2018 24th International Conference on Pattern Recognition (ICPR)
Beijing, China, August 20-24, 2018

An Effective Deep Learning Based Scheme for Network Intrusion Detection

Hongpo Zhang[a,b], Chase Q. Wu[c,b], Shan Gao[b], Zongmin Wang[b], Yuxiao Xu[d] and Yongpeng Liu[b]
[a]State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, China
Email: zhp@zzu.edu.cn
[b]Cooperative Innovation Center of Internet Healthcare, Zhengzhou University, Zhengzhou, China
Email: {sgao, zmwang, ypliu}@ha.edu.cn
[c]Department of Computer Science, New Jersey Institute of Technology, Newark, USA
Email: chase.wu@njit.edu
[d]Department of Research and Development, Hangzhou DPtech Technologies Co., Ltd., Hangzhou, China
Email: xuyuxiao@dptech.com

***Abstract*—Intrusion detection systems (IDS) play an important role in the protection of network operations and services. In this paper, we propose an effective network intrusion detection scheme based on deep learning techniques. The proposed scheme employs a denoising autoencoder (DAE) with a weighted loss function for feature selection, which determines a limited number of important features for intrusion detection to reduce feature dimensionality. The selected data is then classified by a compact multilayer perceptron (MLP) for intrusion identification. Extensive experiments are conducted on the UNSW-NB dataset to demonstrate the effectiveness of the proposed scheme. With a small feature selection ratio of 5.9%, the proposed scheme is still able to achieve a superior performance in terms of different evaluation criteria. The strategic selection of a reduced set of features yields satisfactory detection performance with low memory and computing power requirements, making the proposed scheme a promising solution to intrusion detection in high-speed networks.**

***Index Terms*—Intrusion detection system, denoising autoencoder, feature selection, deep learning.**

I. Introduction

Intrusion detection systems (IDS) have been widely deployed and used to safeguard the cyber space. After a continuous technological evolution for over 30 years, modern IDS fall in two main categories based on the detection strategy being used: misuse based and anomaly based [1]. The latter has attracted a great deal of attention in the network research community due to its ability to detect emerging attacks, and a number of anomaly detection schemes have been proposed in the past decades.

Most of the schemes utilize data mining or machine learning techniques to learn the differences between normal and malicious behaviors. Different approaches including decision trees [2], [3], clustering methods [4], association rule mining [5], [6], and support vector machine [7], [8] have been used to implement classifiers for intrusion identification. Statistical models such as language model or hidden Markov method are also employed in several schemes [9], [10]. In these machine learning based schemes, feature selection is often performed to reduce feature dimensionality and improve computing speed. Popular selection approaches include statistical methods based on information gain [2], principal component analysis [11] and K-means clustering [12]. These intrusion detection schemes have their advantages including the compactness of the system and the speed of classification, but at the cost of a relatively low detection accuracy.

More recently, researchers have also explored the application of deep learning methods in IDS to improve intrusion detection performance [13], [14], [15], [16], [17], [18], [19]. In most of these schemes, different deep learning models such as deep belief networks [13], [14], multilayer perceptrons [16], and recurrent neural networks [17], [18], are built to directly distinguish normal traffics and attacks. Another line of research attempts to learn a better representation of original data before classification [20], [21], [22]. In [21], Yousefi *et al.* build a stacked autoencoder to extract features from original data for later classification. Different types of classifiers are tested in their experiments, and the results show that the performance is distinctly improved by the feature learning process, especially when naïve Bayes classifier is used. The work in [22] offers another way for anomaly detection, where a denoising autoencoder (DAE) is employed to characterize the possibility distribution of normal traffic, and then attacks are identified if the traffic is not consistent with the learned distribution. A good detection accuracy is achieved with this method. However, such deep learning based methods require large memory and high computing power in both the data collection and decision making processes. Hence, their applications to high-speed networks are still very limited.

In this paper, we propose an effective intrusion detection scheme based on deep learning techniques, which overcomes the drawbacks of the previous efforts as stated above. The proposed scheme mainly consists of two deep-learning based components to perform feature selection and classification. The selection is performed by a DAE, where a key process is to add weights to its loss function, which helps improve the selection results by placing more emphasis on the attack samples. The classification is then realized by an MLP with a minimized number of parameters while still achieving a high level of performance.

978-1-5386-3788-3/18/$31.00 ©2018 IEEE

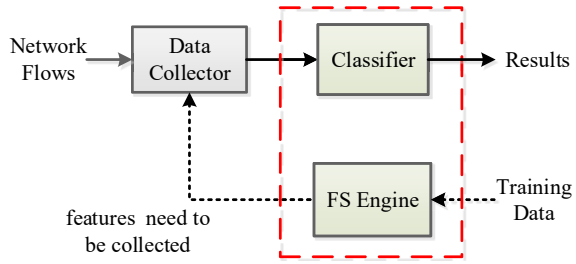


Fig. 1: The schematic diagram of the proposed network IDS scheme, where FS stands for feature selection.

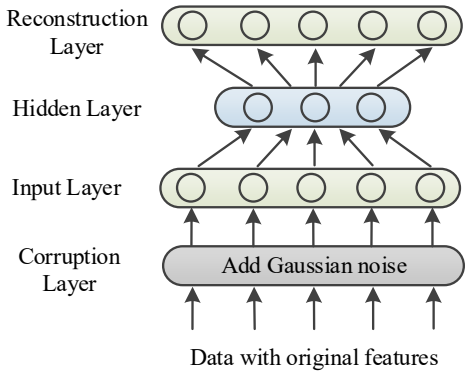


Fig. 2: The structure of a typical denoising autoencoder.

The main advantages of the proposed scheme are as follows. Firstly, the feature selection for IDS is improved by using the weighted loss function. The features that help characterize attack samples are strategically selected, leading to better intrusion detection results. Secondly, the classifier achieves good performance by using deep learning techniques, while the requirements on memory and computing power remain relatively low, as a result of feature selection. With the above advantages, the proposed scheme has a good potential for practical applications in high-speed networks.

The rest of this paper is organized as follows. In Section II, we describe our IDS model in detail and present the principle of operations. The proposed theoretical model is demonstrated by experimental results in Section III. In Section IV, we make a comparison with previous work to show the advantages of our model, and discuss some issues related to the performance. We conclude our work in Section V.

II. MODEL DESCRIPTION

The schematic diagram of the proposed model is shown in Fig. 1. The first component of the system is a data collector, which gathers information and extracts certain features from network flows, according to the results from the second component, i.e., a feature selection engine. This component can be realized by existing mature open-source tools such as Bro, and will not be discussed in this paper. The feature selection engine is trained by pre-collected data to determine which set of features of network flows should be collected. In practical applications, the engine can be improved off-line with newly collected data. The third component is a classifier, which processes the selected feature data and performs intrusion identification.

In this paper, we focus on the two main components of the proposed IDS system: the feature selection engine and the classifier. Their design and operation principles are presented in this section.

A. Feature Selection Engine

The feature selection engine is a critical part of the entire intrusion detection system. A good selection approach should strategically determine and remove redundant features that have a negligible impact on classification accuracy. Towards this end, we build a novel feature selection engine based on a DAE with a weighted loss function. The main advantage of this engine is that it infuses more importance into the features related to the attacks by using the weighted loss function such that the selection results are conducive to the performance improvement of intrusion detection. The design details are provided as follows.

The DAE is a special autoencoder, which receives corrupted data as input and is trained to predict the original data as its output [23], as shown in Fig. 2. In the first stage, the samples $\mathbf{X}$ are corrupted by adding noise with a standard deviation of σ. The corruption process can be described as [24]

$$C(\widetilde{\mathbf{X}}|\mathbf{X}) = N(\widetilde{\mathbf{X}};\, \mu{=}\mathbf{X}, \sum {=}\sigma^2\mathbf{I}), \tag{1}$$

where $\mathbf{I}$ denotes the identity matrix. Then, the distorted samples are sent to the second stage to build a reconstructed version of the original samples. The reconstruction error is characterized by the mean-square error (MSE), calculated as

$$\mathbf{L}_{\text{mse}} = \left\| \mathbf{X} - g(sig(\widetilde{\mathbf{X}}\mathbf{W}_E + \mathbf{b}_E)) \right\|^2, \tag{2}$$

where $g(x)$ and $sig(x)$ denotes the decoding function and sigmoid activation function, respectively, and $\mathbf{W}_E$ and $\mathbf{b}_E$ are the weight matrix and bias vector in the encoding process, respectively. Since each row vector $\mathbf{w}_i$ of $\mathbf{W}_E$ corresponds to the weight of a feature, we can find the most important features by comparing the l_2-norms of the row vectors. To make the weights of the features sparser for a better selection, we use the so-called $l_{2,1}$-norm regularization in the training of DAE [25], [26]. The loss caused by regularization is expressed as

$$L_{reg} = \alpha \cdot \|\mathbf{W}_E\|_{2,1} = \alpha \cdot \sum_i \sqrt{\sum_j |W_{ij}|^2}, \tag{3}$$

where W_{ij} is the element at the i-th row and j-th column of the weight matrix $\mathbf{W}_E$ and α is the coefficient that determines the strength of regularization.

Since the detection of attacks is always more important in the IDS, the features that reflect the characteristics of anomaly traffics should be given more attention. To achieve this goal, the labels of the samples are also used in the feature selection,

different from other autoencoder based schemes [26]. With the labels, we are able to distinguish attacks and normal flows, and then assign different weights to their MSE losses. This way, the DAE are induced to pay more attention to the reconstruction of attack samples during the training. Correspondingly, the feature selection results are inclined to help obtain a more accurate identification of attacks.

If the labels of attacks and normal traffics are set to be '1' and '0', respectively, the weight matrix is calculated as follows:

$$\mathbf{W}_L = (\beta - 1) \cdot \mathbf{Y} + \mathbf{1}, \tag{4}$$

where $\mathbf{Y}$ is the label matrix, $\mathbf{1}$ is a matrix with all the elements having the value of 1, and β is the weight coefficient added to the attack samples. After integrating the weight matrix into the MSE loss, we can obtain the final loss function of the DAE-based feature selector, defined as

$$L_{DAE} = \frac{1}{m}\mathbf{W}_L^T \mathbf{L}_{mse} + L_{reg}, \tag{5}$$

where m denotes the number of samples.

In the selection process, the DAE is trained first to minimize the loss function defined in Eq. 5. We then compare the l_2-norms of the row vectors of $\mathbf{W}_E$ and select k maximum values. The indices of the row vectors are exactly those of the selected features, and are stored as the selection results.

B. Classifier

In the classification process, we simply use a multilayer perceptron (MLP) as the classifier. The choice of using MLP is mainly because the feature dimensionality is significantly reduced after selection. Therefore, an MLP with all fully connected layers is able to sufficiently make use of all the remaining features to mine the hidden information with only a few parameters, hence achieving a good performance without expensive computing. The output layer of the MLP is a sigmoid layer for a binary classification task.

The training target of the classifier is to minimize the cross entropy between the outputs of the network and the labels of the samples. We denote the features and labels as $\mathbf{X}_s$ and $\mathbf{Y}$, respectively, and obtain the loss function of the classifier, expressed as

$$L_{MLP} = H(f_{MLP}(\mathbf{X}_s), \mathbf{Y}) + \lambda \cdot \sum_i \|\mathbf{W}_i\|_2, \tag{6}$$

where $f_{MLP}(x)$ is the transfer function of the MLP and $H(p, q)$ refers to the cross entropy function as defined in [24]. The second term on the right side of Eq. 6 is the l_2-norm regularization term with λ as the coefficient, and $\mathbf{W}_i$ is the weight matrix of the i-th hidden layer.

The training method of the classifier is presented in Algorithm 1. Note that a threshold is also stored with the model after training. It is used in the inference to determine whether a network flow is normal or abnormal. Therefore, a proper value should be the one that can maximize the precision and recall of attack detections, as detailed in lines 11 to 18 of Algorithm 1.

Algorithm 1: Training of an MLP-based classifier

Input: the samples $\mathbf{X}$ with selected features
Output: the parameters of the model and threshold th

1 $th = 0.0$; Build the MLP with a sigmoid output layer;
2 **while** *training* **do**
3 Calculate the $loss$ on the training set according to Eq. 6;
4 Train the MLP using the back propagation method;
5 Evaluate the $test\_loss$ on test set;
6 Record the $prediction$ of classifier on trainset;
7 **if** *test_loss stops decreasing* **then**
8 Store the model and break;
9 **end**
10 **end**
11 **for** *each_th in range(0, 1)* **do**
12 $result = (prediction > each\_th)$;
13 Calculate F_score according to Eq. 9;
14 **if** *F_score > max_F* **then**
15 $max\_F = F\_score$;
16 $th = each\_th$;
17 **end**
18 **end**
19 **return** model parameters and threshold th;

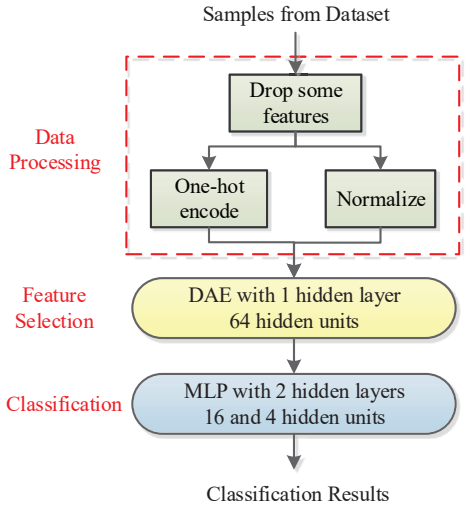


Fig. 3: The experimental demonstration of the proposed IDS.

III. EXPERIMENTS AND RESULTS

We conduct extensive experiments on the UNSW-NB dataset [27] to evaluate the effectiveness of the proposed model and demonstrate its performance superiority over other IDS schemes. These experiments are implemented in Python with Tensorflow Framework of version 1.4.0 and executed on a PC with an i3-6100 processor and 4GB RAM.

The flow chart of the experiments are plotted in Fig. 3. The samples are firstly processed and sent to the DAE based feature selector. In total there are 12 features selected after the selection process. The selected features are then sent to

the MLP based classifier for classification. In the rest of this section, we provide the evaluation metrics and present the details of the experiments.

A. Evaluation Metrics

Commonly used metrics for the performance evaluation of IDS include detection precision, recall, and overall accuracy. Let TP/FP and TN/FN denote the number of samples truely/falsely classified to be positive and negative ones, respectively. These performance metrics can be described as follows

$$pre = TP/(TP + FP), \tag{7}$$

$$rec = TP/(TP + FN), \tag{8}$$

$$accu = (TP + TN)/(TP + TN + FP + FN). \tag{9}$$

Since the precision and recall are both important for an IDS, we use F_score as the final combined evaluation criterion, which is defined as

$$F\_score = 2 \cdot pre \cdot rec/(pre + rec). \tag{10}$$

False positive rate (FPR) is another key indicator concerned in the evaluation of IDS [11]. It reflects the ratio of false alarms, and is calculated as

$$FPR = FP/(FP + TN). \tag{11}$$

B. Dataset Description and Preprocessing

The UNSW-NB dataset is a widely used dataset for the benchmark of network intrusion systems. It is created based on the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) [27], [28]. Compared to other datasets such as NSL-KDD or KDD CUP 99, it is claimed to have the advantages of including modern attacks and normal activities and avoiding the undesired effects caused by the distribution difference between the training and testing sets of the other datasets [27].

In this dataset, there are 2.54 million samples in total, containing 9 types of attack samples and 2.2 million normal samples. Each sample has 47 features. We randomly assign them into two sets for training and testing, respectively, each of which contains 1.905 million and 0.635 million samples. The ratios of normal vs. attack samples of both sets are 6.9, remaining the same as in the original dataset.

The preprocessing procedure contains 3 steps. In the first step, the redundant features are removed, as well as some meaningless features including IP addresses and port numbers. As a result, six original features are dropped. In the second step, we transform the categorical features to vectors using one-hot encoding. For example, a categorical feature with 2 possible values {'type_a', 'type_b'} is encoded by a new vector [1, 0] or [0, 1]. Hence, a 1-D categorical feature is encoded by a 2-D binary one. In the same way, the feature dimensionality is expanded from 41 to 202 after the encoding of our data. In the final step, we standardize the rest scalar features to Gaussian distributions with a mean value of 0 and a standard deviation of 1.

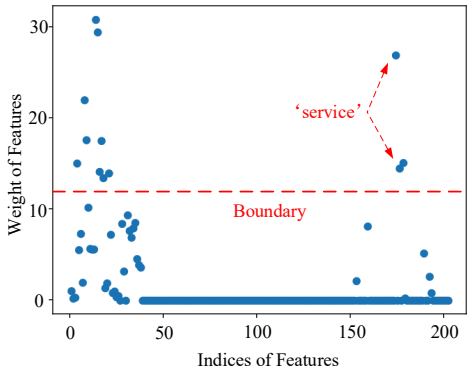


Fig. 4: The weights of the features after training the DAE based feature selector.

TABLE I: The selected features.

'sload', 'dload', 'dmeansz', 'smeansz', 'stcpb', 'dtcpb'
'sttl', 'djit', 'trans-depth', 'service' (unknown, ftp, dns)

C. Feature Selection and Results

Feature selection is realized by a DAE with one hidden layer. The number of hidden units is 64 and the value of the coefficient of class weight β added to the MSE loss is 3.5. In the data corruption layer, a Gaussian noise with a standard deviation of 0.1 is added. After training, we calculate the l_2-norms of the weights of each feature, and the results are presented in Fig. 4, where the horizonal axis denotes the feature index and the vertical axis denotes the l_2-norm of the corresponding feature weight. A feature of more importance has a larger l_2-norm value. We observe that the l_2-norms of most categorical features are nearly zero, except three types of services. With these results, we choose 11 as the boundary value of selected features.

After selection, 12 out of 202 features are chosen, and the selection ratio is 5.9%, which leads to a great reduction in the complexity of the classification system. Their names are presented in Table I . It is also noted that three of the selected ones belong to the same original feature 'service'. Thus only 10 original features are actually used for classification, which is nearly the same number as that in other approaches [2], [6], [11], [29]. However, the selected features are completely different.

D. Classification and Results

It is well known that the performance of an MLP usually increases with the number of parameters, but at the cost of computing power. Therefore, we first implement a deep MLP to maximize the performance, and then reduce the number of hidden layers and hidden units as much as possible while keeping the performance at a satisfactory level. As a result, we choose an MLP with two hidden layers as the classifier, which has 16 and 4 hidden units in the layers, respectively. It is a relatively compact classifier compared to other deep learning

TABLE II: The confusion matrix of the test results.

Category of Samples	Predictions	
	Attack	*Normal*
Attack	75806	4460
Normal	3172	551442

TABLE III: The performance of the proposed IDS.

Category of Samples	Evaluation Metrics		
	Precision	*Recall*	*F_score*
Attack	95.98%	94.43%	0.952
Normal	99.20%	99.43%	0.993
Overall Accuracy	98.80%		

based schemes, where the number of hidden units on the first layer is always over 60 [13], [14], [15], [16], [17], [18], [19]. In the training of the classifier, the regularization coefficient λ is set to be 0.001. The proper threshold value for the attack judgment is found to be 0.55.

The confusion matrix of the test results is shown in Table II. Based on these results, the precision, recall and F_score for both normal traffic and intrusion detection are calculated according to Eq. 7 - Eq. 10, and the results are presented in Table III. The overall detection accuracy is about 98.80%. The precision and recall of anomaly detection are around 95%, and those of normal traffic identification both achieve as high as 99%. These results strongly indicate the effectiveness of the proposed IDS.

IV. DISCUSSIONS

To show the advantages of our IDS scheme, we compare the classification performance with existing work using the same dataset [2], [6], [11], [29], [30]. These existing efforts are all based on statistical or machine learning techniques. It is worth pointing out that the number of selected features is 8 or 10, which are close to those in our experiments (note again that only 10 original features are used in our IDS).

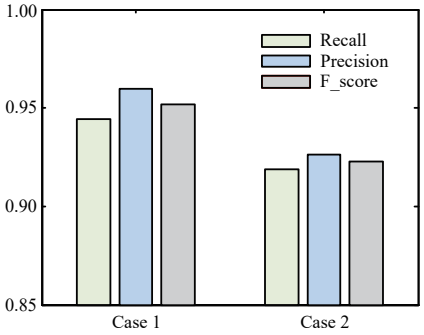
The performance comparison covers all evaluation metrics discussed in Section III-A and the results are presented in Table IV. In the comparison, attack samples are regarded as positive ones. In the table, the placeholder "––" is used to indicate the unavailable results. Obviously, our scheme outperforms all of the others in most aspects, which clearly shows the superiority of the proposed IDS scheme.

We also find that the recall of attack detection is not as good as that of normal flows detection. The reason is that we make use of the entire UNSW-NB dataset for training and testing, where the numbers of samples in different classes are not balanced. The amount of normal samples are about 7 times of that of attack ones, leading the classifier to attach less importance to the precision of attack detection. This problem could be solved by using a class sensitive loss function, upsampling the attack samples or other approaches [31].

In the feature selection, the weight matrix $\mathbf{W}_L$ added to the MSE loss is a key, which leads the selector to select features

TABLE IV: Performance comparison with other IDS schemes.

IDS Schemes	Evaluation Metrics				
	Accuracy	*F_score*	*Precision*	*Recall*	*FPR*
Proposed in [2]	83.00%	0.878	99.00%	71.00%	––
Proposed in [6]	95.98%	––	––	––	––
Proposed in [11]	93.23%	0.947	94.50%	94.89%	9.67%
Proposed in [29]	94.30%	––	––	93.90%	5.80%
Proposed in [30]	96.70%	––	––	95.60%	3.50%
Proposed in this paper	98.80%	0.952	95.98%	94.43%	0.57%



Case 1: Use weighted loss function in feature selection
Case 2: Use unweighted loss function in feature selection

Fig. 5: The comparison of attack detection performance under the two cases when weighted and common loss functions are used in feature selection.

that help identify attacks better. To demonstrate the importance of this process, we implement another experiment without using the weighting process. As a result, the rank of feature weights are changed and finally the feature 'spkts', which means 'source to destination packet count', takes the place of 'sttl' after feature selection. Classification is performed by the same classifier with all the parameters remaining the same. The precision, recall and F_score of the attack detection are shown in Fig. 5, in comparison with the previous experimental results.

The results clearly show that a distinct performance degradation occurs without using the weighted loss function. Both the precision and recall decrease by 3% to 4%, resulting in an evident decrease in the F_score. This fact explicitly indicates that the weighting process to the loss function is a helpful approach to improve the feature selection.

V. CONCLUSION

In conclusion, we proposed an effective deep learning based network IDS scheme in this paper. The IDS mainly consists of a DAE based feature selection engine and an MLP based classifier. One key in the feature selection is adding weights to the loss functions of different samples, which leads the

selector to choose a small set of features that well represent attacks. After selection, only those useful features are retained and a high performance is achieved with a relatively compact classifier.

The performance of the proposed scheme is evaluated by experiments performed on the UNSW-NB dataset, where 12 out of 202 features are selected after feature selection, resulting in a selection ratio of 5.9%. After classification using an MLP with 2 hidden layers, we achieve a high detection accuracy of 98.80%. F_score, which reflects the attack detection performance, also achieves 0.952. Based on a comprehensive comparison with other schemes proposed in the pervious work, we conclude that the proposed scheme is effective for intrusion detection and outperforms the others in terms of the commonly considered evaluation metrics. Taking advantage of relatively low computing resource requirement and high detection performance as illustrated by the experiments, the proposed scheme has shown a good potential for practical applications in high-speed networks.

ACKNOWLEDGMENT

The research is partially supported by the Fund for Integration of Cloud Computing and Big Data, Innovation of Science and Education (FII) under Grant No. 2017A11017.

REFERENCES

[1] H. J. Liao, C. H. R. Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network & Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[2] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques," *arXiv preprint, arXiv:1711.02825*, 2017.

[3] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227–2235, 2011.

[4] R. Perdisci, W. Lee, and N. Feamster, "Behavioral clustering of HTTP-based malware and signature generation using malicious network traces," in *Usenix Conference on Networked Systems Design and Implementation*, pp. 26–26, 2010.

[5] J. J. Treinen and R. Thurimella, "A framework for the application of association rule mining in large intrusion detection infrastructures," in *International Conference on Recent Advances in Intrusion Detection*, pp. 1–18, 2006.

[6] N. Moustafa and J. Slay, "A hybrid feature selection for network intrusion detection systems: Central points," in *Proceedings of the 16th Australian Information Warfare Conference*, pp. 5–13, 2015.

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

[8] S. Rasthofer, S. Arzt, and E. Bodden, "A machine-learning approach for classifying and categorizing android sources and sinks," in *Network and Distributed System Security (Symposium),San Diego, California, USA,*, 2014.

[9] K. Rieck and P. Laskov, "Detecting unknown network attacks using language models," in *the Proceedings of the Third international conference on Detection of Intrusions and Malware & Vulnerability Assessment, Berlin, Germany*, pp. 74–90, 2006.

[10] D. Ariu, R. Tronci, and G. Giacinto, "HMMPayl: An intrusion detection system based on hidden markov models," *Computers & Security*, vol. 30, no. 4, pp. 221–241, 2011.

[11] N. Moustafa, G. Creech, E. Sitnikova, and M. Keshk, "Collaborative anomaly detection framework for handling big data of cloud computing," in *Military Communications and Information Systems Conference*, pp. 1–6, 2017.

[12] U. Ravale, N. Marathe, and P. Padiya, "Feature selection based hybrid anomaly intrusion detection system using K means and RBF kernel function," *Procedia Computer Science*, vol. 45, no. 39, pp. 428–435, 2015.

[13] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *Proceedings of Second International Conference on Advanced Cloud and Big Data, Huangshan, China*, pp. 247–252, 2014.

[14] Z. Alom, V. R. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief network and extreme learning machine," *International Journal of Monitoring and Surveillance Technologies Research*, vol. 3, no. 2, pp. 35–56, 2015.

[15] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China*, pp. 581–585, 2016.

[16] D. Kwon, H. Kim, J. Kim, C. S. Sang, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, no. 5, pp. 1–13, 2017.

[17] C. L. Yin, Y. F. Zhu, J. L. Fei, and X. Z. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[18] A. F. Agarap, "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," *arXiv preprint, arXiv:1709.03082*, 2017.

[19] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, 2013.

[20] 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, New York, United States*, pp. 21–26, 2016.

[21] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, USA*, pp. 3854–3861, 2017.

[22] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *IEEE 4th International Conference on Cyber Security and Cloud Computing, New York, USA*, pp. 193–198, 2017.

[23] G. Alain and Y. Bengio, "What regularized auto-encoders learn from the data generating distribution," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3563–3593, 2014.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[25] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "$\ell_{2,1}$ -norm regularized discriminative feature selection for unsupervised learning," in *International Joint Conference on Artificial Intelligence*, pp. 1589–1594, 2011.

[26] K. Han, C. Li, and X. Shi, "Autoencoder feature selector," *arXiv preprint arXiv:1710.08310*, 2017.

[27] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conference (MilCIS), Canberra, Australia*, pp. 1–6, 2015.

[28] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, vol. 25, pp. 18–31, 2016.

[29] N. Moustafa and J. Slay, "RCNF: Real-time collaborative network forensic scheme for evidence analysis," *arXiv preprint arXiv:1711.02824*, 2017.

[30] N. Moustafa, G. Creech, and J. Slay, "Big data analytics for intrusion detection system: Statistical decision-making using finite dirichlet mixture models," *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*, pp. 127–156, 2017.

[31] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, vol. 21, pp. 1263–1284, 2009.