A Deep Learning Approach for Network Intrusion Detection System

Quamar Niyaz, Weiqing Sun, Ahmad Y Javaid, and Mansoor Alam
College Of Engineering
The University of Toledo
Toledo, OH-43606, USA
{quamar.niyaz, weiqing.sun, ahmad.javaid, mansoor.alam2}@utoledo.edu

ABSTRACT

A Network Intrusion Detection System (NIDS) helps system administrators to detect network security breaches in their organizations. However, many challenges arise while developing a flexible and efficient NIDS for unforeseen and unpredictable attacks. We propose a deep learning based approach for developing such an efficient and flexible NIDS. We use Self-taught Learning (STL), a deep learning based technique, on NSL-KDD - a benchmark dataset for network intrusion. We present the performance of our approach and compare it with a few previous work. Compared metrics include accuracy, precision, recall, and f-measure values.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Miscellaneous; C.2 [**Computer-Communication Networks**]: Security

Keywords

Network security, NIDS, deep learning, sparse autoencoder, NSL-KDD

1. INTRODUCTION

Network Intrusion Detection Systems (NIDSs) are essential tools for the network system administrators to detect various security breaches inside an organization's network. An NIDS monitors and analyzes the network traffic entering into or exiting from the network devices of an organization and raises alarms if an intrusion is observed. Based on the methods of intrusion detection, NIDSs are categorized into two classes: i) signature (misuse) based NIDS (SNIDS), and ii) anomaly detection based NIDS (ADNIDS). In SNIDS, e.g. Snort [1], attack signatures are pre-installed in the NIDS. A pattern matching is performed for the traffic against the installed signatures to detect an intrusion in the network. In contrast, an ADNIDS classifies network traffic as an intrusion when it observes a deviation from the normal traffic pattern. SNIDS is effective in the detection of known attacks and shows high detection accuracy with less false-alarm rates. However, its performance suffers during detection of unknown or new attacks due to the limitation of attack signatures that can be installed beforehand in an IDS. ADNIDS, on the other hand, is well-suited for the detection of unknown and new attacks. Although ADNIDS produces high false-positive rates, its theoretical potential in the identification of novel attacks has caused its wide acceptance among the research community.

There are primarily two challenges that arise while developing an efficient and flexible NIDS for unknown future attacks. First, proper feature selections from the network traffic dataset for anomaly detection is difficult. The features selected for one class of attack may not work well for other categories of attacks due to continuously changing and evolving attack scenarios. Second, unavailability of labeled traffic dataset from real networks for developing an NIDS. Immense efforts are required to produce such a labeled dataset from the raw network traffic traces collected over a period or in real-time. Additionally, to preserve the confidentiality of the internal organizational network structure as well as the privacy of various users, network administrators are reluctant towards reporting any intrusion that might have occurred in their networks [2].

Various machine learning techniques have been used to develop ADNIDSs, such as Artificial Neural Networks (ANN), Support Vector Machines (SVM), Naive-Bayesian (NB), Random Forests (RF), and Self-Organized Maps (SOM). The NIDSs are developed as classifiers to differentiate the normal traffic from the anomalous traffic. Many NIDSs perform a feature selection task to extract a subset of relevant features from the traffic dataset to enhance classification results. Feature selection helps in the elimination of the possibility of incorrect training through the removal of redundant features and noises [3]. Recently, deep learning based methods have been successfully applied in audio, image, and speech processing applications. These methods aim to learn a good feature representation from a large amount of unlabeled data and subsequently apply these learned features on a limited amount of labeled data in a supervised classification. The labeled and unlabeled data may come from different distributions. However, they must be relevant to each other [4].

It is envisioned that the deep learning based approaches can help to overcome the challenges of developing an efficient NIDS [2, 5]. We can collect unlabeled network traffic data from different network sources and a good feature representation from these datasets using deep learning techniques

.

BICT 2015, December 03-05, New York City, United States

DOI 10.4108/eai.3-12-2015.2262516

can be obtained. These features can, then, be applied for supervised classification to a small, but labeled traffic dataset consisting of normal as well as anomalous traffic records. The traffic data for labeled dataset can be collected in a confined, isolated and private network environment. With this motivation, we use self-taught learning, a deep learning technique based on sparse autoencoder and soft-max regression, to develop an NIDS. We verify the usability of the self-taught learning based NIDS by applying on NSL-KDD intrusion dataset, an improved version of the benchmark dataset for various NIDS evaluations - KDD Cup 99. We provide a comparison of our current work with other techniques as well.

Towards this end, our paper is organized into four sections. In Section 2, we discuss a few closely related work. Section 3 presents an overview of self-taught learning and the NSL-KDD dataset. We discuss our results and comparative analysis in Section 4 and finally conclude our paper with future work direction in Section 5.

2. RELATED WORK

This section presents various recent accomplishments in this area. It should be noted that we only discuss the work that have used the NSL-KDD dataset for their performance benchmarking. Therefore, any dataset referred from this point forward should be considered as NSL-KDD. This approach allows a more accurate comparison of work with other found in the literature. Another limitation is the use of training data for both training and testing by most work. Finally, we discuss a few deep learning based approaches that have been tried so far for similar kind of work.

One of the earliest work found in literature used ANN with enhanced resilient back-propagation for the design of such an IDS [6]. This work used only the training dataset for training (70%), validation (15%) and testing (15%). As expected, use of unlabeled data for testing resulted in a reduction of performance. A more recent work used J48 decision tree classifier with 10-fold cross-validation for testing on the training dataset [7]. This work used a reduced feature set of 22 features instead of the full set of 41 features. A similar work evaluated various popular supervised tree-based classifiers and found that Random Tree model performed best with the highest degree of accuracy along with a reduced false alarm rate [8].

Many 2-level classification approaches have also been proposed. One such work used Discriminative Multinomial Naive Bayes (DMNB) as a base classifier and Nominal-to-Binary supervised filtering at the second level along with 10-fold cross validation for testing [9]. This work was further extended to use Ensembles of Balanced Nested Dichotomies (END) at the first level and Random Forest at the second level [10]. As expected, this enhancement resulted in an improved detection rate and a lower false positive rate. Another 2-level implementation used principal component analysis (PCA) for the feature set reduction and then SVM (using Radial Basis Function) for final classification, resulted in a high detection accuracy with only the training dataset and full 41 features set. A reduction in features set to 23 resulted in even better detection accuracy in some of the attack classes, but the overall performance was reduced [11]. The authors improved their work by using information gain to rank the features and then a behavior-based feature selection to reduce the feature set to 20. This resulted in an improvement in reported accuracy using the training dataset [12].

The second category to look at, used both the training and test dataset. An initial attempt in this category used fuzzy classification with genetic algorithm and resulted in a detection accuracy of 80%+ with a low false positive rate [13]. Another important work used unsupervised clustering algorithms and found that the performance using only the training data was reduced drastically when test data was also used [14]. A similar implementation using the k-point algorithm resulted in a slightly better detection accuracy and lower false positive rate, using both training and test datasets [15]. Another less popular technique, OPF (optimum-path forest) which uses graph partitioning for feature classification, was found to demonstrate a high detection accuracy [16] within one-third of the time compared to SVM-RBF method.

We observed a deep learning approach with Deep Belief Network (DBN) as a feature selector and SVM as a classifier in [5]. This approach resulted in an accuracy of 92.84% when applied on training data. Our current work could be easily compared to this work due to the enhancement of approach over this work and use of both the training and test dataset in our work. A similar, however, semi-supervised learning approach has been used in [2]. The authors used real-world trace for training and evaluated their approach on real-world and KDD Cup 99 traces. Our approach is different from them in the sense that we use NSL-KDD dataset to find deep learning applicability in NIDS implementation. Moreover, the feature learning task is completely unsupervised and based on sparse autoencoder in our approach. We recently observed a sparse autoencoder based deep learning approach for network traffic identification in [17]. The authors performed TCP based unknown protocols identification in their work instead of network intrusion detection.

3. SELF-TAUGHT LEARNING & NSL-KDD DATASET OVERVIEW

3.1 Self-Taught Learning

Self-taught Learning (STL) is a deep learning approach that consists of two stages for the classification. First, a good feature representation is learnt from a large collection of unlabeled data, x_u, termed as Unsupervised Feature Learning (UFL). In the second stage, this learnt representation is applied to labeled data, x_l, and used for the classification task. Although the unlabeled and labeled data may come from different distributions, there must be relevance among them. Figure 1 shows the architecture diagram of STL. There are different approaches used for UFL, such as Sparse Autoencoder, Restricted Boltzmann Machine (RBM), K-Means Clustering, and Gaussian Mixtures [19]. We use sparse autoencoder based feature learning for our work due to its relatively easier implementation and good performance [4]. A sparse autoencoder is a neural network consists of an input, a hidden, and an output layers. The input and output layers contain N nodes, and the hidden layer contains K nodes. The target values at the output layer are set equal to the input values, i.e., $\hat{x}_i = x_i$ as shown in Figure 1(a). The sparse autoencoder network finds the optimal values for weight matrices, $W \in \Re^{K \times N}$ and $V \in \Re^{N \times K}$, and bias vectors, $b_1 \in \Re^{K \times 1}$ and $b_2 \in \Re^{N \times 1}$, using back-

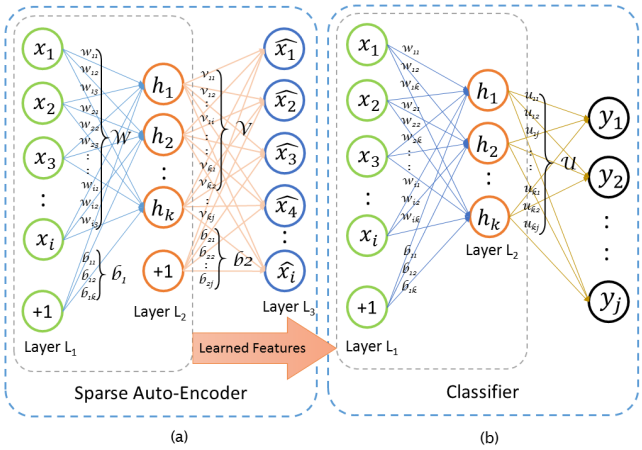

Figure 1: The two-stage process of self-taught learning: a) Unsupervised Feature Learning (UFL) on unlabeled data. b) Classification on labeled data. [18]

propagation algorithm while trying to learn the approximation of the identity function, i.e., output $\hat{x}$ similar to x [18]. Sigmoid function, $g(z) = \frac{1}{1+e^{-z}}$, is used for the activation, $h_{W,b}$ of the nodes in the hidden and output layers:

$$h_{W,b}(x) = g(Wx + b) \tag{1}$$

$$J = \frac{1}{2m}\sum_{i=1}^{m}\|x_i - \hat{x_i}\|^2 + \frac{\lambda}{2}(\sum_{k,n} W^2 + \sum_{n,k} V^2 + \sum_{k} {b_1}^2 + \sum_{n} {b_2}^2) + \beta\sum_{j=1}^{K} KL(\rho\|\hat{\rho}_j) \tag{2}$$

The cost function to be minimized in sparse autoencoder using back-propagation is represented by Eq. (2). The first term is the average of sum-of-square error terms for all m input data. The second term is a weight decay term, with λ as weight decay parameter, to avoid the over-fitting in training. The last term in the equation is sparsity penalty term that puts a constraint into the hidden layer to maintain a low average activation values, and expressed as Kullback-Leibler (KL) divergence shown in Eq. (3):

$$KL(\rho\|\hat{\rho}_j) = \rho log\frac{\rho}{\hat{\rho}_j} + (1-\rho)log\frac{1-\rho}{1-\hat{\rho}_j} \tag{3}$$

where ρ is a sparsity constraint parameter ranges from 0 to 1 and β controls the sparsity penalty term. The $KL(\rho\|\hat{\rho}_j)$ attains a minimum value when $\rho = \hat{\rho}_j$, where $\hat{\rho}_j$ denotes the average activation value of hidden unit j over all training inputs x. Once we learn optimal values for W and b_1 by applying the sparse autoencoder on unlabeled data, x_u, we evaluate the feature representation $a = h_{W,b_1}(x_l)$ for the labeled data, (x_l, y). We use this new features representation, a, with the labels vector, y, for the classification task in the second stage. We use soft-max regression for the classification task as shown in the Figure 1(b) [21].

3.2 NSL-KDD Dataset

As discussed earlier, we used NSL-KDD dataset in our work. The dataset is an improved and reduced version of the KDD Cup 99 dataset [20]. The KDD Cup dataset was prepared using the network traffic captured by 1998 DARPA IDS evaluation program [22]. The network traffic includes normal and different kinds of attack traffic, such as DoS, Probing, user-to-root (U2R), and root-to-local (R2L). The network traffic for training was collected for seven weeks followed by two weeks of traffic collection for testing in raw `tcpdump` format. The test data contains many attacks that were not injected during the training data collection phase to make the intrusion detection task realistic. It is believed that most of the novel attacks can be derived from the known attacks. Finally, the training and test data were processed into the datasets of five million and two million TCP/IP connection records, respectively.

Table 1: Traffic records distribution in the training and test data for normal and attack traffic [20].

Traffic		Training	Test
Normal		67343	9711
Attack	DoS	45927	7458
	U2R	52	67
	R2L	995	2887
	Probe	11656	2421

The KDD Cup dataset has been widely used as a benchmark dataset for many years in the evaluation of NIDS. One of the major drawback with the dataset is that it contains an enormous amount of redundant records both in the training and test data. It was observed that almost 78% and 75% records are redundant in the training and test dataset, respectively [20]. This redundancy makes the learning algorithms biased towards the frequent attack records and leads to poor classification results for the infrequent, but harmful records. The training and test data were classified with the minimum accuracy of 98% and 86% respectively using a very simple machine learning algorithm. It made the comparison task difficult for various IDSs based on different learning algorithms. NSL-KDD was proposed to overcome the limitation of KDD Cup dataset. The dataset is derived from the KDD Cup dataset. It improved the previous dataset in two ways. First, it eliminated all the redundant records from the training and test data. Second, it partitioned all the records in the KDD Cup dataset into various difficulty levels based on the number of learning algorithms that can correctly classify the records. Further, it selected the records by random sampling of the distinct records from different difficulty levels in a fraction that is inversely proportional to their fractions in the distinct records. The multi-steps processing of KDD Cup dataset made the total records statistics reasonable in the NSL-KDD dataset. Moreover, these enhancements made the evaluation of various machine learning techniques realistic.

Each record in the NSL-KDD dataset consists of 41 features[1] and is labeled with either normal or a particular kind of attack. These features include basic features derived directly from a TCP/IP connection, traffic features accumulated in a window interval, either time, e.g. two seconds, or a number of connections, and content features extracted from the application layer data of connections. Out of 41 features, three are nominal, four are binary, and remaining 34

[1]A detailed list of the features can be found at [22]

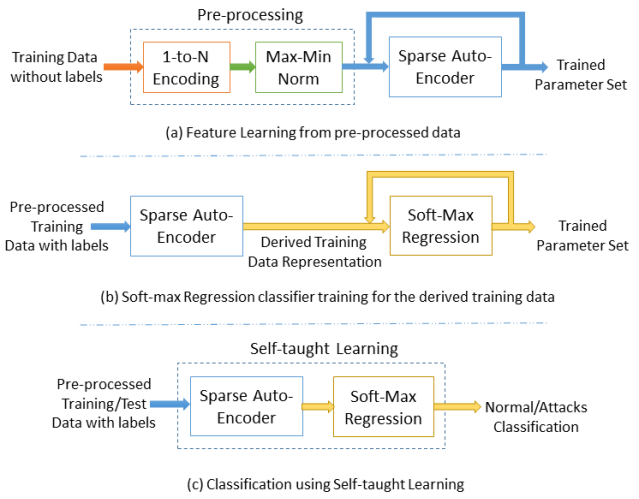

Figure 2: Various steps involved in our NIDS implementation

are continuous. The training data contains 23 traffic classes that include 22 classes of attack and one normal class. The test data contains 38 traffic classes that include 21 attack classes from the training data, 16 novel attacks, and one normal class. These attacks are also grouped into four categories based on the purpose they serve. These categories are DoS, Probing, U2R, and R2L. Table-1 shows the statistics of records for the training and test data for normal and different attack classes.

4. RESULTS AND DISCUSSION

As discussed in Section 2, there are two approaches applied for the evaluation of NIDSs. In the most widely used approach, the training data is used for both training and testing either using *n-fold* cross-validation or splitting the training data into training, cross-validation, and test sets. NIDSs based on this approach achieved very high accuracy and less false-alarm rates. The second approach uses the training and test data separately for the training and testing. Since the training and test data were collected in different environments, the accuracy obtained using the second approach is not as high as in the first approach. Therefore, we emphasize on the results of the second approach in our work for accurate evaluation of NIDS. However, we present the results of the first approach as well for completeness. We describe our NIDS implementation before discussing the results.

4.1 NIDS Implementation

As discussed in the previous section, the dataset contains different kinds of attributes with different values. We preprocess the dataset before applying self-taught learning on it. Nominal attributes are converted into discrete attributes using *1-to-n* encoding. In addition, there is one attribute, *num_outbound_cmds*, in the dataset whose value is always 0 for all the records in the training and test data. We eliminated this attribute from the dataset. The total number of attributes become 121 after performing the steps mentioned above. The values in the output layer during the feature learning phase, shown in Figure 1(a), is computed by the sigmoid function that gives values between 0 and 1. Since the output layer values are identical to the input layer values in this phase, it results in normalization of the values at the input layer in the range of [0, 1]. To obtain this, we perform *max-min* normalization on the new attributes list.

With the new attributes, we use the NSL-KDD training data without labels for feature learning using sparse autoencoder for the first stage of self-taught learning. In the second stage, we apply the newly learned features representation on the training data itself for the classification using soft-max regression. In our implementation, both the unlabeled and labeled data for feature learning and classifier training come from the same source, i.e., NSL-KDD training data. Figure 2 shows the steps involved in our NIDS implementation.

4.2 Accuracy Metrics

We evaluate the performance of self-taught learning based on the following metrics:

- Accuracy: Defined as the percentage of correctly classified records over the total number of records.
- Precision (P): Defined as the % ratio of the number of true positives (TP) records divided by the sum of true positives (TP) and false positives (FP) classified records.

$$P = \frac{TP}{(TP + FP)} \times 100\% \quad (4)$$

- Recall (R): Defined as the % ratio of number of true positives records divided by the sum of true positives and false negatives (FN) classified records.

$$R = \frac{TP}{(TP + FN)} \times 100\% \quad (5)$$

- F-Measure (F): Defined as the harmonic mean of precision and recall and represents a balance between them.

$$F = \frac{2.P.R}{(P + R)} \quad (6)$$

4.3 Performance Evaluation

We implemented the NIDS for three different types of classification: a) Normal and anomaly (2-class), b) Normal and four different attack categories (5-class), and c) Normal and 22 different attacks (23-class). We have evaluated classification accuracy for all types. However, precision, recall, and f-measure values are evaluated in the case of 2-class and 5-class classification. We have computed the weighted values of these metrics in the case of 5-class classification.

4.3.1 Evaluation based on Training data

We applied 10-fold cross-validation on the training data to evaluate the classification accuracy of self-taught learning (STL) for 2-class, 5-class, and 23-class. We also compared its performance with the soft-max regression (SMR) when applied directly to the dataset without feature learning. From Figure 3, we observed that STL shows better performance for 2-class classification as compared to SMR. However, its performance is very similar in the case of 5-class and 23-class classification. We also noticed that STL achieved a classification accuracy rate more than 98% for all types of classification.

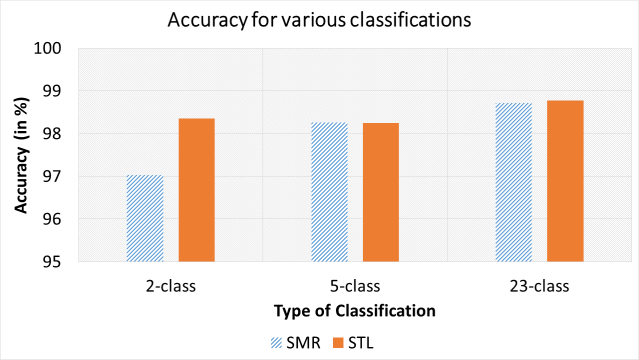

Figure 3: Classification accuracy using self-taught learning (STL) and soft-max regression (SMR) for 2-Class, 5-Class, and 23-Class when applied to training data

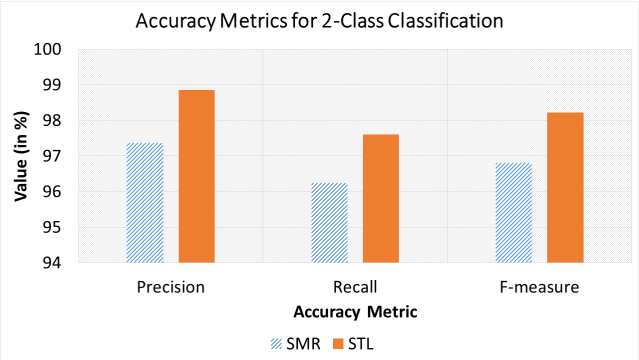

Figure 4: Precision, Recall, and F-Measure values using self-taught learning (STL) and soft-max regression (SMR) for 2-Class when applied to training data

In this case, we evaluated the precision, recall, and f-measure values for only 2-class classification. While performing 10-fold cross-validation, a few kinds of records were missed during the training or test phase for 5-class and 23-class classification. Therefore, we only evaluated these metrics for 2-class. We observed that STL achieved better values for all these metrics as compared to SMR. As shown in the Figure 4, STL achieved 98.84% for f-measure, whereas SMR achieved 96.79%.

Based on the evaluation using training data, we found that performance of STL is comparable to the best results obtained in various previous work.

4.3.2 *Evaluation based on Training and Test Data*

In this case, we evaluated the performance of STL for 2-class and 5-class using the test data. As observed from Figure 5 that STL performs very well as compared to SMR. For the 2-class classification, STL achieved 88.39% accuracy rate, whereas SM achieved 78.06%. The accuracy achieved using STL for a 2-class classification outperforms many of the previous work results. In [20], the best accuracy rate achieved was 82% with NB-Tree. For the 5-class classification, STL achieved an accuracy of 79.10% whereas SM achieved 75.23%.

Figure 6 and Figure 7 show the precision, recall, and f-

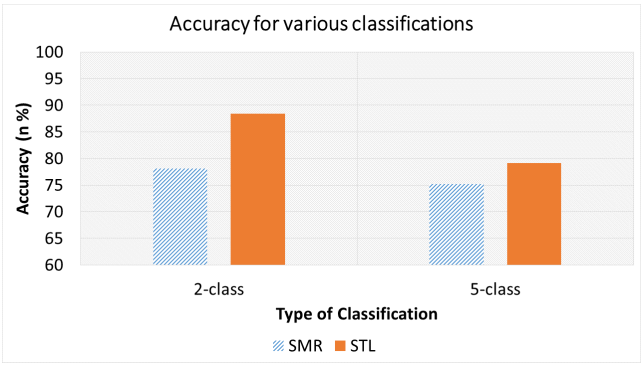

Figure 5: Classification accuracy using self-taught learning (STL) and soft-max regression (SMR) for 2-class and 5-class when applied to test data

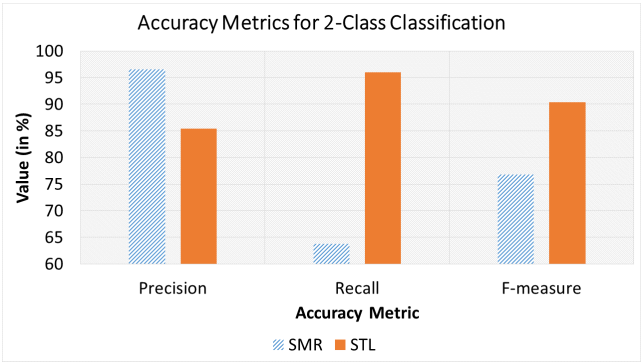

Figure 6: Precision, Recall, and F-Measure values using self-taught learning (STL) and soft-max regression (SMR) for 2-class when applied to test data

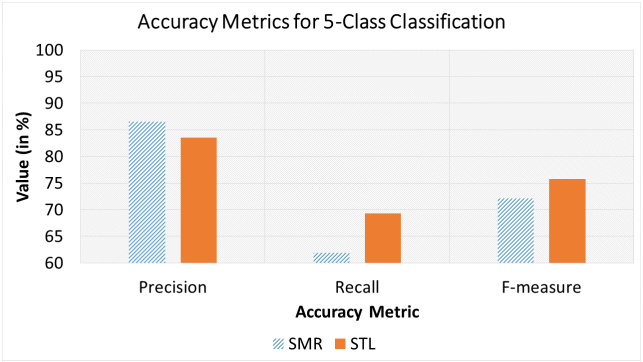

Figure 7: Precision, Recall, and F-Measure values using self-taught learning (STL) and soft-max regression (SMR) for 5-class when applied to test data

measure values for 2-class and 5-class. For the 2-class, STL achieved lesser precision as compared to SM. The precision values for STL and SM are 85.44% and 96.56%, respectively. However, STL achieved better recall values as compared to SM. The recall values for STL and SM are 95.95% and 63.73%, respectively. Due to a good recall value, STL outperformed SM for the f-measure value as well. STL achieved 90.4% f-measure value whereas SM achieved only 76.8%. Similar observations were made for the 5-class as shown in Figure 7. The f-measure values for STL and SM are 75.76% and 72.14%, respectively.

5. CONCLUSION AND FUTURE WORK

We proposed a deep learning based approach for developing an efficient and flexible NIDS. A sparse autoencoder and soft-max regression based NIDS was implemented. We used the benchmark network intrusion dataset - NSL-KDD to evaluate anomaly detection accuracy. We observed that the proposed NIDS performed very well compared to previously implemented NIDSs for the normal/anomaly detection when evaluated on the test data. The performance can be further enhanced by applying techniques such as Stacked Autoencoder, an extension of sparse autoencoder in deep belief nets, for unsupervised feature learning, and NB-Tree, Random Tree, or J48 for further classification. It was noted that the latter techniques performed well when applied directly on the dataset [20]. In future, we plan to implement a real-time NIDS for actual networks using deep learning technique. Additionally, on-the-go feature learning on raw network traffic headers instead of derived features can be another high impact research in this area.

6. REFERENCES

[1] Snort, "https://www.snort.org/."

[2] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network Anomaly Detection with the Restricted Boltzmann Machine," *Neurocomput.*, vol. 122, pp. 13–23, 2013.

[3] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, "Intrusion Detection by Machine Learning: A Review ," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11994 – 12000, 2009.

[4] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught Learning: Transfer Learning from Unlabeled Data," in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, (New York, NY, USA), pp. 759–766, 2007.

[5] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid Intelligent Intrusion Detection Scheme," in *Soft computing in industrial applications*, pp. 293–303, Springer, 2011.

[6] R. S. Naoum, N. A. Abid, and Z. N. Al-Sultani, "An Enhanced Resilient Backpropagation Artificial Neural Network for Intrusion Detection System," *International Journal of Computer Science and Network Security*, vol. 12, no. 3, pp. 11–16, 2012.

[7] H. S. Chae, B. O. Jo, S. H. Choi, and T. K. Park, "Feature Selection for Intrusion Detection using NSL-KDD," *Recent Advances in Computer Science*, pp. 184–187, 2013.

[8] S. Thaseen and C. A. Kumar, "An Analysis of Supervised Tree based Classifiers for Intrusion Detection System," in *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on*, pp. 294–299, IEEE, 2013.

[9] M. Panda, A. Abraham, and M. R. Patra, "Discriminative Multinomial Naive Bayes for Network Intrusion Detection," in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pp. 5–10, IEEE, 2010.

[10] M. Panda, A. Abraham, and M. R. Patra, "A Hybrid Intelligent Approach for Network Intrusion Detection," *Procedia Engineering*, vol. 30, pp. 1–9, 2012.

[11] H. F. Eid, A. Darwish, A. E. Hassanien, and A. Abraham, "Principle Components Analysis and Support Vector Machine based Intrusion Detection System," in *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*, pp. 363–367, IEEE, 2010.

[12] H. F. Eid, M. A. Salama, A. E. Hassanien, and T. H. Kim, "Bi-layer Behavioral-based Feature Selection Approach for Network Intrusion Classification," in *Security Technology*, pp. 195–203, Springer, 2011.

[13] P. Krömer, J. Platoš, V. Snáael, and A. Abraham, "Fuzzy Classification by Evolutionary Algorithms," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pp. 313–318, IEEE, 2011.

[14] I. Syarif, A. Prugel-Bennett, and G. Wills, "Unsupervised Clustering Approach for Network Anomaly Detection," in *Networked Digital Technologies*, pp. 135–145, Springer, 2012.

[15] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and Flow based Network Intrusion Dataset," in *Contemporary Computing*, pp. 322–334, Springer, 2012.

[16] C. R. Pereira, R. Y. Nakamura, K. A. Costa, and J. P. Papa, "An Optimum-Path Forest Framework for Intrusion Detection in Computer Networks," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 6, pp. 1226–1234, 2012.

[17] Z. Wang, "The Applications of Deep Learning on Traffic Identification." `https://goo.gl/WouIM6`.

[18] A. Ng, "Sparse Autoencoder," 2011.

[19] A. Coates, A. Y. Ng, and H. Lee, "An Analysis of Single-layer Networks in Unsupervised Feature Learning," in *International conference on artificial intelligence and statistics*, pp. 215–223, 2011.

[20] M. Tavallaee, E. Bagheri, W. Lu, and 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*, pp. 1–6, July 2009.

[21] Soft-Max Regression, "http://ufldl.stanford.edu/wiki/index.php/softmax_regression."

[22] KDD Cup 99, "http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html."