Deep Learning Combined with De-noising Data for Network Intrusion Detection

Phai Vu Dinh, Tran Nguyen Ngoc
Department for Information Security
Le Quy Don Technical University
Hanoi, Vietnam
dinhphai88@gmail.com, ngoctn@mta.edu.vn

Nathan Shone, Áine MacDermott, Qi Shi
Department of Computer Science
Liverpool John Moores University
Liverpool, UK
{n.shone, a.m.macdermott, q.shi}@ljmu.ac.uk

***Abstract*—Anomaly-based Network Intrusion Detection Systems (NIDSs) are a common security defense for modern networks. The success of their operation depends upon vast quantities of training data. However, one major limitation is the inability of NIDS to be reliably trained using imbalanced datasets. Network observations are naturally imbalanced, yet without substantial data pre-processing, NIDS accuracy can be significantly reduced. With the diversity and dynamicity of modern network traffic, there are concerns that the current reliance upon un-natural balanced datasets cannot remain feasible in modern networks. This paper details our de-noising method, which when combined with deep learning techniques can address these concerns and offer accuracy improvements of between 1.5% and 4.5%. Promising results have been obtained from our model thus far, demonstrating improvements over existing approaches and the strong potential for use in modern NIDSs.**

***Keywords*—de-noising data, deep learning, anomaly detection, auto-encoders, KDD, NSL-KDD, network security.**

I. INTRODUCTION

Despite the many advances made in anomaly-based Network Intrusion Detection Systems (NIDSs), there are still inherent flaws that affect the level of attainable accuracy. One major challenge is the imbalanced and diverse nature of the datasets used to train the detection and classification models used. The imbalance can skew the perspective of behaviour (i.e. malicious behaviour is treated as normal, or vice versa), usually this causes significant problems with detection effectiveness and accuracy. Instead, current approaches rely on data being cleansed or pre-processed beforehand to circumvent such issues.

Data pre-processing provides a shrewd solution to redress the imbalance of the training datasets. This usually this involves a separate process prior to analysis, to remove data outliers or the rebalancing of datasets (e.g. balance the proportion of malicious and normal datum). However, such processes require comparatively high levels of human expert interaction; meaning it is error-prone and time-consuming process [2]. Additionally, pre-processing is not always the most suitable approach, for example the exclusion of low frequency attacks.

As networks continue to increase in dynamicity and complexity, the longevity of behavioural models is reducing. Hence, detection and classification models require frequent retraining, to account for changes. Similarly, since the adoption of shallow machine learning-based methods (e.g. Naive Bayes, Decision Trees and Support Vector Machines (SVM) [1]) into NIDSs research, the sizes of the datasets required have drastically increased. It is therefore no longer feasible to rely on separate processes to restructure and rebalance datasets.

A technique currently receiving substantial interest within NIDS research, is that of deep learning. This is an advanced subset of machine learning, which uses superior layer-wise feature learning to improve upon the performance of shallow learning techniques [3]. It is capable of facilitating a deeper analysis of network data and faster identification of any anomalies. However, to assist in this process a more reliable technique is required to combat the effects of an imbalanced dataset.

In this paper, we propose the use of our de-noising technique, which can be combined with various deep learning techniques. The model is capable of de-noising a wide-range of network traffic to facilitate improved NIDS operation within modern networks. We have evaluated our de-noising model using two popular deep learning techniques, which are stacked auto-encoder (SAE) and deep belief network (DBN) models, which were implemented using GPU-enabled TensorFlow. We used these implementations to analyse the widely-used benchmark NSL-KDD dataset.

This paper offers the following novel contribution:

- A de-noising data method that can be combined with various deep learning techniques to strengthen its classification accuracy. Using the benchmark NSL-KDD data, our de-noising method is able to offer improvements in classification accuracy of between 1.5% and 4.5%.

The remainder of this paper is structured as follows. In Section II, relevant background information is presented. Section III examines existing research within this area and highlights their merits and findings. Section IV specifies our proposed solution, which is subsequently evaluated in Section V. Section VI discusses our findings which are derived from the evaluation. Finally the paper concludes in Section VII.

II. BACKGROUND

A. Deep Learning

Deep learning is an advanced form of machine learning, which moves Machine Learning closer to its ultimate goal of Artificial Intelligence. It allows complex concepts and relationships to be modelled using multiple levels of representation [4]. Supervised and unsupervised learning algorithms are used to build increasingly higher levels of abstraction using the output features from subsequent levels [7].

Auto-encoder: A technique receiving much interest within the field of deep learning. It is an unsupervised feature extraction algorithm, which learns the best parameters required to reconstruct its output as close to its input as possible. This is achieved by applying back propagation and setting the target values to be equal to the inputs. In other words, it is trying to learn an approximation to the identity function. The fact that the structure of the model is so intrinsically linked to the dataset, it is easy to see why the initial balance of the dataset is so important.

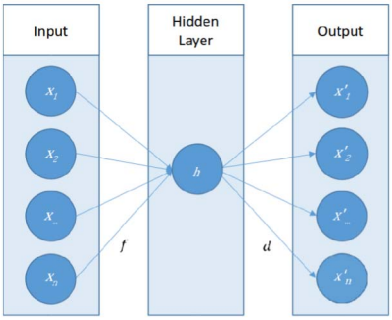

Figure 1. Example Auto-encoder

An auto-encoder usually has an input layer, output layer (which are both of equal dimension) and a hidden layer. This hidden layer normally has a smaller dimension than that of the input. An example of an auto-encoder is illustrated in Fig. 1.

The hidden layer is used to construct a lower dimensionality version of the data (known as encoding). By reducing the dimensionality, the auto-encoder is forced to capture the most prominent features of the data. In an ideal scenario, the data features generated by the auto-encoder will provide a better representation of the data points than the raw data itself.

Stacked auto-encoder: The deep learning construct can be applied to auto-encoders, in a technique known as stacked auto-encoder. Here, the hidden layers represent the simple concepts and multiple hidden layers are used to provide depth. The increased depth can reduce computational costs and the amount of required training data, as well as yielding greater degrees of accuracy [4]. The output from each hidden layer is used as the input for a progressively higher level, which is used to learn increasingly higher level features. An illustrative example of a stacked auto-encoder is shown in Fig. 2.

Within deep learning, there are also Deep Neural Networks (DNNs) and Deep Belief Networks (DBNs). DNNs use unsupervised learning techniques to adjust the weights between hidden layers, allowing the network to identify the best internal representation (features) of the inputs, which enables flexible modelling of the complex and non-linear relationship between the input and output of the network [23]. DBNs are generative graphical models, or class of DNN composed of multiple layers of latent variables or hidden units where there are connections between the layers but not between units within each. DBNs allow unsupervised pre-training over unlabelled samples at first and then a supervised fine-tuning over labelled samples.

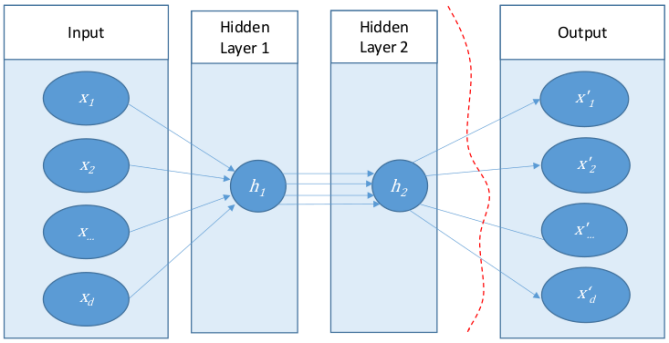

Figure 2. Example stacked auto-encoder

III. EXISTING WORK

Deep learning is increasingly gaining significant interest and its application is being investigated within many research domains, such as: healthcare [6], [7]; automotive design [8], [9]; manufacturing [10] and law enforcement [11], [12]. There are also several existing works within the domain of NIDS. In this section, we will discuss the most current notable works.

The work of Dong and Wang present a literary and experimental comparison between the use of specific traditional NIDS techniques and deep learning methods [1]. The authors concluded that the deep learning-based methods offered improved detection accuracy across a range of sample sizes and traffic anomaly types. The authors also demonstrated that problems associated with imbalanced datasets can be overcome by using oversampling for which, they used the Synthetic Minority Oversampling Technique (SMOTE).

Zhao et al. [2] presented a state-of-the-art survey of deep learning applications within machine health monitoring. They experimentally compared conventional machine learning methods against four common deep learning methods (auto-encoders, Restricted Boltzmann Machine (RBM), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). Their findings concluded that deep learning methods offer better accuracy than conventional methods.

Alrawashdeh and Purdy [13] proposed using a RBM with one hidden layer to perform unsupervised feature reduction. The weights are passed to another RBM to produce a DBN. The pre-trained weights are passed into a fine tuning layer consisting of a Logistic Regression classifier (trained with 10 epochs) with multi-class soft-max. The proposed solution was evaluated using the KDD Cup '99 dataset. The authors claimed a detection rate of 97.90% and a false negative rate of 2.47%. This is an improvement over results claimed by authors of similar papers.

The work by Kim et al. [14] aspired to specifically target advanced persistent threats. They propose a Deep Neural Network (DNN) using 100 hidden units, combined with the Rectified Linear Unit activation function and the ADAM optimiser. Their approach was implemented on a GPU using TensorFlow, and evaluated using the KDD data set. The authors claimed an average accuracy rate of 99%, and summarized that both RNN and Long Short-Term Memory (LSTM) models are needed for improving future defenses.

Potluri and Diedrich [15] propose a method using 41 features and their DNN has 3 hidden layers (2 auto-encoders and 1 soft-max). The results obtained were mixed, those focusing on fewer classes were more accurate than those with more classes. The authors attributed this to insufficient training data for some classes.

Cordero et al. [16] proposed an unsupervised method to learn models of normal network flows. They use RNN, auto-encoder and the dropout concepts of deep learning. The exact accuracy of their proposed method evaluated is not fully disclosed.

Similarly, Tang et al. [17] also propose a method to monitor network flow data. The paper lacked details about its exact algorithms but does present an evaluation using the NSL-KDD dataset, which the authors claim gave an accuracy of 75.75% using six basic features.

Kang and Kang [18] proposed the use of an unsupervised DBN to train parameters to initialise the DNN, which yielded improved classification results (exact details of the approach are not clear). Their evaluation shows improved performance in terms of classification errors.

In addition, there is other relevant work, including the fault monitoring in semiconductor manufacturing as proposed by Lee et al. [19]. They use a Stacked de-noising Auto-encoder (SdA) approach to provide an unsupervised learning solution. A comparison with conventional methods has demonstrated that throughout different use cases the approach increases accuracy by up to 14%. in different use cases. They also concluded that among the SdAs analysed (1-4 layers) those with 4 layers produced the best results.

You et al. [11] propose an automatic security auditing tool for short messages (SMS). Their method is based upon the RNN model. The authors claimed that their evaluations resulted in an accuracy rate of 92.7%, thus improving existing classification methods (e.g. SVM and Naive Bayes).

IV. PROPOSED METHODOLOGY

A. De-noising data

After assessing the NSL-KDD datasets, we know that the five of attacks and normal labels are imbalanced on the training datasets. While the labels of DoS, Probe and normal labels are most prevalent with approximate 99% in total of training datasets, both of two attacks R2L and U2L are small with 1% [12]. Besides, the dimension of this datasets is 41, which is responsible for being not suitable with deep-learning methods that are necessary datasets with high dimension and a huge number of training datasets. In preparation of reducing the drawbacks of above issues in NSL-KDD datasets, we proposed a new method that helps to increase the dimension to 82 and the number of training datasets is higher 5 times while the all labels are binary. To specify, we will represent following several steps below:

Step 1: Given C = {C_1, C_2,.., C_N} are N cluster centers of data train. (N is number of labels on the training data, we can calculate N cluster centers by adding all the records of the same type and then dividing them by the sum of each type).

Step 2:

Input: C = {C_1, C_2,.., C_N}, dataTrain, dataTest, labelTrain, labelTest.

L = {L_1, L_2, …, L_N} are N labels of C.

M is number of data train.

Output: newDataTrain, newDataTest, newLabelTrain, newLabelTest

1. **Set** newDataTrain = Θ, newDataTest = Θ, newLabelTrain = Θ , newLabelTest = Θ, count = 0

2. **For** i = 1 **to** M {

    For k **in** L {

        2.1. tempData = C[k] **concatenate** dataTrain[i]

        2.2. **if** (k == labelTrain[i]) tempLabel = 1

            Else tempLabel = 0

        2.3. newDataTrain[count] = tempData

            newLabelTrain[count] = tempLabel

        2.4 count = count + 1

    }

}

3. We do the same with the testing dataset, followed by getting the remaining datasets of newDataTest and newLabelTest.

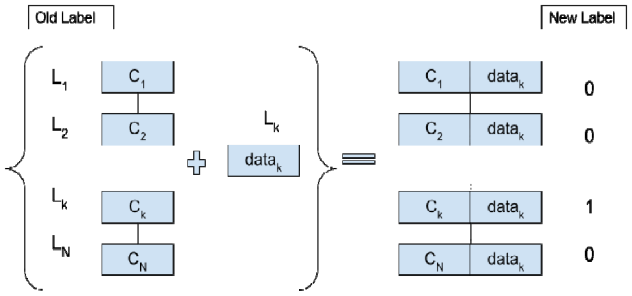

Figure 3. De-noising data method

As the method is presented above, both sets of newLabelTrain and newLabelTest contain binary values. Providing the binary value is 1, it means that data is combined with a cluster center of the similar type. By contrast, the value zero illustrates that the data is combined with a different one. Moreover, the label of data train and data test may separate by many blocks of N consecutive records. For example, if a block is a set of five values of an array B = [0, 0, 0, 1, 0], we can know that the block B is generated by cluster C4 and a record of the similar type. If all values of B are zero, B may be a zero-

day attack. Otherwise, if B has greater equal than 2 value of 1, B is called a confused attack. Therefore, the remaining of the algorithm will utilize these blocks to predict the true label.

B. Classification

Decreasing the reliance on human operators is a crucial requirement for future-proofing NIDSs. Hence, our aim is to devise a technique capable of providing reliable unsupervised feature learning, which can improve upon the performance and accuracy of existing techniques.

Auto-encoders are currently a popular research area. One of their desirable characteristic is their capability to provide more powerful and non-linear generalisation than Principle Component Analysis (PCA). We believe that they offer significant advantages in terms of intrusion detection and are utilised as our starting point. Unlike a simple auto-encoder, a deep auto-encoder is composed of two symmetrical deep-belief networks, which typically have four or five shallow layers for encoding, and a second set of four or five layers for decoding. The work by Hinton and Salacukhudinov [21] has produced promising results by implementing a deep learning algorithm to convert high dimensional data to low dimensional data by utilising a deep auto-encoder.

Most researchers [21], [22] use auto-encoders as a non-linear transformation to discover interesting data structures, by imposing other constraints on the network, and compare the results with those of PCA (linear transformation). These methods are based on the encoder-decoder paradigm. The input is first transformed into a typically lower-dimensional space (encoder), and then expanded to reproduce the initial data (decoder). Once a layer is trained, its code is fed to the next, to better model highly non-linear dependencies in the input. This paradigm focuses on reducing the dimensionality of input data. To achieve this, there is a special layer - the code layer [21], at the centre of the deep auto-encoder structure. This code layer is used as a compressed feature vector for classification or for combination within a stacked auto-encoder [21].

In fulfilment of building up a model that can help solving the NIDSs, we decide to select the auto-encoder to reduce the dimension of datasets, following by utilising the Random Forest for classification. This classification is named to SAE-RF. We use DBNs as a classification for comparison. However, the classifications are implemented with the binary label, therefore, we need to convert this result to N original labels. The method is described below:

Step 1:

Given {predictTest} is set of data test after using SAE-RF to predict. The predicted test contains binary values. We will separate the predicted test into many blocks of N consecutive value.

Given B = {$block_1$, $block_2$, …, $block_M$ } is all blocks that are separated from predicted test. The value M is equal the number of data test.

Step 2:

We will determine the label which is created $block_i$

Input: $block_i$; C = {C_1, C_2,.., C_N} and L = {L_1, L_2, …, L_N} are N labels of C.

Output: label of $block_i$

1. $sum_i = 0$

2. For j=1 to N { sum_i += $block_i[j]$}

3. There are two cases of sum_i:

3.1. **If** (sum_i == 1)

index = -1;

For k =1 to N {

If ($block_i[j]$ == 1) {

Return L[index]

}

}

3.2. **Else**

Given D = {D_1,D_2,..,D_N} and k from 1 to N. (With D_k is Euclidean distance between C_k and newDataTest[i]). We know the $block_i$ belongs to cluster C_k if the value of D_k is the first minimum of set D.

In the 3.2 of the algorithm above, if the value of sum_i is 0, we can understand that the record is zero-day attacks. Otherwise, there are difficult records which can trigger confused classification. Therefore, we can use this method to make a censorship that can support the pre-treatment of the training data.

V. EVALUATION & RESULTS

The classification models were implemented using TensorFlow. All of our evaluations were performed using GPU-enabled TensorFlow running on a 64-bit Ubuntu 16.04 LTS PC with an Intel Xeon 3.60 GHz processor, 16 GB RAM and an NVIDIA GTX 1060 GPU. To perform our evaluations, we have used the NSL-KDD dataset. Both of these datasets are considered as benchmarks within NIDS research. Furthermore, using these datasets assists in drawing comparisons with existing methods and research.

In this section, The following metrics will be used: *True Positive* (TP) - attack data correctly classified as an attack; *False Positive* (FP) - normal data incorrectly classified as an attack; *True Negative* (TN) - normal data correctly classified as normal; *False Negative* (FN) - attack data incorrectly classified as normal.

Additionally, the following measures will be used in the evaluation of our proposed de-noising method:

Accuracy: *(TP+TN) / n* – which measures the proportion of the total number of correct classifications.

Precision: *TP / (TP+FP)* – which measures the number of correct classifications penalised by the number of incorrect classifications.

Recall: *TP / (TP + FN)* – which measures the number of correct classifications penalised by the number of missed entries.

F-score: *2 (Precision* x *Recall) / (Precision + Recall)* – which is used as an effectiveness measurement using the harmonic mean of precision and recall.

TABLE I
COMPOSITION OF DATASETS

Class	Attack Type	No. Training	No. Test
DoS	'back'	956	359
	'land'	18	7
	'neptune'	4124	4657
	'pod'	201	41
	'smurf'	2646	665
	'teardrop'	892	12
Probe	'ipsweep'	3599	141
	'nmap'	1493	73
	'portsweep'	2931	157
	'satan'	3633	735
R2L	'ftp_write'	8	3
	'guess_password'	53	1231
	'imap'	11	1
	'multihop'	7	18
	'phf'	4	2
	'spy	2	0
	'warezclient'	890	0
	'warezmaster'	20	944
U2R	'loadmodule'	9	2
	'buffer_overflow'	30	20
	'rootkit'	10	13
	'perl'	3	2
Normal		67343	9711
Total		**125973**	**18794**

TABLE II
DBN – NORMAL DATA VS DE-NOISING DATA

		Normal	DoS	Probe	R2L	U2R	Total
Accuracy (%)	Normal DBN	95.64	87.96	72.97	0.00	0.00	80.58
	Denoising DBN	99.92	92.23	58.23	**19.87**	**18.92**	**85.59**
Precision (%)	Normal DBN	100.00	100.00	100.00	0.00	0.00	88.10
	Denoising DBN	100.00	100.00	100.00	**100.00**	**100.00**	**100.00**
Recall (%)	Normal DBN	95.64	87.96	72.97	0.00	0.00	80.58
	Denoising DBN	99.92	92.23	58.23	**19.87**	**18.92**	**85.59**
F-score (%)	Normal DBN	97.77	93.60	84.37	0.00	0.00	84.08
	Denoising DBN	99.96	95.96	73.60	**33.16**	**31.82**	**89.24**

TABLE III
DBN - CONFUSION MATRIX FOR NORMAL DATA

Attack Class	Normal	DoS	Probe	R2L	U2R
Normal	9288	312	111	0	0
DoS	634	5050	57	0	0
Probe	127	172	807	0	0
R2L	2182	3	14	0	0
U2R	37	0	0	0	0

TABLE IV
DBN - CONFUSION MATRIX ON DE-NOISING DATA

Attack Class	Normal	DoS	Probe	R2L	U2R
Normal	9703	8	0	0	0
DoS	446	5295	0	0	0
Probe	2	460	644	0	0
R2L	1758	4	0	437	0
U2R	30	0	0	0	7

TABLE V
SAE-RF – NORMAL DATA VS DE-NOISING DATA

	Data Type	Normal	DoS	Probe	R2L	U2R	Total
Accuracy (%)	Normal SAE-RF	94.58	97.73	94.67	3.82	2.70	85.42
	Denoising SAE-RF	97.67	96.06	97.47	**8.09**	**13.51**	**86.52**
Precision (%)	Normal SAE-RF	100.00	100.00	100.00	100.00	100.00	100.00
	Denoising SAE-RF	100.00	100.00	100.00	100.00	100.00	100.00
Recall (%)	Normal SAE-RF	94.58	97.73	94.67	3.82	2.70	85.42
	Denoising SAE-RF	97.67	96.06	97.47	**8.09**	**13.51**	**86.52**
F-score (%)	Normal SAE-RF	97.22	98.85	97.26	7.36	5.26	87.37
	Denoising SAE-RF	98.82	97.99	98.72	**14.98**	**23.81**	**88.60**

TABLE VI
SAE-RF - CONFUSION MATRIX ON NORMAL DATA

Attack Class	Normal	DoS	Probe	R2L	U2R
Normal	9491	58	158	3	1
DoS	260	5430	51	0	0
Probe	58	1	1047	0	0
R2L	2112	0	3	84	0
U2R	36	0	0	0	1

TABLE VII
SAE-RF - CONFUSION MATRIX ON DE-NOISING DATA

Attack Class	Normal	DoS	Probe	R2L	U2R
Normal	9485	52	169	4	1
DoS	223	5515	3	0	0
Probe	28	0	1078	0	0
R2L	2019	0	2	178	0
U2R	32	0	0	0	5

VI. DISCUSSION

Our evaluations show that our proposed de-noising data method combined deep learning algorithms has produced a promising set of results. With regards to the DBN classification, we can see that utilizing the de-noising data method combined deep learning techniques yields significantly improved results when compared to the normal datasets using deep learning techniques. For instance, while the latter one has the total accuracy on the data attacks is 80.85%, the former one experiences a higher result with 85.59% on Table II.

Moreover, the significant result is also shown on the F-score measurement with an approximate 5%. When it comes to the confusion matrix, the remarkable performance of de-noising data method and normal data is illustrated Table III and Table IV. The first table depicts an imbalance of attack detection result on total labels since this algorithm cannot detect any attacks under the labels R2L and U2R. By contrast, the two labels R2L and U2R is detected more effectively on Table IV, which is responsible for a higher accuracy measurement about 19% and more than 30% of F-score measurement on Table II.

In terms of the SAE-RF classification, both accuracy and F-score measurement of de-noising combined deep learning data method are higher than normal datasets using deep learning on Table V. To specify, the total accuracy of the latter one is 85.42% while the former one is 86.52%. In addition, the higher value 1% is presented on the F-score measurement during comparison between the first one and the last one. We can use

Table VI and Table VII to explain why the first one is better the last one. Although the number of true positive records of three labels such as Normal, DoS and Probe is quite the same on both Table VI & VII, the last two labels R2L and U2R of de-noising data method are higher than the normal data. This may help us explain why the de-noising data method can contribute to the results of detection more balanced.

VII. CONCLUSION & FUTURE WORK

In this paper, we have outlined the imbalanced dataset problems faced by NIDSs. To address this, we have devised the de-noising data method presented. Our paper has demonstrated how the method can be combined with several common deep learning methods and offer improvements. We have implemented our method using TensorFlow and evaluated its capabilities against the benchmark NSL-KDD dataset. Our results have demonstrated that by using our de-noising method offers improvements to both the SAE and DBN deep learning techniques evaluated. Our method was able to offer improvements of up to 4.5% across the measures of accuracy, precision, and recall. Most notably, the detection accuracy is significantly more balanced across the 5 NSL-KDD dataset labels (especially in the R2L and U2R labels).

In our future work, we will evaluate a more comprehensive selection of deep learning techniques to ascertain which offers the best performance, when partnered with our de-noising model. We will then look to enhance our current evaluations by utilizing real-world (and therefore unbalanced) network traffic to demonstrate the merits of our method.

ACKNOWLEDGMENT

The authors would like to thank the Royal Academy of Engineering for their support provided through the Newton Research Collaboration Programme.

REFERENCES

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

[2] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep Learning and Its Applications to Machine Health Monitoring: A Survey,” submitted to IEEE Transactions on Neural Networks and Learning Systems, vol. 14, no. 8, pp. 1–14, December 2016. [Online]. Available: http://arxiv.org/abs/1612.07640

[3] S. Hou, A. Saas, L. Chen, and Y. Ye, “Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs,” in 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW). Omaha, Nebraska, USA: IEEE, October 2016, pp. 104–111.

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016, http://www.deeplearningbook.org.

[5] L. Deng, “Deep Learning: Methods and Applications,” R in Signal Processing, vol. 7, no. 3-4, pp. Foundations and Trends 197–387, August 2014.

[6] Z. Liang, G. Zhang, J. X. Huang, and Q. V. Hu, “Deep learning for healthcare decision making with EMRs,” in 2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), no. Cm. IEEE, November 2014, pp. 556–559. [Online]. Available: http://ieeexplore.ieee.org/document/6999219/

[7] S. P. Shashikumar, A. J. Shah, Q. Li, G. D. Clifford, and S. Nemati, “A deep learning approach to monitoring and detecting atrial fibrillation using wearable technology,” in 2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI). Florida, USA: IEEE, 2017, pp. 141–144.

[8] F. Falcini, G. Lami, and A. M. Costanza, “Deep Learning in Automotive Software,” IEEE Software, vol. 34, no. 3, pp. 56–63, May 2017.

[9] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, “Deep learning in the automotive industry: Applications and tools,” in 2016 IEEE International Conference on Big Data (Big Data). IEEE, December 2016, pp. 3759–3768.

[10] H. Lee, Y. Kim, and C. O. Kim, “A Deep Learning Model for Robust Wafer Fault Monitoring With Sensor Measurement Noise,” IEEE Transactions on Semiconductor Manufacturing, vol. 30, no. 1, pp. 23–31, February 2017.

[11] L. You, Y. Li, Y. Wang, J. Zhang, and Y. Yang, “A deep learning-based RNNs model for automatic security audit of short messages,” in 2016 16th International Symposium on Communications and Information Technologies (ISCIT). Qingdao, China: IEEE, September 2016, pp. 225–229.

[12] S. Revathi , Dr. A. Malathi, “A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection”, International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 2 Issue 12, December - 2013

[13] K. Alrawashdeh and C. Purdy, “Toward an Online Anomaly Intrusion Detection System Based on Deep Learning,” in 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA). Anaheim, California, USA: IEEE, December 2016, pp. 195–200.

[14] Jin Kim, Nara Shin, S. Y. Jo, and Sang Hyun Kim, “Method of intrusion detection using deep neural network,” in 2017 IEEE International Conference on Big Data and Smart Computing (BigComp). Hong Kong, China: IEEE, February 2017, pp. 313–316.

[15] S. Potluri and C. Diedrich, “Accelerated deep neural networks for enhanced Intrusion Detection System,” in 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), vol. 2016-Novem. Berlin, Germany: IEEE, September 2016, pp. 1–8.

[16] C. Garcia Cordero, S. Hauke, M. Muhlhauser, and M. Fischer, “Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks,” in 2016 14th Annual Conference on Privacy, Security and Trust (PST). Auckland, New Zealand: IEEE, December 2016, pp. 317–324.

[17] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep learning approach for Network Intrusion Detection in Software Defined Networking,” in 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM). Fez, Morocco: IEEE, October 2016, pp. 258–263.

[18] M.-J. Kang and J.-w. Kang, “Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security,” PLOS ONE, vol. 11, no. 6, p. e0155781, June 2016.

[19] H.-W. Lee, N.-r. Kim, and J.-h. Lee, “Deep Neural Network Self-training Based on Unsupervised Learning and Dropout,” The International Journal of Fuzzy Logic and Intelligent Systems, vol. 17, no. 1, pp. 1–9, mar 2017.

[20] Y. Wang, W.-D. Cai, and P.-C. Wei, “A deep learning approach for detecting malicious JavaScript code,” Security and Communication Networks, vol. 9, no. 11, pp. 1520–1534, July 2016.

[21] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” science, vol. 313, no. 5786, pp.504–507, 2006.

[22] Y. Wang, H. Yao, and S. Zhao, “Auto-encoder based dimensionality reduction,” Neurocomputing, vol. 184, pp. 232–242, 2016.

[23] F. Movahedi, J. L. Coyle, and E. Sejdić. "Deep belief networks for electroencephalography: A review of recent contributions and future outlooks." IEEE Journal of Biomedical and Health Informatics (2017).