Anomaly-based Intrusion Detection Using Deep Neural Networks

[1] Fahimeh Farahnakian, [2] Jukka Heikkonen
[1,2] *Department of Future Technologies, University of Turku, Turku, Finland*

Abstract

Identification of network attacks is a matter of great concern for network operators due to extensive the number of vulnerabilities in computer systems and creativity of the attackers. Anomaly-based Intrusion Detection Systems (IDSs) present a significant opportunity to identify possible incidents, logging information and reporting attempts. However, these systems generate a low detection accuracy rate with changing network environment or services. To overcome this problem, we present a deep neural network architecture based on a combination of a stacked denoising autoencoder and a softmax classifier. Our architecture can extract important features from data and learn a model for detecting abnormal behaviors. The model is trained locally to denoise corrupted versions of their inputs based on stacking layers of denoising autoencoders in order to achieve reliable intrusion detection. Experimental results on real KDD-CUP'99 dataset show that our architecture outperformed shallow learning architectures and other deep neural network architectures.

Keywords: *Intrusion detection systems, deep neural networks, stacked denoising autoencoders, unsupervised learning*

1. Introduction

In the past few years, there have been several attempts to tackle security problems by designing efficient Intrusion Detection Systems (IDSs) [1] [2]. An IDS is a security tool that monitors the network traffic and identifies malicious actions. Generally, IDSs can be categorized into four types: signature-based, anomaly-based, specification-based and hybrid [1]. Signature-based IDSs use pre-determined signatures or patterns to detect intrusions in a network. They generate an alarm if a system or network activity does not match with the signatures. Effective at detecting known attacks and easy understanding of their mechanism is the advantage of signature-based IDSs. However, they are not able to identify new attacks variants of known attacks. Anomaly-based IDSs are more suitable than signature-based detection systems for detecting unknown or novel attacks. They can capture any deviations from profiles of normal behavior. For this reason, the most of the existing IDSs in the Internet of Things (IoT) and general purpose computing are built based on the anomaly-based detection. However, learning the entire scope of normal behavior is not a simple task. The anomaly-based IDSs identify an activity as an intrusion if it does not match a normal behavior and therefore generate high false alarms. Specification-based IDSs use a set of rules and thresholds in order to define the normal behavior of network components such as protocols, nodes and routing tables. Therefore, they detect an intrusion when the network behavior deviates from specification definitions. Specification-based IDSs have a similar idea of anomaly-based IDS as both of them identify deviations from normal behavior. However, there is a human expert in specification-based IDSs to manually determine the rules of each specification. Manually defining the specifications usually provide fewer false alarms in these systems compared with the anomaly-based IDSs [2]. Hybrid IDSs use concepts of three former IDSs to enhance their advantages and reduce their respective limitation. For example, Raza et al. [3] proposed a hybrid IDS for IoT in order to provide a solution that minimizes the storage cost of signature-based IDS and computing cost of anomaly-based IDS.

The other classification is made with regard to the location where IDSs is deployed, e.g. network-based IDSs, host-based IDSs, and hypervisor-based IDSs. Network-based IDSs are placed in network points such as routers or gateway for analyzing the network traffic and detecting intrusions. Host-based IDSs are deployed between a router and Cloud host for monitoring physical or virtual hosts to find abnormal activities such as deletion or modification of system files or unwanted configuration changes in Cloud components. Hypervisor-based IDSs are placed at the hypervisors or Virtual Machine Monitors (VMMs) for checking the information of all VMMs and communication between VMMs to detect suspicious behaviors.

In this paper, we present an anomaly-based IDS for identifying network traffic intrusions. Designing an accurate and efficient anomaly-based IDS is a challenging task as they have low detection accuracy rate when the network environments or services are changing over time. To address this challenge, our IDS uses a popular Machine Learning (ML) technique called deep learning. ML methods have gained interest recently to construct anomaly-based IDSs. These widely-used techniques are k-nearest neighbor [4][5], decision tree [4], support vector machine [5], artificial immune system [6], genetic algorithm [7] and artificial neural network [8][9]. All of these techniques learn a model by using a training data and then use the learned model to classify unseen data. Deep learning has been chosen for its ability to discover a massive amount of

structures by extracting inherent features from the data. The most popular deep learning models include deep belief networks with restricted Boltzmann machine [10], convolutional neural network [11], long short-term memory recurrent neural network[12] and stacked autoencoders [13]. Inspired by the success of applying stacked autoencoder in many numbers of challenging classification problems [10][11][12][13][13], our anomaly-based IDS uses Stacked AutoEncoders (SAEs) [13]. SAEs are built by multi-layer autoencoder where the output of each auto-encoder in the current layer is used as the input of the autoencoder in the next layer. Dimensionality reduction is another reason that we motivated to use autoencoder. In order to improve the generalization ability of a learned model by SAEs, our proposed IDS uses a specific kind of SAEs that is named Stacked Denoising AutoEncoders (SDAEs). SDAEs make the learned model robust to the partial corruption of the input data [13].

Our proposed SDAEs is firstly pre-trained layer by layer based on an unsupervised fashion. The unsupervised pre-training fashion can help the model to avoid local optima and overfitting problem. Moreover, important features can be extracted from unlabeled data at different layers after pre-training. Then, a softmax classifier is added on the top of SDAEs for representing the desired outputs. Finally, our network is fine-tuned based on a supervised manner using labeled data.

We had performed various experiments on the KDD-CUP'99 dataset [15] which is the most widely used as a standard dataset for the evaluation of IDSs. The performance of deep networks is strongly dependent on the network topology. Therefore, we investigate how the performance of SDAEs changes based on the breadth (number of hidden neurons per layer) and depth (number of hidden layers) of the network. The obtained results show that our SDAEs can achieve better detection accuracy when it consists of four hidden layers and 64 hidden neurons per layer. We also explore the capabilities of different deep networks without and with unsupervised pre-training. Experimental results show that the performance of SDAEs is better than the previous methods when it is pre-trained based on an unsupervised fashion. We also compared SDAEs with other deep architectures and shallow learning architectures for intrusion detection. The results show that SDAEs can produce a high detection rate (96.85%) on new data, a separate test dataset.

To the best of our knowledge, currently there are no existing works on using the SDAE for anomaly-based IDS. Our main contributions are as follows:

- We employ SDAEs to discover important feature representations from the training data and generate a model to detect normal and abnormal behaviors.
- In order to improve the generalization ability of the model, training data is corrupted by noise. SDAEs can extract more robust features from the hidden layer by training autoencoders to construct the input data from a corrupted version of it.
- 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).
- The performance of the proposed SDAEs model is evaluated by the KDD-CUP'99 dataset. The KDD-CUP'99 dataset is a common benchmark for network intrusion detection consisting of real network data. The results show that SDAEs outperform other deep architectures and sallow learning architecture in terms of detection accuracy. The detection accuracy is estimated by using a separate test dataset.
- A series of experiments are conducted to explore the performance of SDAEs based on the different number of hidden layers and units. We also evaluate the effect of different training strategies and corruption levels on the SDAEs performance.

The remainder of the paper is organized as follows. Section 2 discusses some of the most important related works. We briefly review the sallow learning architectures and deep architectures in Section 3. Section 4 presents our proposed deep neural network architecture for intrusion detection. Section 5 shows the implementation issue of our approach, pre-processing tasks on the dataset and experimental results. Finally, we present our conclusions in Section 6.

2. Related works

In recent years, there has been major significant research in designing Intrusion Detection Systems (IDSs) to improve network security. Anomaly-based IDSs is a defensive solution to model normal behaviors of network traffic and then identify attacks or intrusions as deviations from the normal behaviors. The main problem in designing of anomaly-based IDSs is that they generate low detection accuracy when the network environments or services change [6]. Another problem is most of the existing anomaly-based IDSs are not able to detect unknown attack types.

To address these problems, significant research has been focused on improving the performance of IDs by using ML techniques. These techniques generally divided into supervised and unsupervised learning methods. The supervised ML methods use a corrected labeled training dataset to learn a model and then employ the model to classify the unknown data. Most anomaly-based IDSs use the supervised learning algorithms in order to detect intrusions. Recent research has shown

that Artificial Neural Network (ANN) perform efficiently for intrusion detection. Jiang et al. [16] applied Radial Basis function (RBF) network for both signature and anomaly detection. Because of the short training time and high accuracy of the RBF neural networks, their proposed IDS can monitor network traffic in a real-time environment. Moreover, Hodo et al. [9] presented a Multilayer Perceptron (MLP), a supervised ANN, to collect the information from different parts of IoT network and then detect a Denial of Service (DoS) attack on the network. However, labeled data can be extremely difficult or impossible to obtain in real applications. In addition, a set of labeled data may be not have been covered all possible attacks on a real application.

The performance of supervised learning algorithms is significantly reduced if unknown attacks are present in the test data. Unsupervised learning algorithms can overcome some of the limitations of supervised learning-based IDS as they can find information from unlabeled data. They detect attacks by determining abnormal behaviors that statistically deviate from normal behaviors. Cannady [8] developed an ANN-based approach to classify and detect misuse in the network. He mentioned that one of the advantages of using ANN for detecting misuse is flexibility for analyzing the data from the network, even if the data is not completed or distorted.

Deep Neural Networks (DNNs) are a class of ML techniques where classification is conducted by training data through a hierarchical ANNs. They were recently used in the development of IDS as they capable exploiting unknown structures in training data for discovering good representations. Farahnakian et al. [17] proposed an approach based on deep autoencoder to model the normal and abnormal behaviors in the KDD-CUP'99 dataset. Yuancheng et al. [18] 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, they utilized a Deep Belief Network (DBN) in order to train their systems using the KDD-CUP'99 dataset. DBN is composed of multi-layer Restricted Boltzmann Machines (RBM) where each RBM consists of the visible units and hidden units. Fiore et al. [18] also employed an RBM to detect intrusions by training a model with real workload traces from 42hour workstation traffic and they test the accuracy of RBM with the KDD-CUP'99 dataset.

Training deep networks is a challenging task as the standard learning strategy consisting of random weights initialization and applying the gradient descent method empirically finds poor solutions for a network with more than two hidden layers. In order to overcome this challenge, Hinton et al. [20] introduced a greedy-layer-wise algorithm for training a DBN model with many hidden layers. This algorithm can be extended to the stacked autoencoder networks as presented in [21]. Gao et al. [22] trained a DBN model for intrusion detection systems based on an unsupervised greedy learning algorithm. Their model learns a similarity representation over the nonlinear and high-dimensional data. It was 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. Alom et al. [10] also used a DBN to detect the unknown attacks on the NSL-KDD dataset. In another work [12], a Long Short-Term Memory (LSTM) architecture is proposed to a Recurrent Neural Network (RNN) and trained the IDS model using KDD-CUP'99 dataset. They found more accuracy and detection rate in comparison with other classification methods such as K-nearest neighbor and support vector machine. Potluri et al. [23] introduced an accelerated DNN architecture to identify the abnormalities in the network data. They used NSL-KDD dataset for training the proposed DNN model.

3. Background

This section briefly reviews all classification algorithms, both shallow and deep learning, which were used in this work for their performance comparison.

3.1. K-nearest neighbor

K-Nearest Neighbor (K-NN) is a nonparametric technique for classification. The class of an unlabeled instance is determined by k nearest neighbors of that instance. As the performance of the K-NN algorithm is strongly dependent on the parameter *k*, finding the best values of k is essential [24]**.** A large *k* value decreases the effect of noise and minimizes the prediction losses. However, a small *k* value allows simple implementation and efficient queries. Cross-validation is a common method in order to estimate the accuracy and validity of the classifier with different *k* values.

3.2. Decision tree

A Decision Tree (DT) is a tree structure which classifies an instance through a sequence of decisions. The classification of an instance proceeds from the root node to a suitable end leaf node, where each end leaf node represents a classification category. The features of the instances are assigned to each node, and the value of each branch is corresponding to the features. The well-known algorithms for creating a DT are C4.5 and ID3 [25]**.**

3.3. Support vector machine

Support vector machines (SVM) is a supervised learning algorithm for classifying data. The goal of SVM is to find the optimal decision boundary, i.e. the separating hyperplane, in the higher dimensional feature space. Binary SVM classification categorizes two classes in such a way that the distance between the hyperplane and the nearest data examples of each class is maximized. As a decision boundary is determined by support vectors rather than the whole training examples, SVM is extremely robust to intrusions. Moreover, SVM defines a penalty factor parameter for creating a tradeoff between the width of a decision boundary and the number of misclassified examples. Particularly, it provides high generalization ability when the number of features, m, is much more than the number of data points, n, is low ($m >> n$).

3.4. Ensemble learning

Ensemble learners are proposed to improve the performance of a single learner. The term "ensemble" refers to the combination of multi weak learners to build a strong learner. One of the most popular ensemble algorithms is Adaptive Boosting (AdaBoost) [26]. The AdaBoost trains several weak learners and then combine their predictions from all of them by a weighted majority vote to produce the final prediction. The main advantage of AdaBoost is outlier detection when examples are either mislabeled in the training dataset, or they are inherently ambiguous and hard to categorize. The Random Forest [27] is another ensemble learning method that composed of many classifications trees. Each tree is built based on randomly picked data features. After the forest (collection of trees) is formed, each tree gives a vote that indicates the tree's decision about the class of the instance. Finally, the class of a new instance is determined by majority voting or weighted voting. IsolationForest or iForest [28] as one of the ensemble learning algorithms detects intrusions based on two steps. In the first step, iForest builds isolation trees using sub-instances of the training dataset. The second step passes the test instances through isolation trees to get an anomaly score for each instance.

3.5. Multilayer perceptron

Multilayer perceptron (MLP) is one of ANN architecture that consists of three type of layers: input, hidden, and output layers. Each interconnection between neurons has associated with a scalar weight. The MLP model is trained by adjusting weights in order to minimize the error of misclassification. Traditional MLP typically contains one to three of hidden layers, whereas deeper MLP can have tens or even hundreds of hidden layers to support higher generalization capability in comparison to traditional MLP. However, deep neural networks require a large amount of data to train and build a general model. The networks generate overfitting if there is not enough quality labeled data. One of the approaches to solve this problem is using regularization methods. Dropout [29] is a well-known regularization method used to avoid overfitting. It provides a way to approximately combine many different neural networks efficiently. The key idea is to randomly drop out hidden and visible units (along with their connections) from the network during training. On the other hand, applying dropout to a deep network generate many thinned network models. At test time, a very simple approximated averaging method can be used to explicitly average the predictions from exponentially many thinned models.

3.6. Deep belief networks

Deep Belief Networks (DBN) can be constructed by stacking several Restricted Boltzmann Machines (RBMs). RBM is a specific energy model that uses a hidden layer to model a distribution over visible variables. The energy function for each visible unit v of vector $V = \{v_1, v_2, \ldots, v_n\}$ and for each hidden unit h of vector $H = \{h_1, h_2, \ldots, h_l\}$ is calculated as

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

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

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

where Z is a normalizing term or partition function. It is obtained by summing over all possible pairs of visible and hidden vectors as

$$Z = \sum_{v,h} e^{-E(v,h)} \qquad (3)$$

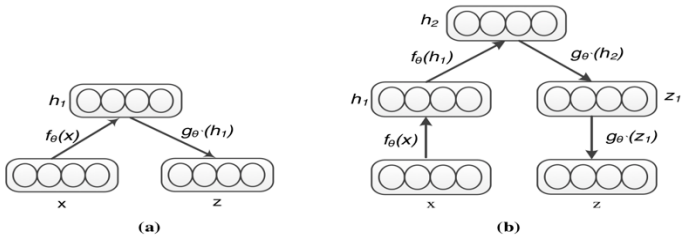

Figure 1. Example of: (a) shallow autoencoder; (b) stacked autoencoder

The goal of training an RBM is learning the parameter $\theta = (b, c, w)$ that the likelihood probability is reached the maximum value. The RBM learns to extract a hierarchical representation of the training data by modeling the joint distribution between an input vector x and the *l* hidden layer h_l as:

$$P(x, h_1, \dots, h_l) = \left(\prod_{k=0}^{l-2} P(h_k|k+1)\right) P(h_{l-1}, h_l) \qquad (4)$$

One of the main algorithms for training DBN is greedy layer-wise learning algorithm [20]. Based on this algorithm, the first layer of DBN is trained as an RBM. Then the first layer is used as the input data for the training of the second layer. After all layers are trained, all parameters of DBN are fine-tuned with respect to log-likelihood or a supervised training criterion.

3.7. Stacked autoencoder

Stacked AutoEncoder (SAE) is one of the most well-known deep learning models that is created by daisy-chaining autoencoders together [14]. Figure 1 (a) illustrates a shallow autoencoder that consists of one hidden layer. Autoencoder can encode a representation of the input layer into the hidden layer and then decode it into the output layer [11]. It includes two parts: encoder and decoder. For a given training dataset $X = \{x_1, x_2, \dots, x_m\}$ with *m* instances, where x_i is a d-dimensional feature vector, the encoder part maps an input vector x_i to a latent representation vector h_i by deterministic mapping f_θ as

$$h_i = f_\theta(x_i) = s(Wx_i + b) \qquad (5)$$

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 the sigmoid activation function denoted as

$$s(t) = \frac{1}{1 + exp^{-t}} \qquad (6)$$

where *t* can affect on the shape of the function.
The decoder part maps back the latent representation h_i to a reconstructed vector z_i of the same shape as *x*:

$$z_i = g_{\acute{\theta}}(h_i) = s(\acute{W}h_i + \acute{b}) \qquad (7)$$

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

The goal of autoencoder training is to minimize the difference between the input vector *x* and the reconstruction vector *z* which is called reconstruction error. Reconstruction error can be measured in different ways such as traditional mean squared error as

$$L(x, z) = \frac{1}{m} \sum_{i=1}^{m} \|x_i - z_i\|^2 \qquad (8)$$

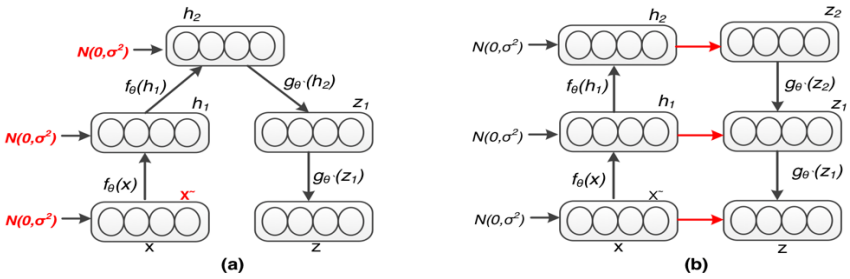

Figure 2. Example of two hidden-layer models:(a) stacked denoising autoencoder; (b) ladder Network

where *m* is the total number of the training dataset. If the input is interpreted as either bit vectors or vectors of bit probabilities, cross-entropy of the reconstruction is another alternative for loss calculation as:

$$L_H(x,z) = \sum_{i=1}^{m}[x_i log z_i + (1-x_i)\log(1-z_i)] \qquad (9)$$

This shallow autoencoder is stacked into a multi-layer autoencoder which is named SAE. Figure 1 (b) shows an example of an SAE model including two autoencoders. The first layer autoencoder gets the input of the SAE, and the hidden layer of the last autoencoder represents the output. The hidden layer of autoencoder at layer *l-1* becomes the input of the autoencoder at layer *l*. In SAE, the latent representation vector of layer *l*, h_l, is calculated by the following formula:

$$h_l = f_\theta(h_{l-1}) = s(W_l h_{l-1} + b_l) \qquad (10)$$

The decoder part reconstructs the vector z_{l-1} at layer *l-1* from the upper layer *l* as

$$z_{l-1} = g_{\hat{\theta}}(z_l) = s\left(\acute{W}_l z_l + \acute{b}_l\right) \qquad (11)$$

For the last layer *L*, the encoder and decoder parts are connected as $z_L := h_L$. In SAE, all information has to go through the highest layers to represent all details of input *x*. Moreover, intermediate hidden layer activations z_l cannot independently represent information as they only receive information from the highest layer.

3.8. Ladder network

The model structure of a ladder network is an autoencoder with skip connections from the encoder to the decoder for relieving the pressure to represent details in the higher layers of the model [30]. From the learning task aspect, the ladder network is similar to Stacked Denoising AutoEncoders (SDAEs) [13] but applied to every layer, not just inputs. In SDAEs, the original input is corrupted with noise and the objective of the network is to reconstruct the original uncorrupted input *x* from the corrupted $\tilde{x}$. On the other hand, SDAEs are trained to reconstruct the original inputs from a corrupted version in order to force the hidden layer to discover more robust features and prevent it from simply learning the patterns. Learning is based on minimizing the difference of the original *x* and its reconstruction *z* from the corrupted $\tilde{x}$.

Figure 2 shows an example of SDAEs and ladder network when they have two hidden layers. This figure represents the inference structure of a ladder network in comparison to SDAEs. The inference structure of SDAEs is similar to SAEs (Figure 2 (a)) but a Gaussian noise is added to each layer. In the ladder network (Figure 2 (b)), each hidden layer activations h_l can represent information independently from the other higher layers by lateral connections at each layer. Moreover, abstract invariant representations at the higher levels can be interpreted in the context of detailed information without the higher levels having to represent all the details. In this work, we compare our proposed approach to a semi-supervised learning algorithm with ladder work that presented in [30]. This algorithm combines supervised and unsupervised learning in deep neural networks. It achieves the state-of-the-art performance in semi-supervised MNIST and CIFAR-10 classification.

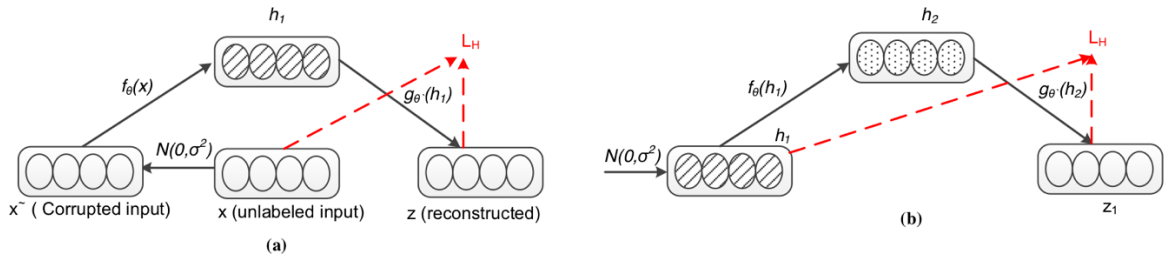

Figure 3. The unsupervised pre-training phase of the (a)first hidden layer; (b) second hidden layer of stacked denoising autoencoder

4. Stacked denoising autoencoders-based intrusion detection system

In this section, we describe how our proposed deep neural network architecture, Stacked Denoising AutoEncoders-based Intrusion Detection System (SDAEs-IDS), can address the network intrusion detection problem. The intrusion detection problem can be formulated as a classification problem as the main goal is assigning a label (i.e., normal or attack) to each unlabeled instance. The classification problem is considered as a binary classification if we categorize the data into normal or attack instances. However, we propose a multi-classification problem in this work, and our architecture classifies an instance into a normal or one of the four main attack classes.

Based on a series of experiments (Section 5.4), the best performing SDAEs model is built by daisy-chaining four autoencoders together where each autoencoder has 64 neurons. The output of an autoencoder at each current layer l is used as the input of the autoencoder at the next layer $l+1$. Finally, the output of the last layer is used as the output of the entire SDAEs model that contains five neurons corresponding to five classes. The training of the SDAEs model is based on the two following phases:

(1) Unsupervised pre-training phase: in the first phase, each hidden layer is trained as a Denoising AutoEncoder (DAE) by minimizing the error in reconstructing its input. The pre-training of the first hidden layer is performed by considering it as a regular DAE. The first layer uses the training dataset without labels as inputs (unsupervised) and creates a compressed representation $f_\theta(x)$ of corrupted input (Figure 3 (a)). Therefore, it is trained to reconstruct x from a stochastically corrupted (noisy) transformation of it ($\tilde{x}$). After the first hidden layer is completely pre-trained, the second hidden layer is trained by using the first layer s' output as input in order to learn a second level compressed representation $f_\theta(h_1)$ (Figure 3 (b)). This pre-training task sequence is iterated for all layers. So that the output of each DAE at layer l is the code vector h_l. In our experiments $h_l = s(b_l + W_l h_{l-1})$ is an ordinary neural network layer, with hidden unit biases b, and weight matrix W. Therefore, the output of this phase is a list of code vectors where each function h_l shows the reconstructed values in training the DAE corresponding to the hidden layer with same index l.

(2) Supervised fine-tuning phase: once all layers are pre-trained, the network goes through the second phase of training called supervised fine-tuning. The obtained parameters from the first phase are used as initialization for a network optimized with respect to a supervised training criterion. For a classification task, all parameters of the pre-trained network are fine-tuned to minimize the error in predicting the supervised target (e.g. class). For this, a softmax layer with n neurons firstly is added to the top of the pre-trained network where n is the number of labels in the dataset. Then, the pre-trained network is trained using both inputs and labels (x_i, y_i) of the training dataset (supervised) in order to minimize a cross-entropy loss function measuring the error between the classifier's predictions and the correct labels.

5. Experiments and results

The performance of the proposed SDAEs approach was tested in several experiments. This section firstly describes the KDD-CUP99 dataset [15] that we used for our experiments. Then, we investigate the influence of the various corruption levels on the proposed approach. After that, we report the results from a series of experiments to find the optimal model hyper-parameters on a separate test dataset. Finally, we compared our architecture with the classification methods based on shallow learning architecture described in Section 3. Moreover, we conduct a comparison between our proposed architecture and other deep architectures.

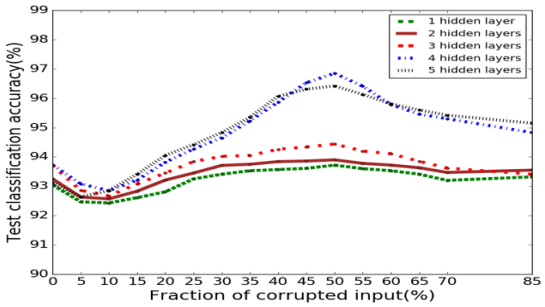

Figure 4. Effect of numbers of corruption level on the performance of SDAEs

5.1. Dataset description

The KDD-CUP'99 dataset [15] is a common benchmark for evaluation of network intrusion detection systems. The data consists of about 4 GB of compressed raw tcpdump data of network traffic for seven weeks [31]. Each instance of the dataset has 41 features which are described various aspects of the traffic pattern in a specific time interval. Those features consist of 38 continuous or discrete numerical features and 3 categorical features. 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 [18]**.** 10% of the original KDD-CUP'99 dataset contains 494,021 instances. For evaluating the trained model, we used the standard test dataset containing 311,029 instances with corrected labels. Each instance is labeled as either normal or as an attack with exactly one attack type among of four classes. Originally, there are 40 different attacks that are categorized into four main attack classes as follows:

- Denial of Service (DoS): is an attack in which an attacker attempts to prevent legitimate users access to a machine or make a memory or some computing resources too 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 networks or target host for the apparent purpose of circumventing its security controls.

Table 1 shows the number of instances for both normal and four attack classes in training and test datasets.

Table 1: Number of instances of each class in both 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

5.2. Pre-processing

The following pre-processing steps were performed 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 [35]. 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 categories. We used one hot encoding to convert five categorical classes into five binary classes, with only one active.

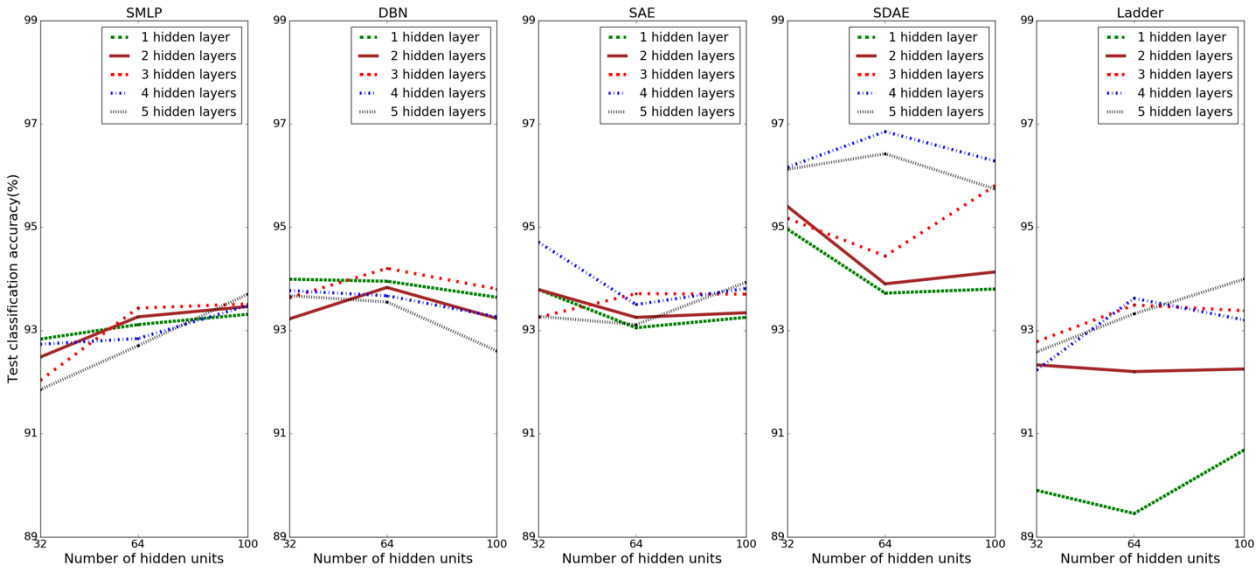

Figure 5. Effect of numbers of hidden layers and hidden units on the performance of MLP, DBN, SAEs, SDAEs and Ladder

(3) Feature Normalization: the numeric features must be normalized for removing the effect of original feature value scales. All numeric features values are ranged between 0 and 1.
(4) Redundancies reduction: one of the main problems of the data is a large number of duplicate records in both training and testing datasets that lead to the bias towards more frequent records. Therefore, the data should be cleaned as there are lots of redundancies in the dataset. To solve this problem, we removed all duplicate records in the training and test datasets 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.

5.3. Effect of corruption level

We investigated the influence of the input corruption on SDAEs performance. We added a stochastic corruption step operating on the input. The stochastic process randomly sets some of the inputs to zero. Then our model is trying to predict the corrupted values from the uncorrupted values for randomly selected subsets of missing inputs. The corruption level *cl%* means *cl* percentage of input values randomly are assigned to zero. Figure 4 shows that the test accuracy of SDAE in different corruption level from 0% to 85%. Note 0% corruption corresponds to SAEs (regular stacked autoencoders).

The results show that SDAEs appear to perform better that SAEs for a rather wide range of noise levels, regardless of the number of hidden layers. Moreover, SDAEs obtained the high detection accuracy when the corruption level is 50%.

5.4. Choice of hyperparameters

Designing an efficient model involves a challenging problem called hyper-parameters optimization. The performance of the model is changed depending on the value of hyper-parameters. In order to tune the hyperparameters for all models in this paper, we randomly split the 145,586 training instances into a14,000-instance validation dataset and used 131,586 instances as the training dataset. We used the training dataset for training a model and the validation dataset for validation of the model. After the best value of hyperparameters is selected, the final model is trained with all 145,586 instances.

The performance of neural networks highly depends on the network topology. The neural network topology represents the breadth (number of neurons per layer) and depth (number of layers) of the network. For this reason, we tried to find the network topology that is optimal to intrusion detection. The proposed neural network takes the input from the training dataset. Therefore, the input layer of networks represents all 117 features of the KDD-CUP99 dataset. The output layer represents five available classes of the KDD-CUP99 dataset. The number of hidden layers varied from one to five with 32, 64 and 100 neurons at each layer.

Table 2. List of hyper-parameters for shallow learning architecture

Model	Hyperparameter	Best value
K-NN	number of neighbors	50
	the algorithm used to compute the nearest neighbor	ball tree
	leaf size	500
DT	the maximum depth of the tree	500
SVM	kernel type	rbf
	penalty parameter	1.0
AdaBoost	learning rate	0.001
	the maximum number of estimators	50
Random Forest	the maximum depth of the tree	11000
	the number of trees in the forest	10
iForest	max number of instances	145586
	the number of base estimators in the ensemble	100
MLP	hidden layer size	32
	activation function	Relu
	optimizer	Adam
	learning rate	0.001

Figure 5 shows the test classification accuracy of the proposed deep models for the different number of hidden layers and units. As you can see in Figure 5, a higher number of hidden layers or units not necessarily improves the model accuracy. The SDAEs model with four hidden layers and 64 hidden units at each layer is best to other deep networks giving 96.85% testing accuracy for intrusion detection. The MLP model with five hidden layers and 100 units in each layer can get high accuracy (93.70%). The highest accuracy of DBN and SAE are 94.20% and 94.71% with three number of hidden layers and 64 units, respectively. The Ladder can also produce 93.62% classification accuracy when the network has four hidden layers and 64 units.

We consider four different network training strategies: without any pre-training (such as MLP and Ladder), with pre-training (such as DBN), with ordinary autoencoders pre-training (such as SAEs) and with denoising autoencoder pre-training (such as SDAEs). We clearly see that unsupervised pre-training gives substantially higher test classification accuracy than no pre-training for the same depth (Figure 5). Figure 5 shows that denoising pre-training being better than autoencoder pre-training. Moreover, the autoencoder pre-training being better than no pre-training. This result is a typical illustration of what is gained by pre-training deep networks with a good unsupervised criterion.

Optimization of shallow models was performed over key hyper-parameters and their values are given in Table 2. List of hyper-parameters for shallow learning architecture. Our proposed SDAEs-based deep architecture achieves the best result of accuracy when batch size and epochs of pre-training are 100 and 150, respectively. Moreover, the model got the highest accuracy with 100 batch size and 100 epochs of fine-tuning. There are a lot of choices to select activation functions in the hidden and output layers. We choose the Sigmoid function for the hidden layer in this work based on a series of preliminary experiments. In addition, the best optimizer for our neural network model in order to learn properly and tune the internal parameter is Adam based on our experiments. We also tune the learning rate parameter that is used in Adam with the grid search. Learning rate controls the speed of weight updating at the end of each batch. We tried a suite small standard learning rate from 0.001 to 0.3 in steps of 0.1. The best value for the learning rate is 0.001. In MLP, we explore the effect of dropout on the proposed model. The value of the dropout ranges from 0.0 to 0.9. We see that as dropout is 0.5, the model can get better accuracy.

5.5 Comparison with shallow and deep learning architectures

In order to design an efficient intrusion detection system performance, the system was trained and evaluated to classify five different attacks using shallow and deep learning architectures. Eight first rows of Table 3 shows the accuracy of shallow learning architecture. Maximum test classification accuracy was achieved at 92.33% for K-Nearest Neighbor (K-NN), followed by 93.74% for CART decision tree, 92.86% for Support Vector Machine (SVM), 87.95% for AdaBoost, 92.32% for Random Forest, 90.89% for iForest, 93.17% for Multi-Layer Perceptron (MLP) and 93.13% for AutoEncoder (AE). The two best performing classifiers were CART and MLP, respectively. Although none of these shallow learning architecture's performance was as high as Stacked Deionising AutoEncoders (SDAEs). In addition, Table 3 shows the classification accuracy of SDAEs-IDS in comparison to the other deep architectures: Multi-Layer Perceptron (MLP), Deep Belief Network

(DBN), Stacked AutoEncoder (SAEs) and Ladder [30], AutoEncoder+DBN$^{10-10}$ [18] and DBN^{4} [22]. MLP is tested with and without dropout to investigate how dropout can affect on the accuracy.

Table 3. Comparison of Stacked Denoising AutoEnocders (SDAEs) with shallow and deep learning architectures

Model	Test classification accuracy(%)
K-NN	92.36
CART	93.74
SVM	92.86
AdaBoost	87.95
Random Forest	92.32
iForest	90.89
MLP	93.17
AE	93.13
MLP with dropout	93.70
MLP without dropout	93.28
SAEs	94.71
DBN	94.20
Ladder [33]	93.62
AutoEncoder+DBN$^{10-10}$ [19]	92.10
DBN^{4} [23]	93.49
SDAEs	**96.85**

Results show that the accuracy of MLP is improved by considering dropout as it can avoid overfitting. The best performance is achieved by Stacked Denoising AutoEncoder (SDAEs) among the proposed deep models as shown in Table 3. The representation ability of shallow learning architecture or neural network is limited in comparison to the deep neural networks. Another reason for achieving the better accuracy by SDAEs is a denoising task. SDAEs is forced to discover more robust features when is reconstructed the input from a corrupted version of it.

6. Conclusion

We have introduced a deep neural network architecture for improving the performance of anomaly-based intrusion detection systems. Our architecture used Stacked Denoising AutoEncoders (SDAEs) that is formed by stacking several autoencoders to improve the representation capability of learned features from training data. To the best of our knowledge, currently there are no existing works on using SDAEs to detect intrusions in networks. SDAEs can extract robust features by reconstructing the data from the corrupted version of it. Moreover, the proposed architecture is designed to perform first unsupervised pre-training and then followed by supervised fine-tuning for intrusion detection.

A series of experiments were performed to evaluate the proposed architecture on the KDD-CUP'99 dataset. The empirical results support the following conclusions: unsupervised pre-training gives substantially higher test classification accuracy than no pre-training. Denoising pre-training being better than autoencoder pre-training. Our SDAEs-based architecture can produce a high accuracy compared with other deep architectures and shallow learning architectures for intrusion detection. We also presented a series of experiments aimed at evaluating the link between the performance of our deep neural network architecture and aspects of their topology such as depth and breadth. Finally, we explored the effect of corruption levels on the proposed architecture performance. Overall the results should have a wide interest in the IDS community.

7. References

[1] B. B. Zarpelo, R. S. Miani, C. T. Kawakani, S. C. Alvarenge, "A survey of intrusion detection in internet of things", Journal of Network and Computer Applications, vol. 84, No. C, pp. 25-37, 2017.

[2] R. Mitchell, I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems", Journal of ACM Computing Surveys, vol. 46, no. 55, pp. 1-29, 2014.

[3] S. Raza, L. Wallgren, T. Voigt, "Svelte: Real-time intrusion detection in the internet of things", Journal of Ad Hoc Elsiver, vol. 11, Issue 8, pp. 2661-2674, 2013.

[4] C. A. S., S. Gupta, "An effective model for anomaly ids to improve the efficiency", International Conference on Green Computing and Internet of Things (ICGCIoT), pp. 190-194, 2015.

[5] A. A. Aburomman, M. B. Ibne Reaz, "A novel svm-knn-pso ensemble method for intrusion detection system", Journal of Applied Soft Computing, vol. 38, no. C, pp. 360-372, 2016.

[6] F. Hosseinpour, P. Vahdani Amoli, F. Farahnakian, J. Plosila, T. Hämäläinen, "Artificial immune system based intrusion detection: Innate immunity using an unsupervised learning approach", Journal of Digital Content Technology and Its Applications, vol. 8, no. 5, pp. 1-12, 2014.
[7] S. Akbar, J. A. Chandulal, K. N. Rao, G. S. Kumar, "Improving network security using machine learning techniques", in the proceeding of IEEE International Conference on Computational Intelligence and Computing Research, pp. 1-5, 2012.
[8] J. Cannady, "Artificial neural networks for misuse detection", in the proceeding of National Information System Security Conference, pp. 443-456, 1998.
[9] E. Hodo, X. Bellekens, A. Hamilton, P. L. Dubouilh, E. Iorkyase, C. Tach-tatzis, R. Atkinson, "Threat analysis of IoT networks using artificial neural network intrusion detection system", in the proceeding of International Symposium on Networks, Computers and Communications (ISNCC), pp. 1-6, 2016.
[10] M. Z. Alom, V. Bontupalli, T. M. Taha, "Intrusion detection using deep belief networks", in the proceeding of National Aerospace and Electronics Conference (NAECON), pp. 339-344, 2015.
[11] Y. Bengio, "Learning deep architectures for AI", Journal of Found. Trends Mach. Learn, vol. 2, no. 1, pp.1-127, 2009.
[12] J. Kim, J. Kim, H. L. T. Thu, H. Kim, "Long short-term memory recurrent neural network classifier for intrusion detection", in the proceeding of International Conference on Platform Technology and Service (PlatCon), pp. 1-5, 2016.
[13] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion", Journal of J. Mach. Learn. Res, vol. 11, pp. 3371-3408, 2010.
[14] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures", in the proceeding of International Conference on Unsupervised and Transfer Learning Workshop, pp. 37-50, 2011.
[15] Nsl-kdd dataset for network-based intrusion detection systems, 2009. URL Available on http://nsl.cs.unb.ca/NSL-KDD/
[16] J. Jiang, C. Zhang, M. Kamel, "RBF-based real-time hierarchical intrusion detection systems", in the proceeding of the International Joint Conference on Neural Networks, pp. 1512-1516, 2013.
[17] F. Farahnakian, J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system", in the proceeding of the 20th International Conference on Advanced Communication Technology (ICACT), pp.178-183, 2018.
[18] L. Yuancheng, M. Rong, 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.
[19] U. Fiore, F. Palmieri, A. Castiglione, A. De Santis, "Network anomaly detection with the restricted Boltzmann machine", Journal of Neurocomputing, vol. 122, pp. 13-23. 2013.
[20] G. E. Hinton, S. Osindero, Y.-W. Teh, "A fast learning algorithm for deep belief nets", Journal of Neural Comput., vol. 18, no. 7, pp. 1527-1554, 2006.
[21] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, "Exploring strategies for training deep neural networks", Journal of J. Mach. Learn. Res., vol. 10, pp.1-40, 2009.
[22] N. Gao, L. Gao, Q. Gao, H. Wang, "An intrusion detection model based on deep belief networks", in the proceeding of the Second International Conference on Advanced Cloud and Big Data, pp. 247-252, 2014.
[23] S. Potluri, C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system", in the proceeding of the 21st International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-8, 2016.
[24] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, "Energy-aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers", in the proceeding of the 6th International Conference on Utility and Cloud Computing, pp. 256-259, 2013.
[25] J. R. Quinlan, "Induction of decision trees", Journal of Machine Learning, vol. 1, pp. 81-106, 1986.
[26] Y. Freund, R. E. Schapire, "Experiments with a new boosting algorithm", in the proceeding of the 13 International Conference on Machine Learning, Morgan Kaufmann, pp. 148-156,1996.
[27] L. Breiman, "Random forests", Journal of Machine Learning, vol. 45, pp. 5-32, 2001.
[28] F. T. Liu, K. M. Ting, Z.-H. Zhou, "Isolation-based anomaly detection", Journal of ACM Trans. Knowl. Discov. Data, vol. 6, no. 1, pp. 3:1-3:39, 2012.
[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", Journal of J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929-1958, 2014.
[30] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, T. Raiko, "Semi-supervised learning with ladder networks", in the proceeding of Advances in Neural Information Processing Systems, pp. 3546-3554, 2015.
[31] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory", Journal of ACM Trans. Inf. Syst. Secur. vol. 3, no. 4, pp. 262-294, 2000.
[32] K. Alrawashdeh, C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning", in the proceeding of 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 195-200, 2016.