2016 15th IEEE International Conference on Machine Learning and Applications

Toward an Online Anomaly Intrusion Detection System Based on Deep Learning

Khaled Alrawashdeh, Carla Purdy
Department of Electrical Engineering and Computing Systems
University of Cincinnati
Cincinnati, OH, USA
Email: alrawakm@mail.uc.edu, purdycc@ucmail.uc.edu

***Abstract*—In the past twenty years, progress in intrusion detection has been steady but slow. The biggest challenge is to detect new attacks in real time. In this work, a deep learning approach for anomaly detection using a Restricted Boltzmann Machine (RBM) and a deep belief network are implemented. Our method uses a one-hidden layer RBM to perform unsupervised feature reduction. The resultant weights from this RBM are passed to another RBM producing a deep belief network. The pre-trained weights are passed into a fine tuning layer consisting of a Logistic Regression (LR) classifier with multi-class soft-max. We have implemented the deep learning architecture in C++ in Microsoft Visual Studio 2013 and we use the DARPA KDD-CUP'99 dataset to evaluate its performance. Our architecture outperforms previous deep learning methods implemented by Li and Salama in both detection speed and accuracy. We achieve a detection rate of 97.9% on the total 10% KDDCUP'99 test dataset. By improving the training process of the simulation, we are also able to produce a low false negative rate of 2.47%. Although the deficiencies in the KDDCUP'99 dataset are well understood, it still presents machine learning approaches for predicting attacks with a reasonable challenge. Our future work will include applying our machine learning strategy to larger and more challenging datasets, which include larger classes of attacks.**

***Index Terms*—Intrusion Detection, DBN, RBM, Attacks, Cyber Security, Neural Network, Machine Learning, Data Mining.**

I. Introduction

Intrusion detection systems methods can be divided into two main categories: signature based intrusion detection and anomaly or misused intrusion detection. Recent years have witnessed an increase in the number of attacks on computer networks, SCADA infrastructures, and the internet of things (IoT). Despite the differences between each of these sectors, they all share connection points and data. The inter-connectivity between these sectors and their reliance on software as explained by Ozsoy [1] have produced many vulnerabilities for intruders to exploit. Concurrently, anomaly based intrusion detection systems are being used more frequently to protect against known attacks. But the challenges for cyber security remain severe due to the continual increase in the number of zero-day attacks for which conventional intrusion detection systems cannot provide protection. The majority of the cyber security measures in the IoT and in general purpose computing rely on signature based detection systems, whether anti-virus or intrusion detection systems. Signature based detection methods are less effective due to the growing number of new cyber attacks. In this paper we introduce a method for anomaly detection based on combining the efficient learning methods of a deep belief network and a logistic regression classifier. Deep learning methods based on neural networks have gained interest recently beacuse deep learning algorithms are able to perform self-learning by extracting features from unlabeled data. Examples of such algorithms are the Restricted Boltzmann Mashine (RBM) and the AutoEncoder algorithms. Both algorithms can extract important features from unlabeled data as described by Hinton et al. [2] and by Bengio et al. [3] when the weights are initialized properly. This is a very important advantage because most of the data in real life is unlabeled and in many cases has a large number of features, sometimes as many as millions [3], [4]. The self-learning of the deep learning network is a form of unsupervised learning where the network can extract important features without human involvement. The self-learning advantage in deep learning is essential in the design of an online intrusion detection system. Deep learning can also include a semi-supervised layer of training, such as a back propagation layer, to fine tune the extracted features, thereby increasing the detection rate.

Our method implements a deep network architecture that is trained in an unsupervised manner to extract important features from the dataset. We train the deep learning network layer by layer, using 10 epochs for each layer. Then a classifier based on the logistic regression algorithm is added as a fine-tuning stage. We train the network in the logistic regression layer using 10 epochs. As a result, we obtain a detection accuracy rate of 97.9% on the total 10% testing set of the KDDCUP'99 dataset. This compares with previous methods such as those used in [5], which reported a maximum accuracy of 92.10% by testing on selected records from the dataset.

We construct an RBM-based deep belief network using C++ Visual Studio to train the dataset. To verify our approach we perform statistical analysis on the KDDCUP'99 dataset and identify the potentials of the deep learning method in

978-1-5090-6167-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ICMLA.2016.167

intrusion detection.

The KDDCUP'99 dataset from which we obtain our training and testing data suffers from several defects that previous researchers have pointed out. For example in [6], the authors explained that the KDDCUP'99 dataset lacks new attack classes and contains duplicate records. The machine learning approach we implement enables us to address some of these issues and to minimize their effects during training and testing. The majority of the improved datasets used in intrusion detection systems research are also based on the KDDCUP'99 dataset. For example, one improved dataset is the NSL-KDD [6] which is an improved version of the KDDCUP'99 dataset and the Kyoto datset [7] from Kyoto University. But they still suffer from deficiencies because they don't profile a real life network.

The rest of this paper is divided into six sections. In section two we review previous work on intrusion detection using machine learning. We also review previous methods that used the KDDCUP'99 dataset and summarize their results. The third section is a detailed description of the deep learning algorithm that we implement in this work. In the fourth section we perform statistical analysis and pre-processing on the dataset. The fifth section describes the experimental setup and the steps for evaluating the performance of the method. In the sixth section we analyze the results and compare them to previous detection methods. The seventh section gives an overall evaluation of our results and plans for future work.

II. Intrusion Detection Systems and Related Work

Machine learning based intrusion dtetection is a valuable approach to protect against the drasic increase in zero-day attacks targeting computer networks. Intrusion detection is especially important in distributed systems such as the internet of things (IoT) and SCADA infrastructure as described in [8],[9],[10]. Moreover, the automotive industry has seen an increase in the number of vulnerabilities in their newly adapted connected technologies that can be exploited by intruders[11]. Previous work in network intrusion detection produced limited success due to the lack of suitable datasets that can evaluate real life attacks for detection methods. Moreover, intrusion detection systems lack the speed to guard against attacks before they cause damage as discussed by Das el al. [9].

Deep learning methods have produced highly accurate results in image recognition applications. The learning algorithms described by Hinton et al. [12] in their 2006 paper produced state of the art results on the MNIST dataset for hand written character classification. The core design of the deep network is the use of an RBM in an unsupervised layer by layer greedy learning method. Recently, researchers began experimenting with deep learning for intrusion detection. Examples of this work include results of Lane and Georgiev [13] and Haque and Alkharobi [14]. The work by Li et al. [5] has produced promising results by implementing a hybrid deep learning method for intrusion detection. In [5] Li et al. used a deep belief network and AutoEncoder to train their systems using the KDDCUP'99 dataset. Despite the moderate results obtained, their method requires lengthy pre-processing. Moreover, Li et al. tested 2000 records of the dataset, but this does not provide an adequate evaluation of the performance of the method as explained by Ghorbani et al. in [6]. In [15] Salama et al. used a deep belief network with an RBM to train on the NSL-KDD dataset. They used a Support Vector Machine (SVM) as the classifier to detect the type of attacks in the dataset. Their work failed to identify the performance of the deep learning algorithms on the different classes of attacks in the dataset.

III. Deep learning algorithms

The RBM is the building block of the deep belief network. The connections between the neurons in the visible and hidden layers can only be between layers as shown in Figure 1. Each neuron stores weight calculations that take place in each layer. Each node can transmit the input weight in a random process using a randomly generated stochastic coefficient.

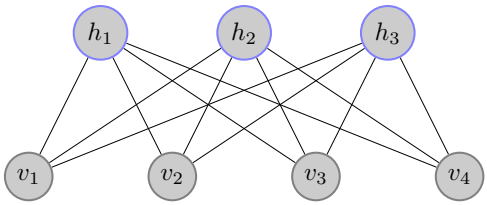

Fig. 1: Restricted Boltzmann Machine (RBM)

- For each visible unit $v = (v_1, v_2, ..., v_n)$ and for each hidden unit $h = (h_1, h_2, ...h_m)$ for a given RBM, the energy function can be defined as follows:

$$Energy(v, h) = -b^T h - c^T v - hWv^T \tag{1}$$

  where b, c are offsets/biases and W is the weight matrix connecting the neurons.
- The joint probability of (v,h) is given by

$$P(v, h) = \frac{1}{Z} e^{-Energy(v,h)} \tag{2}$$

  where Z is the normalization term, introduced to restrict the result to lie between 0 and 1, because this represents a probability in the formula above as noted by Hinton et al. in [12].
- We compute the conditional probability for each visible unit given the hidden unit by

$$P(v_i = 1|h) = \frac{P(v_i = 1, h)}{P(v_i = 0, h) + P(v_i = 1, h)} \tag{3}$$

$$= \frac{e^{c_i + W_i v}}{1 + e^{c_i + W_i v}} = sigm(c_i + W_i v) \quad (4)$$

- Conditional probabilities for hidden units can be obtained similarly

$$P(h_j = 1|v) = sigm(b_j + W'_j h) \quad (5)$$

- The initial sample in Gibbs sampling can be random values [12]
- We start with training examples because we want $P(v) \approx P_{train}(v)$
- Contrastive divergence does not wait for the chain to converge. In order to compute the likelihood and its gradient, Gibbs sampling is used to reduce the computational complexity as described by Fischer and Igel [16]
- We apply Gibbs sampling for only k steps, where in practice $k = 1$ has been proved to work well [3]

A. *Deep Belief Network*

The Deep Belief Network (DBN) can be formed by stacking several RBMs, where the top RBM has undirected connections between the visible and hidden units [3].

- Top most layer is RBM as shown in Figure 2
- Others are directed belief networks

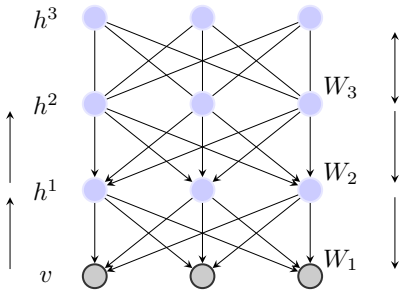

Fig. 2: Deep Belief Network (DBN)

IV. Experimental Setup

We implemented the algorithm using C++ and Microsoft Visual Studio 2013 to simulate the detection method [17]. Using C++ enables dynamic memory allocation that speeds up the training and testing compared to the Matlab environment which is typically used. Reducing the training and testing time for deep learning is an essential part of any online detection system. The first step is to implement the first RBM and approximate the maximum learning likelihood of the data and the probability of the network output using Gibbs sampling. Contrastive divergence is then implemented to efficiently train the network.

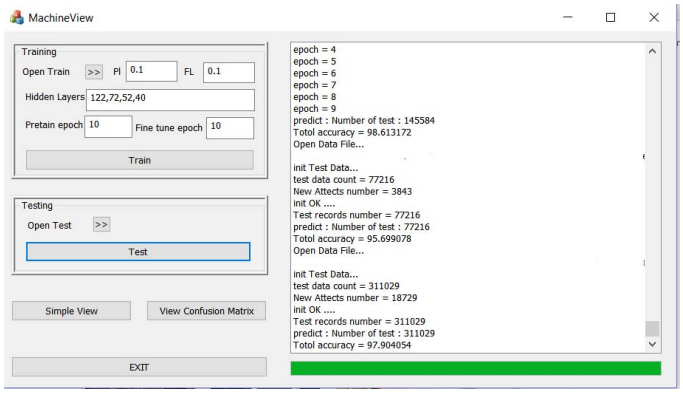

Fig. 3: Simulation results for the network (User Interface)

The RBM is the pre-training layer that takes the input with the full features and applies parallel sampling between the visible and the hidden units. The training proceeds as follows:

- Train the first RBM of the first layer of the deep belief network using contrastive divergence [12]. Using contrastive divergence we obtain an estimate using Gibbs sampling. We run the chain for k steps [18]. Each step consists of sampling h(t) from the probability of the hidden unit given the visible unit value $p(h|v(t))$ and sampling v(t+1) from $p(v|h(t))$.
- The RBM adjusts the weights between each iteration in order to allow the network to learn a high representation of the data. The RBM is able to learn the features from the input data and pass the learned reduced features to the next layer in the network.
- The first layer RBM output is the input of the second layer that is a directed RBM as shown in Figure 2.
- Then we fine tune the network by implemeting a semi-supervised learning method which is a combination of supervised and unsupervised learning. To achieve that we add a set of hidden units that correspond to the classification labels and implement a variation of the wake-sleep algorithm [12].
- The RBM-based deep belief network feeds the resultant data to a layer that consists of logistic regression with soft-max classifier to classify more than one class of attacks in the training and testing data.
- We design the application with a user interface (UI) that enables us to change the number of hidden layers and the learning rate for both the pre-training and fine-tuning layer as shown in Figure 3

Based on our experiment, we determined that a logistic regression classifier outperforms back propagation in the fine-tuning layer. Our results show a high accuracy rate that outperforms previous deep learning methods using feature reduction property of the deep learning.

V. DATASET

We simulate the deep learning approach on a subset of 1000 records from the CSIC 2010 HTTP Dataset that contains 18 attributes with two output classes normal and anomaly [19]. Using the RBM and DBN learning method we also allow the network to learn the characters in each record in the dataset in addition to the 18 features. Preliminary results show 99% accuracy rate. The dataset lacks larger classes of attacks, therefore we decided to simulate the deep learning approach for intrusion detection using the KDDCUP99 dataset. Although this dataset suffers from deciencies such as the lack of new attacks and the presence of duplicate records, it is still widely used to demonstrate the efciency of the intrusion detection concepts

- The KDDCUP'99 dataset was generated in 1999 in a collaboration by DARPA and MIT Lincoln Lab [6]
- The 10% set contains 494,021 training records and 311029 testing records
- It is common practice to use 10% of the original data because this represents the original data and also allows for reduced computation [5]

The dataset contains four classes of attacks, with each class having a variable number of example attacks. In [6] Tavallaee et al. provide a detailed analysis of the KDDCUP'99 data set and the attack types according to the list below:

1) Denial of Service Attack (DoS)
2) User to Root Attack (U2R)
3) Remote to Local Attack (R2L)
4) Probing Attack

A. Data statistics and prerocessing

Training the RBM is best achieved for data values in the range 0 to 1 in order for the network to train properly [12]. We use the testing data to test the network and to validate the results. The testing data in the KDDCUP'99 contains 17 attacks that are not in the training data. In order to train the raw data using deep learning, we convert the nonnumeric columns into numerical values [5].

- We convert the three protocols into binary values as follows: tcp, udp, icmp (1,0,0),(0,1,0),(0,0,1).
- Services are 70:
  " 'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'daytime', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'efs', ... 'Z39_50'"
- Flags are 11: " 'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH' "
  The total dataset features are derived from adding 3 protocols + 70 Services +11 flags + 38 numerical features= 122. We normalized and standardized the numerical values according to equations 6 and 7.

$$X(i, std) = \frac{(x_i - mean(x_i))}{standardDeviationOf(x_i)} \tag{6}$$

In equation (6) X_i is the given feature to be standardized and normalized. This transforms X_i into a vector with unit standard deviation

$$X_(i, norm) = \frac{x_(i, std) - min_(x_(i, std))}{max_(x_(i, std)) - min_(x_(i, std))} \tag{7}$$

In equation (7) the input variables will transform to a vector whose minimum value is 0 and maximum value is 1.

One major issue in the data is the large number of duplicate records in both the training and testing dataset. Neural network classifiers can be biased toward classes with the highest number of instances as noted by Tavallee in [6]. Another potential problem that can downgrade the performance of the learning algorithm is the presence of different attacks that have the same features. We give an example for two records:

- Record 1:
  "0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
  1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,
  0.00,0.00,0.00,0.00,0.00,0.00,normal."

- Record 2:
  "0,udp,private,SF,105,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  1,1, 0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,254,1.00,0.01,
  0.00,0.00,0.00,0.00,0.00,0.00,snmpgetattack."

  The example records above can resemble what attackers may use as a way to disguise their malicious data. In the first record the data contains legitimate information and in the second record an attack was injected to evade detection.

The intrusion detection method first analyzes the data and converts the nonnumeric data into numerical values in the interval [0,1]. The second step is to feed the 122 features in the data to the RBM layer for pre-training using Gibbs sampling and contrastive divergence to reduce the features. The final layer uses Soft-max regression for multi-class classification of the attack classes where the classes are mutually exclusive. The gradient descent algorithm for maximum likelihood estimation of multi-class logistic regression using the soft-max function is defined below:

$$p(y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^T \mathbf{x})} \tag{8}$$

The probability that an input vector x is a member of a class i, a value of a stochastic variable Y, can be written as:

$$P(Y = i|x, W, b) = softmax_i(W_x + b) \tag{9}$$

Logistic regression is a probabilistic, linear classifier where the weight matrix W and the bias vector b are the parameters as shown in equation 9. The classification is done by projecting an input vector onto a set of hyperplanes, each of which corresponds to a class. The distance from the input to a

hyperplane reflects the probability that the input is a member of the corresponding class.

VI. Experimental Results

Our experiment shows that using logistic regression improves the accuracy rate of the classes that have low instances such as the U2R, unlike previous neural networks that incorrectly classified U2R and R2L classes.

The result in Table I shows one RBM correctly classified 92% of the attacks. The second method uses the deep belief network (DBN). Using the application UI we change the number of hidden layers and the training rate. The highest accuracy rate we obtain is 95% using two hidden layers each with 80 units. Using DBN^4+ LR, we report in Table I accuracy based on the total testing data of 311029 records. This is in contrast to previous work such as in [5] where testing was done on selected records that do not accurately represent the full dataset. Our results produce a low false negative result of 2.48% and a true positive of 97.5% as shown in Table I

In order to improve the accuracy rate we remove the duplicate records from the 10% training dataset and we train the network using 145584 records. We carried out the experiment using a Samsung laptop with 4 GB of RAM and 2.1 GHz processor running Visual Studio 2013 edition on Windows 10 operating system. The result produces the highest accuracy compared to previous methods as shown in Table III with only 10 iterations for pre-training and fine-tuning methods. The training of the deep learning method produced the highest accuracy rate of 97.9% for 17 minutes training time using four hidden layers in this form [122,72,52,40,5] as shown in Table II. The result shows the improvement in accuracy based on improving the training method and dataset. However, to accurately evaluate the network performance we tested the network using the total dataset without changes to profile an online detection.

In Figure 4 we show the ROC curve that evaluates the performance and in Figure 5 we show a graphical presentation of the confusion matrix for the results. In order to analyze the results of our experiment we count the number of the attacks in the testing dataset that are not in the training dataset. The results in Figure 5 show that the deep network correctly detects 15,246 of the new attacks which is 81.4% of the total new attacks. The confusion matrix also demonstrates the challenge for the neural network in detecting attacks with a small number of instances. However, the deep network shows improved detection for the R2L class.

1) Toward OnLine Testing: To simulate an active intrusion detection system which resembles online detection, we tested a random number of batches consisting of 1000 records each from the testing dataset continuously. The accuracy results varied between 95% and 100% and the CPU time was 0.70 seconds for each batch. We reduce the training time by reducing the number of hidden layers and simulate the network

TABLE I: Network Methods by Layers and Results

Method	Accuracy	TP	TN	FP	FN
RBM	92%	95%	92%	3.0%	6.0%
DBN^2	95%	94%	92%	2.0%	5.7%
DBN^4+LR	97.9%	97.51%	99.48%	.51%	2.48%

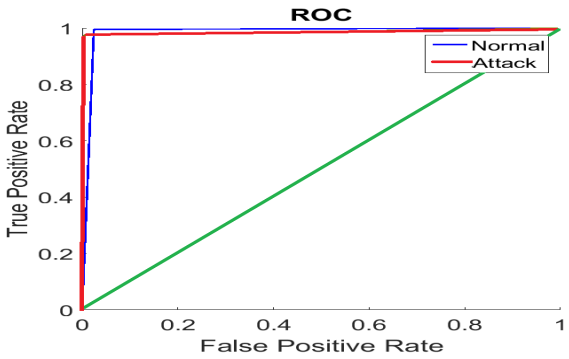

Fig. 4: ROC Simulation results for two classes

using 10 epochs and two hidden layers. The architecture has the following number of units [122,72,5]. Similarly, training the deep learning network on the training data without duplicate and noise improves the accuracy of the results to 97.908% as shown in Table IV and reduces the training time by 50% to 9 minutes with true positive rate of 97.5%. Below is a summary of the metrics we adopted to evaluate the detection method:

- Accuracy = (TP+TN)/(TP+TN+FP+FN)
- True positive = TP/(TP+TN)
- False positive = FP/(FP+TN)
- True negative = TN/(TN+FP)
- False negative = FN/(TP+FN)

VII. Conclusion and Future Work

The process of detecting anomalies using our deep learning method produced a high accuracy rate on the total 10% of the testing data of the KDDCUP'99 dataset. Despite the deficiencies in the KDDCUP'99 dataset, the dataset still represents a benchmark to demonstrate the concept of deep

TABLE II: Result Using Four Hidden Layers

Attack Class	Number of Attacks	Accurately Classified	Acuracy
Normal	60,593	60284	99.49%
Dos	229,854	229,040	99.65%
Probe	4,166	591	14.19%
R2L	16,347	14,590	89.25%
U2R	70	5	7.14%
Total	311029	304510	97.904%

TABLE III: Experimental Results and Evaluation

Method	Accuracy	TP	CPU Time (S)
This Method	97.9%	97.5%	8
Gao et al. DBN^4 [19]	93.49%	92.33%	N/A
Li et al. $AutoEncoder + DBN^{10-10}$ [5]	92.10%	92.20%	1.24

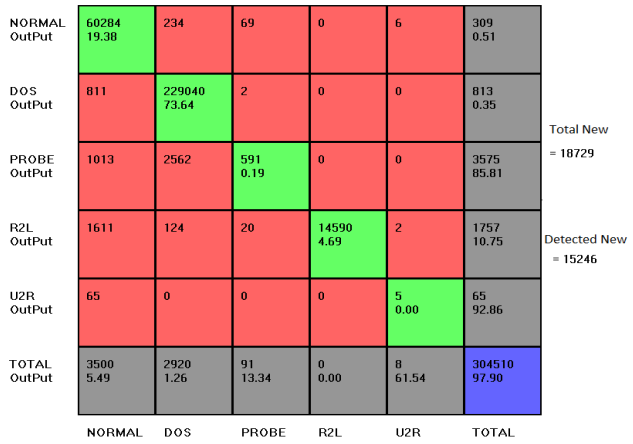

Fig. 5: Confusion matrix for five classes

learning for feature reduction and intrusion detection. The method we implement is based on a deep belief network using Logistic Regression soft-max for fine-tuning the deep network. We implemented the multi-class Logistic Regression layer and trained with 10 epochs on the improved pre-trained data to improve the overall performance of the network. We simulated a network with short training time and reduced the pre-processing of the dataset. Yet an online intrusion detection using deep learning requires further improvement in training and testing speed.

In future work we will attempt to improve the method to maximize the feature reduction process in the deep learning network and to improve the dataset. Furthermore, we plan to implement a hardware assisted detection method using the underlying unsupervised training of deep learning as examined in [20] and in [21].

TABLE IV: Result Using Two Hidden Layers

Attack Class	Number of Attacks	Accurately Classified	Acuracy
Normal	60,593	60270	99.47%
Dos	229,854	228,984	99.62%
Probe	4,166	654	15.7%
R2L	16,347	14,600	89.31%
U2R	70	16	22.89%
Total	311029	304524	97.91%

REFERENCES

[1] M. Ozsoy, "Architectural support for efficient prevention and detection of malware," Ph.D. dissertation, State University of New York at Binghamton, 2015.

[2] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[3] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[4] J. Maria, J. Amaro, G. Falcao, and L. A. Alexandre, "Stacked autoencoders using low-power accelerated architectures for object recognition in autonomous systems," *Neural Processing Letters*, pp. 1–14, 2015.

[5] Y. Li, R. Ma, and R. Jiao, "A hybrid malicious code detection method based on deep learning," *methods*, vol. 9, no. 5, 2015.

[6] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009.

[7] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation," in *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. ACM, 2011, pp. 29–36.

[8] A. Das, D. Nguyen, J. Zambreno, G. Memik, and A. Choudhary, "An FPGA-based network intrusion detection architecture," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 1, pp. 118–132, 2008.

[9] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 289–302, 2016.

[10] A. Jain, H. S. Koppula, S. Soh, B. Raghavan, A. Singh, and A. Saxena, "Brain4cars: Car that knows before you do via sensory-fusion deep learning architecture," *arXiv preprint arXiv:1601.00740*, 2016.

[11] X. Wang, N. L. Or, Z. Lu, and D. Pao, "Hardware accelerator to detect multi-segment virus patterns," *The Computer Journal*, vol. 58, no. 10, pp. 2443–2460, 2015.

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

[13] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing?" in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015, pp. 117–122.

[14] M. E. Haque and T. M. Alkharobi, "Adaptive hybrid model for network intrusion detection and comparison among machine learning algorithms," *International Journal of Machine Learning and Computing*, vol. 5, no. 1, p. 17, 2015.

[15] 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*. Springer, 2011, pp. 293–303.

[16] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer, 2012, pp. 14–36.

[17] Sugomori, "Deep learning (python, c, c++, java, scala, go)," 2015, accessed April 14, 2016. [Online]. Available: https://github.com/yusugomori/DeepLearning

[18] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International conference on artificial intelligence and statistics*, 2009, pp. 448–455.

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

[20] L. Yang, H. Yang, W. Li, and Z. Li, "Efficiently exploring FPGA design space based on semi-supervised learning," *Chinese Journal of Electronics*, vol. 25, no. 1, 2016.

[21] N. Ulltveit-Moe, H. Nergaard, L. Erdödi, T. Gjøsæter, E. Kolstad, and P. Berg, "Secure information sharing in an industrial internet of things," *arXiv preprint arXiv:1601.04301*, 2016.