000, pp. 162–182.

[8] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Ieee communications surveys & tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[9] S. Chavan, K. Shah, N. Dave, S. Mukherjee, A. Abraham, and S. Sanyal, "Adaptive neuro-fuzzy intrusion detection systems," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, vol. 1. IEEE, 2004, pp. 70–74.

[10] R. Agarwal and M. V. Joshi, "Pnrule: A new framework for learning classifier models in data mining (a case-study in network intrusion detection)," in *Proceedings of the 2001 SIAM International Conference on Data Mining*. SIAM, 2001, pp. 1–17.

[11] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets," in *Proceedings of the third annual conference on privacy, security and trust*, 2005.

[12] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[13] W. Hu, W. Hu, and S. Maybank, "Adaboost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 577–583, 2008.

[14] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM, 2003, pp. 47–58.

[15] A. Valdes and K. Skinner, "Adaptive, model-based monitoring for cyber attack detection," in *Recent Advances in Intrusion Detection*. Springer, 2000, pp. 80–93.

[16] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayesian networks in intrusion detection systems," in *Proc. Workshop on Probabilistic Graphical Models for Classification, 14th European Conference on Machine Learning (ECML) and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), 23rd September, in Cavtat–Dubrovnik, Croatia*, 2003, p. 11.

[17] D.-Y. Yeung and C. Chow, "Parzen-window network intrusion detectors," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4. IEEE, 2002, pp. 385–388.

[18] L. Didaci, G. Giacinto, and F. Roli, "Ensemble learning for intrusion detection in computer networks," in *Workshop Machine Learning Methods Applications, Siena, Italy*, 2002.

[19] W. Li, "Using genetic algorithm for network intrusion detection," *Proceedings of the United States Department of Energy Cyber Security Group*, vol. 1, pp. 1–8, 2004.

[20] B.-j. Kim and I.-k. Kim, "Kernel based intrusion detection system," in *Computer and Information Science, 2005. Fourth Annual ACIS International Conference on*. IEEE, 2005, pp. 13–18.

[21] M. Moradi and M. Zulkernine, "A neural network based system for intrusion detection and classification of attacks," in *Proceedings of the IEEE International Conference on Advances in Intelligent Systems-Theory and Applications*, 2004, pp. 15–18.

[22] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[23] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[25] A. Akusok, K.-M. Björk, Y. Miche, and A. Lendasse, "High-performance extreme learning machines: a complete toolbox for big data applications," *IEEE Access*, vol. 3, pp. 1011–1025, 2015.

[26] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 2009, pp. 1–6.

[27] R. Staudemeyer and C. Omlin, "Feature set reduction for automatic network intrusion detection with machine learning algorithms," in *Proceedings of the southern African telecommunication networks and applications conference (SATNAC)*, 2009, p. 105.

[28] R. C. Staudemeyer and C. W. Omlin, "Extracting salient features for network intrusion detection using machine learning methods," *South African computer journal*, vol. 52, no. 1, pp. 82–96, 2014.