A Deep Auto-Encoder based Approach for Intrusion Detection System

Fahimeh Farahnakian, Jukka Heikkonen
Department of Future Technologies, University of Turku, Turku, Finland
fahime.farhnakian@utu.fi, jukka.heikkonen@utu.fi

***Abstract*—One of the most challenging problems facing network operators today is network attacks identification due to extensive number of vulnerabilities in computer systems and creativity of attackers. To address this problem, we present a deep learning approach for intrusion detection systems. Our approach uses Deep Auto-Encoder (DAE) as one of the most well-known deep learning models. The proposed DAE model is trained in a greedy layer-wise fashion in order to avoid overfitting and local optima. The experimental results on the KDD-CUP'99 dataset show that our approach provides substantial improvement over other deep learning-based approaches in terms of accuracy, detection rate and false alarm rate.**

***Index Terms*—Intrusion detection systems, deep neural networks, stacked autoencoders, unsupervised learning, anomaly detection**

I. Introduction

In recent years, significant research has been focused on developing Intrusion Detection Systems (IDSs) to improve software and system security. Generally, IDSs can be divided into two main categories: misuse-based IDSs and anomaly-based IDSs. Misuse-based IDSs detect known attacks based on the predetermined signature. Therefore, dynamic signature updating is so important and new attack definitions are frequently released by IDS vendors. However, the misuse-based IDS cannot incorporate the rapidly growing number of vulnerabilities and exploits. Anomaly-based IDSs are designed to capture any deviation from profiles of normal behavior. Therefore, they are more suitable than misuse-based detection systems for detecting unknown or novel attacks without any prior knowledge [1].

In this paper, we present an anomaly-based IDS in order to identify intrusions using deep learning. Deep learning algorithms have gained interest recently as they can efficiently learn a model in order to classify unknown samples. The most popular techniques include deep belief networks with restricted boltzmann machine [2], convolutional neural network [3], long short term memory recurrent neural network [4] and stacked auto-encoders [5].

Inspired by the success of auto-encoder based approaches in number of challenging classification problems [6], our IDS employs a stacked (deep) auto-encoders model for intrusion detection. Dimensionality reduction is another reason that we motivated to use autoencoder. The other advantage of using autoencoder is the capability of the minority and the majority classes to address the imbalanced classification problems. Deep Auto-Encoders (DAEs) [3] are created by daisy chaining auto-encoders together. Our Deep Auto-Encoder based Intrusion Detection System (DAE-IDS) consists of four auto-encoders where the output of each auto-encoder in the current layer is used as the input of the auto-encoder in the next layer. In addition, training an auto-encoder is started when training the pervious one is completed. In the last hidden layer, a softmax classifier classifies the attack classes from the input dataset. Therefore, DAE-IDS performs unsupervised feature learning, supervised fine-tuning, and thus intrusion detection. Since the performance of a deep model depends on hyper-parameters that are used for model initiation, we conduct a series of preliminary experiments to find the optimal hyper-parameters. Moreover, we investigate the effect of both number of hidden layers size and hidden neurons on the DAE-IDS performance.

We had performed various experiments on KDD-CUP'99 dataset [7] which is the mostly widely used standard dataset for the evaluation of IDSs. Experimental results show that DAE-IDS can produce low false negative rate (0.42%), high accuracy (94.71%) and high detection rate (94.53%). We also compared DAE-IDS with other existing deep learning based methods for intrusion detection. The results show that the performance of DAE-IDS is better than the previous methods.

The remainder of the paper is organized as follows. Section II discusses some of the most important related works. We briefly review auto-encoder model in Section III. Section IV presents the DAE-based approach for anomaly detection. Section V shows the implementation issue of our approach and describes pre-processing tasks on the dataset. Finally, we give the experimental results and compare our approach with other existing deep-learning based methods in Section VI. Finally, we present our conclusions in Section VII.

II. Related Work

IDSs have been broadly researched as defensive techniques to identify unknown or zero-day attacks. Anomaly-based IDSs model the normal behavior of network and then identify attacks as deviations from the normal behavior. The main challenge in designing of anomaly-based IDS is the potential for high false alarm rates as previously unseen system behaviors may be categorized as anomalies [8]. Therefore, an efficient IDS is able to handel a large amount of data with changing patterns in real time situations [9].

Recently, deep learning methods have drawn a lot of industrial and academic interest [3]. Deep learning is a branch of machine learning that uses Artificial Neural Network (ANN) as an architecture. Traditional ANNs typically contain one to three of hidden layers, whereas a Deep Neural Network (DNN) can have tens or even hundreds of hidden layers to support higher generalization capability in comparison to ANN. DNN models have used in the development of IDS as they capable to exploit the unknown structure in the input distribution in order to discover good representations. Yuancheng et al. [10] presented a hybrid deep learning technique for intrusion detection. They first used an autoencoder in order to reduce the dimensionality of data and extract the main features of data. Then, a Deep Belief Network (DBN) is used to train their intrusion detection systems using the KDD-CUP'99 dataset. DBN is composed of multi-layer Restricted Boltzmann Machines (RBM) which each RBM consists of the visible units and hidden units. Fiore et al [11] also employed a RBM to detect anomalies by training a model with real workload traces from 42 hours work station traffic and they test the accuracy of RBM with KDD-CUP'99 dataset. Their model achives about 85% accuracy on the total 10% KDDCUP99 dataset. In a similar work, Gao et al. [12] proposed a DBN model for intrusion detection systems. This model used an unsupervised greedy learning algorithm to learn a similarity representation over the nonlinear and high-dimensional data. It is evaluated on the KDD-CUP'99 dataset and the results show that four-hidden-layer RBM can produce the higher accuracy in comparison with SVM and ANN. Jihyun et al. [4] proposed a Long Short Term Memory (LSTM) architecture to a Recurrent Neural Network (RNN) and train the IDS model using KDD Cup 1999 dataset. They found more accuracy and detection rate in comparison with other classified such as KNN and SVM. Potluri et al. [13] introduced an accelerated DNN architecture to identify the abnormalities in the network data. They evaluated the performance of the DNN training related to different processor types and numbers of cores. The acceleration of the training process using the multicore CPUs was faster than the serial training mechanism. In this paper, we proposed a deep auto-encoder based approach for improving intrusion detection system performance. Our main contributions are as follows:

1) We employed a DAE model to discover important feature representations from the imbalanced training data and generate a model to detect normal and abnormal behaviors.

2) Our model is pre-trained using an unsupervised learning algorithm to avoid overfitting and local optima. A softmax classifier is added on the top of the model to represent the desired outputs (normal or type of attack).

3) The performance of the proposed DAE is evaluated by KDD-CUP'99 dataset. KDD-CUP'99 dataset is a common benchmark for network intrusion detection consisting of real network data. Moreover, a series of preliminary experiments is conducted to explore the performance of DAE based on different number of hidden layers and units.

III. Background

An auto-encoder includes two parts: encoder and decoder. Encoder aims to compress input data into a low-dimensional representation, and decoder reconstruct input data based on the low-dimension representation generated by the encoder. On the other hand, auto-encoder can encode a representation of an input layer into a hidden layer and then decode it into an output layer [3].

For a given training dataset $X = \{x_1, x_2, ..., x_m\}$ with m samples, where x_i is a d-dimensional feature vector, the encoder maps the input vector x_i to a hidden representation vector h_i through a deterministic mapping f_θ as given in (1)

$$h_i = f_\theta(x_i) = s(Wx_i + b), \tag{1}$$

where W is a $\acute{d} \times d$, $\acute{d}$ is the number of hidden units, b is a bias vector, θ is the mapping parameter set $\theta = \{W, b\}$. s is sigmoid activation function denoted as

$$s(t) = \frac{1}{1 + exp^{-t}}, \tag{2}$$

where parameter t affects for the shape of the function.

The decoder maps back the resulting hidden representation h_i to a reconstructed d-dimensional vector y_i in input space as

$$y_i = g_{\acute{\theta}}(x_i) = s(\acute{W}h_i + \acute{b}), \tag{3}$$

where $\acute{W}$ is a $d \times \acute{d}$, $\acute{b}$ is a bias vector and $\acute{\theta} = \{\acute{W}, \acute{b}\}$.

The goal of training the autoencoder is to minimize the difference between input and output. Therefore, a loss function is calculated by the following equation

$$L(x, y) = \frac{1}{m}\sum_{i=1}^{m} \left\| x_i - y_i \right\|^2, \tag{4}$$

where m is the total number of training dataset.

The main objective is to find the optimal parameters (θ and $\acute{\theta}$) which can be effectively minimize the difference between input and reconstructed output over the whole training set as

$$\theta = \{W, b\} = arg_\theta minL(x, y). \tag{5}$$

IV. Deep Auto-Encoder based Intrusion Detection System (DAE-IDS)

In this section, we describe how the intrusion detection problem is addressed through our deep learning based approach. The intrusion detection problem statement can be stated as follows: given a dataset D with m samples defined below, assign a label (i.e., normal or attack) to each unlabeled sample based on its feature vector.

The dataset D is denoted to be of the form $\langle F_i, C_i \rangle$, where F_i is the feature vectors of sample x_i ($i = \{1, 2, ..., m\}$) and C_i is the class label of sample i (where $C_i \in \{normal, attack\}$).

In order to solve the intrusion detection problem, Deep Auto-Encoder based Intrusion Detection System (DAE-IDS) runs in two phases: training and testing. In the training phase, the system uses a training dataset and creates a model based on the proposed Deep Auto-Encoder (DAE) model. Then, the system employs the model for identifying the label of unseen data (test dataset) in the testing phase to estimate the performance of the system if is used on-line.

The proposed deep auto-encoder architecture for creating the model in the training phase is shown in Fig. 1. DAE consists of three type of layers: input, hidden and output layers. The input layer takes the input from the training dataset. The input layer of our DAE model represents all the 117 features of the KDD-CUP'99 dataset. The proposed DAE model is built by daisy chaining auto-encoder together. The hidden layer 1 is presented the first auto-encoder which selects 32 features of 117 features from the input data. The output of hidden layer h is used as a input of hidden layer $h+1$. Based on the layer-wise algorithm [14], an auto-encoder at layer $h+1$ is trained after the completion of training an auto-encoder at layer h. The last hidden layer is a supervised layer which classifies attack classes by using the softmax classifier. Finally, the output layer is used as an output of the entire DAE model and it represents five available classes of the KDD-CUP'99 dataset. There are lot of choices to select activation functions in the hidden layers such as linear, softmax, sigmoid, tanh and rectified linear functions. The DAE model can extract more useful features when it utilizes a non-linear activation function. Therefore, we choose the sigmoid function for hidden layers in this paper based on a series of preliminary experiments.

To train the proposed DAE model, we employed a greedy layer-wise unsupervised learning algorithm that is developed by Hinton et al. [15]. This algorithm can improve the poor performance of a gradient-based training algorithm for training a deep model. The greedy layer-wise unsupervised learning algorithm trains a deep network layer by layer in a bottom-up manner (pre-training) and then uses back propagation in a top-down manner to fine-tune parameters of all layers in DAE. Therefore, it produces a greedy layer-wise training fashion in order to avoid many of training problems of the deep model in a supervised manner. The pseudocode of the training DAE-IDS is given as Algorithm 1. Generally, the training procedure consists of three main steps:

(1) Pre-training: each hidden layer is trained as an auto-encoder to minimize the reconstruction error. At first, the first auto-encoder uses the training dataset without labels as inputs (unsupervised) and creates a compressed representation of the inputs. Then, the second auto-encoder is trained by using the first auto-encoder s' output as inputs. This task sequence is iterated for all auto-encoders in DAE-IDS. Finally, a set of robust feature is built at the end of this step.

(2) Fine-tuning: after pre-training step, a supervised layer/model takes the output of the last auto-encoder as inputs and then is trained using labels of the training dataset (supervised).

(3) Full fine-tuning: after fine-tuning step, all layers are further fine-tuned through backpropagation in a supervised way.

The pseudocode in Algorithm 1 expects as input a training dataset $X = \{x_1, x_2, ..., x_m\}$ with m samples and the DAE architecture with L hidden layers (line 1). In the pre-training step (line 2–12), each hidden layer of DAE is trained in a supervised fashion. Finally, this step generate a set of parameters of all layers $\theta = \{\theta_1, \theta_2, ..., \theta_L\}$ where $\theta_l = \{W_l, b_l\}$, $l \in \{1, 2, ..., L\}$. For each layer l (line 2), the parameters are initialized to zero. The algorithm iterates when its stopping criteria is not reached (line 6). In each iteration, the hidden representation vector of layer l, h_l, is computed based on the previous layer $l-1$ (line 7). Then, the algorithm computes l-th hidden layer output y_l (line 8). After that, the loss function is computed by using (4) (line 9). Finally, the parameters (θ and $\acute{\theta}$) are updated based on the loss function (line 10). After the pre-training step, a softmax classifier is employed at the supervised layer to determine the class of sample (line 13–14). In the last step (fine-tuning), DAE-IDS is further fine-tuned by performing back propagation in a supervised mechanism to tune the parameter set of all layers (line 15).

Algorithm 1 Training algorithm

```
1:  Input: Dataset $X = \{x_1, x_2, ...x_m\}$ with $m$ samples,
    number of hidden layers $L$
2:  for $l \in [1, L]$ do
3:      initialize $W_l = 0$, $\acute{W_l} = 0$, $b_l = 0$, $\acute{b_l} = 0$
4:      define the $l$-th hidden layer representation vector $h_l = s(W_l h_{l-1} + b_l)$
5:      define the $l$-th hidden layer output $y_l = s(\acute{W_l} h_l + \acute{b_l})$
6:      while not stopping criterion do
7:          compute $h_l$ from $h_{l-1}$
8:          compute $y_l$
9:          compute the loss function
10:         update layer parameters $\theta_l = (W_l, b_l)$ and $\acute{\theta_l} = \{\acute{W_l}, \acute{b_l}\}$.
11:     end while
12: end for
13: initialize $(W_{l+1}, b_{l+1})$ at the supervised layer
14: calculate the labels for each sample $x_i$ of the training
    dataset $X$
15: perform BP in a supervised way to tune parameter of all
    layers;
```

V. Performance Evaluation

A. Dataset

We evaluated our proposed approach on the KDD-CUP'99 dataset [7] which is mostly widely used for the evaluation the intrusion detection system. This dataset is built based on the data captured in DARPA98 IDS evaluation program [16] and provides four gigabytes of compressed raw tcpdump data of seven weeks of network traffic. We used 10% of the original KDD-CUP'99 dataset contains 494,021 samples for training the model in the training phase. It is common practice to use 10% of the original data as a training dataset since this dataset can represent the original KDD-CUP'99 data and allow for reduced computation [10]. In the testing phase, the trained model is tested by using a test dataset contains 311029 samples with corrected labels. Each sample of

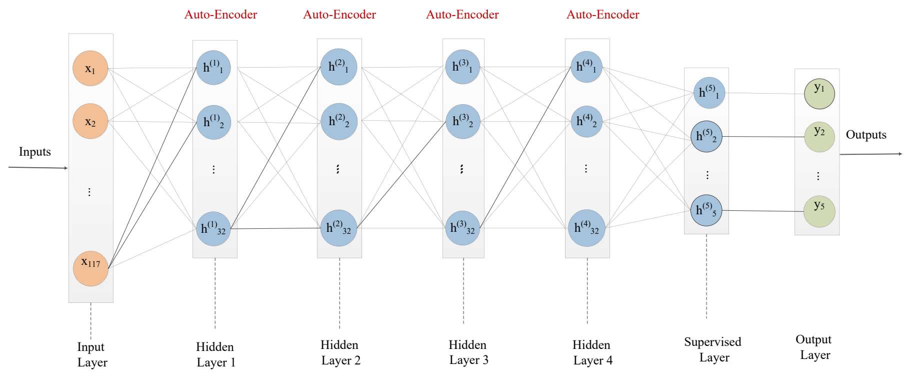


Fig. 1. The proposed deep auto-encoder architecture

TABLE II
OVERVIEW ON TRAINING AND TEST DATASETS

Dataset	Normal	DoS	U2R	R2L	Probe	Total
Training	97,278	391,458	52	1,126	4,107	494,021
Test	60,593	229,854	70	16,347	4,166	311,029

the training and test datasets has 41 features. Those features consist of 38 continuous or discrete numerical features and 3 categorical features. Moreover, each sample is labeled as normal or a particular kind of attack. Our experimental results are collected based on two scenarios. In the first scenario, a sample is categorized into two main classes: normal and attack. Therefore, the intrusion detection problem is assumed as a binary classification problem since the sample belongs to normal class or attack class. In the second scenario, we define a multi-classification problem to classify a sample into a normal or one of the four attack classes. Four attack classes fall in one of the following categories:

Denial of Service (DoS): is an attack in which an attacker attempts to prevent legitimate users access to a machine or make memory or some computing resources too full/busy for handling legitimate requests.

User to Root (U2R): is an attack in which an attacker access to the system by a normal user account and then exploits some vulnerability to gain root access to the system.

Remote to Local (R2L): is an attack in which an attacker sends packets to a machine on the network without any accounts on that machine and is able to exploits some vulnerability to gain local access as a user of that machine.

Probing (Probe): is an attack in which an attacker collects information about a network of computers for the apparent purpose of circumventing its security controls.

There are 40 different attacks in training and test dataset that are totaly categorized into four attack classes in multi-classification problem. Table I shows which attack types belong to each class. In addition, the number of samples in normal and four attack classes for both training and test datasets are shown in Table II.

B. Data Pre-processing

We performed the following pre-processing procedures on the KDD-CUP'99 training and test datasets:

(1) Feature Numeralization: the symbolic features (protocol type, services and flag) are mapped to numerical features by binary coding. For example tcp, udp and icmp protocols are mapped to (1,0,0), (0,1,0) and (0,0,1), respectively. Similarity, the 'flag' feature with 11 values and 'services' feature with 65 values can be mapped to numerical features. Therefore, 41 original features are finally numeralized to 117 features.

(2) Class Numeralization: the non-numerical attack types are converted into the numeric categorizes. In the binary classification, 1 and 0 are assigned to the normal and attack class by using binary coding, respectively. In the multi-classification, we used one hot encoding to convert five categorical classes into five binary classes, with only one active.

(3) Feature Normalization: the numeric features must be normalized for removing the effect of original feature value scales. Each feature is normalized as

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}, \quad (6)$$

where the training dataset $X = \{x_1, x_2, ..., x_m\}$ with m samples, where x_i is a d-dimensional feature vector and z_i is the ith normalized data. Therefor all numeric features values are ranged between 0 and 1.

(4) Redundancies reduction: one of the main problems of the KDD-CUP'99 data is the large number of duplicate records that leads to the bias towards more frequent records [17]. To solve this problem, we removed all duplicate records in data, and kept only one copy of each record. After redundancies reduction, the training and test datasets consist of 145,586 and 77,291 instances, respectively.

C. Evaluation Metrics

The main aim of the evaluation to show the implications of enriching the IDS with the deep auto-encoder model on: i) maximize Detection Rate (DR); ii) maximize Accuracy (AC); and iii) minimize False Alarm rate (FA). The performance

TABLE I
CATEGORY OF THE ATTACKS

Category	Attacks
DoS	back, land, neptune, pod, smurf, teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm
U2R	buffer-overflow, loadmodule, perl,rootkit, Sqlattack, Xterm, Ps
R2L	ftp-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named
Probe	ipsweep, nmap, portsweep, satan, Portsweep, Mscan, Saint

of DAE-IDS is assessed through the following metrics which have been widely used in intrusion detection system [10], [12].

$$DR = \frac{TP}{(TP + FN)}, \tag{7}$$

$$FA = \frac{FP}{(FP + TN)}, \tag{8}$$

$$AC = \frac{TP + TN}{(TN + TP + FN + FP)}, \tag{9}$$

where True Positive (TP) is the number of attacks classified rightly as attack; True Negative (TN) is the number of normal events rightly classified normal. False Positive (FP) is the number of normal events misclassified as attacks and False Negative (FN) is the number of attacks misclassified as normal.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

A. *Model hyper-parameters*

Designing an efficient deep model involves a challenging problem called hyper-parameters optimization. Optimization of our deep model was performed over key hyper-parameters and their values are given in Table III. In order to tune the hyperparameters for all models in this paper, we used 131,586 instances for training and the remaining 14,000 instances for validation. After the best value of hyperparameters are selected, the final model is trained with all 145,586 instances. Moreover, the model achieves the best result of accuracy when batch size and epochs of pre-training are 100 and 150, respectively. Moreover, the model got highest accuracy with 100 batch size and 100 epochs of fine-tuning.

TABLE III
THE TESTED VALUES OF HYPER-PARAMETERS

Hyper-parameter name	Values	Best Values
Pre-training batch size	10, 20, 40, 60, 80, 100	100
Pre-tuning epochs	10, 50, 100,150	150
Fine-tuning batch size	10, 20, 40, 60, 80, 100	100
Fine-tuning epochs	10, 50, 100	100

B. *Different numbers of hidden layers and hidden units*

In the other experiments, we investigate an impact the number of hidden layers and hidden units on the performance of DAE-IDS. Table IV shows test classification accuracy of DAE-IDS with different numbers of hidden layer and units. The DAE-IDS with four hidden layers and 32 hidden units at each layer is superior to other deep networks in testing accuracy for intrusion detection.

TABLE IV
THE PERFORMANCE OF THE DAE-IDS WITH DIFFERENT NUMBER OF HIDDEN LAYERS AND HIDDEN UNITS

Number of hidden layers	Neurons	Testing AC(%)
1	(32)	93.79
	(64)	93.05
	(100)	93.25
2	(32,32)	93.79
	(64,64)	93.25
	(100, 100)	93.34
3	(32,32,32)	93.25
	(64,64,64)	93.71
	(100,100,100)	93.70
4	**(32,32,32,32)**	**94.71**
	(64,64,64,64)	93.50
	(100,100,100,100)	93.81
5	(32,32,32,32,32)	93.27
	(64,64,64,64,64)	93.11
	(100,100,100,100,100)	93.93

C. *Binary and multi classification*

The results are collected based on two defined scenarios in the pervious section. Table V shows the detection rate, false alarm rate and accuracy for binary-classification and multi-classification. The results show that DAE-IDS obtain 96.53% and 94.71% accuracy in binary-classification and multi-classification, respectively. Moreover, DAE-IDS produces a low false alarm (FA) of 0.35% in binary-classification.

D. *Comparison benchmarks*

In order to investigate the effectiveness of our DAE model on intrusion detection performance, we compared our approach to the previous deep neural network methods in [10] and [12]. Table VI shows that the performance of DAE-IDS is better than other benchmark algorithms in terms of accuracy

TABLE V
THE DETECTION RATE, FALSE ALARM AND ACCURACY BY DAE-IDS FOR TWO PROPOSED SCENARIOS

Scenario	DR(%)	FA(%)	AC(%)
Binary-classification	95.65	0.35	96.53
Multi-classification	94.53	0.42	94.71

and TP. This is because, DAE can learn a set of features with better classification capability of the minority and majority classes to address the imbalanced data. Another reason for achieving the better accuracy by DAE is unsupervised pre-training task. Unsupervised pre-training gives substantially higher test classification accuracy than no pre-training

TABLE VI
THE TP AND ACCURACY BY DAE-IDS AND BENCHMARK APPROACHES

Method	TP(%)	AC(%)
DAE-IDS	94.42	94.71
DBN^4 [12]	92.33	93.49
AutoEncoder+DBN^{10-10} **[10]**	92.20	92.10

VII. Conclusion and Future Work

Designing efficient Intrusion Detection Systems (IDSs) have gained a lot of attractions due to huge increase different kinds of attacks and network traffic. In this paper, we presented a deep auto-encoder approach for improving the intrusion detection system. The auto-encoder is one of the most interesting models to extract features from the high-dimensional data in the context of deep learning. Our proposed Deep Auto-Encoder based Intrusion Detection System (DAE-IDS) consists of four auto-encoders which the output of the auto-encoder at the current layer is used as the input of the auto-encoder in the next layer. In addition, an auto-encoder at the current layer is trained before the auto-encoder at the next layer. To train DAE-IDS, we utilized a greedy unsupervised layer-wise training mechanism that helps to improve the deep model performance. After training four auto-encoders, we used a softmax layer to classify the inputs into the normal and attack. We used the KDD-CUP'99 dataset to evaluate the performance of DAE-IDS as this dataset has been utilized extensively for evaluating IDSs. The proposed approach achieved detection accuracy 94.71% on the total 10% KDDCUP99 test dataset.

We will further explore how sparsity constraints are imposed on auto-encoder and how sparse deep auto-encoders can be designed to further improve intrusion detection effectiveness. Meanwhile, it would be interesting to investigate other deep learning and classification methods for intrusion detection.

References

[1] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," 2002.

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

[3] Y. Bengio, "Learning deep architectures for ai," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1561/2200000006

[4] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 International Conference on Platform Technology and Service (PlatCon)*, Feb 2016, pp. 1–5.

[5] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1953039

[6] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, ser. Proceedings of Machine Learning Research, I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, Eds., vol. 27. Bellevue, Washington, USA: PMLR, 02 Jul 2012, pp. 37–49.

[7] "Nsl-kdd data set for network-based intrusion detection systems," March 2009. [Online]. Available: Available on: http://nsl.cs.unb.ca/NSL-KDD/

[8] F. Hosseinpour, P. Vahdani Amoli, F. Farahnakian, J. Plosila, and T. Hmlinen, "Artificial immune system based intrusion detection: Innate immunity using an unsupervised learning approach," *International Journal of Digital Content Technology and Its Applications*, vol. 8, no. 5, p. 112, 2014.

[9] E. Hodo, X. J. A. Bellekens, A. Hamilton, C. Tachtatzis, and R. C. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," *CoRR*, vol. abs/1701.02145, 2017. [Online]. Available: http://arxiv.org/abs/1701.02145

[10] L. Yuancheng, M. Rong, and J. Ruhai, "a hybrid malicious code detection method based on deep learning," *Journal of Security and Its Applications*, vol. 9, no. 5, pp. 205–216, 2015.

[11] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted boltzmann machine," *Neurocomput.*, vol. 122, pp. 13–23, Dec. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2012.11.050

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

[13] 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)*, Sept 2016, pp. 1–8.

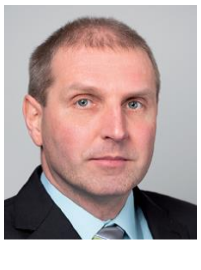
[14] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, pp. 153–160. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976456.2976476

[15] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1162/neco.2006.18.7.1527

[16] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000. [Online]. Available: http://doi.acm.org/10.1145/382912.382923

[17] 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 International Conference on Computational Intelligence for Security and Defense Applications*, ser. CISDA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 53–58. [Online]. Available: http://dl.acm.org/citation.cfm?id=1736481.1736489

Fahimeh Farahnakian received her PhD degree from the University of Turku, Finland in 2016. Currently she is working as a postdoctoral researcher at the University of Turku, Finland. Her research interests include machine learning, neural networks, deep learning, big data, autonomous system and cloud computing. She is a member of IEEE and is a frequent reviewer for research journals.



Jukka Heikkonen has been a professor of computer science of University of Turku since 2009. His research is related to machine learning and probabilistic and information theoretical modelling applied in wide varying application domains. He has worked at top level research laboratories and Center of Excellences in Finland and international organizations (EU, Japan) and has led many international and national research projects.