Future Generation Computer Systems 98 (2019) 308–318

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A deep learning approach for proactive multi-cloud cooperative intrusion detection system

Adel Abusitta, Martine Bellaiche [*], Michel Dagenais, Talal Halabi

Department of Computer and Software Engineering, Ecole Polytechnique de Montr'eal, Montr'eal, Canada

HIGHLIGHTS

- We propose a proactive multi-cloud cooperative Intrusion Detection System (IDS).
- The proposed solution can accelerate the decision making in real-time environments.
- The decisions about intrusions are done without applying aggregation methods.
- The results show the effectiveness of our approach compared to other approaches.

ARTICLE INFO

Article history:
Received 30 October 2018
Received in revised form 5 February 2019
Accepted 18 March 2019
Available online 29 March 2019

Keywords:
Intrusion detection systems
Deep learning
Cloud computing
Security

ABSTRACT

The last few years have witnessed the ability of cooperative cloud-based Intrusion Detection Systems (IDS) in detecting sophisticated and unknown attacks associated with the complex architecture of the Cloud. In a cooperative setting, an IDS can consult other IDSs about suspicious intrusions and make a decision using an aggregation algorithm. However, undesired delays arise from applying aggregation algorithms and also from waiting to receive feedback from consulted IDSs. These limitations render the decisions generated by existing cooperative IDS approaches ineffective in real-time, hence making them unsustainable. To face these challenges, we propose a machine learning-based cooperative IDS that efficiently exploits the historical feedback data to provide the ability of proactive decision making. Specifically, the proposed model is based on a Denoising Autoencoder (DA), which is used as a building block to construct a deep neural network. The power of DA lies in its ability to learn how to reconstruct IDSs' feedback from partial feedback. This allows us to proactively make decisions about suspicious intrusions even in the absence of complete feedback from the IDSs. The proposed model was implemented in GPU-enabled TensorFlow and evaluated using a real-life dataset. Experimental results show that our model can achieve detection accuracy up to 95%.

1. Introduction

The complex architecture of cloud computing systems make them vulnerable to many kinds of attacks. Recently, promising results have shown that the use of cooperative Intrusion Detection Systems (IDSs) can improve the detection accuracy compared with the traditional single IDSs [1–3]. This is due to the fact that it is becoming largely difficult for a single IDS to detect all existing attacks [2,3], due to its limited knowledge of such attack patterns and implications. The cooperation among IDSs that belong to different Cloud Providers (CPs) can be achieved by allowing them to exchange their intrusion analysis feedback and exploit each other's expertise to cover unknown threat patterns, thus achieving mutual benefits.

Unfortunately, there are considerable delays associated with the use of existing cooperative IDSs. These delays are mostly due to the computation complexity of using the aggregation algorithms such as Dempster–Shafer Theory (DST), and also the large geographic distances that might separate the IDSs. In fact, each IDS, after receiving feedback from consulted IDSs about a suspicious intrusion, is required to use a suitable feedback algorithm, in order to make a final decision about the suspicious intrusion. The aggregation technique is usually costly in terms of computation time and depends on many factors such as the number of consulted IDSs, the IDSs' expertise and trust levels [2,3]. Moreover, due to uneven IDSs' connections and Internet speeds and other unknown factors (e.g., busy IDSs, compromised IDSs), there is no guarantee that feedback will be received simultaneously. Thus, decisions on whether or not to rise an alarm about suspicious intrusions might be excessively delayed due to the missing feedback of a single IDS. Hence, the decisions generated

* Corresponding author.
E-mail addresses: adel.abusitta@polymtl.ca (A. Abusitta), martine.bellaiche@polymtl.ca (M. Bellaiche), michel.dagenais@polymtl.ca (M. Dagenais), talal.halabi@polymtl.ca (T. Halabi).

https://doi.org/10.1016/j.future.2019.03.043
0167-739X/© 2019 Published by Elsevier B.V.

by the cooperative IDS are ineffective in a real-time setting, making it unsustainable.

To overcome these limitations, we design a proactive multi-cloud cooperative IDS that integrates a machine learning-based approach. The proposed model exploits the historical feedback to predict the status of suspicious intrusions. This can be done proactively without having to apply any aggregation method on consulted IDSs' feedback, nor having to wait until receiving all the feedbacks from the consulted IDSs, i.e., only partial or incomplete feedback can be used to predict the status of suspicions intrusions. This, in turn, makes our solution reliable and feasible in real-time environments, where decisions about intrusions must be taken rapidly in order to effectively apply the required action measures at the right time. However, due to the structure of used data, which are collected from many IDSs with different expertise and trust levels, and due to the fact that decisions regarding suspicious intrusions must be taken despite the existence of missing feedback, the use of traditional machine learning techniques (e.g., SVM) for such a problem produces inaccurate results in terms of prediction accuracy [4,5]. For this purpose, we propose a deep learning approach that consists of multiple layers to learn the representation of the data with multiple levels of abstraction [4,5]. This allows us to learn how to obtain a "good" representation of the data to be used later as inputs to supervised machine learning techniques in order to achieve better detection accuracy [6–10].

More particularly, our model is based on Stacked Denoising Autoencoders (SDAE), where a denoising autoencoder is used as a building block to train a deep network [9,10]. Our model exploits the fact that a denoising autoencoder can learn how to reconstruct original inputs giving partial data inputs, by allowing deep neural networks to learn (during unsupervised pre-training stage) how to extract features that are robust to incomplete IDSs' feedback. Such robust features can be seen as useful representations of data to yield a better intrusion detection accuracy in such real-time environments. This makes our detection model proactive on two levels: (1) by making decisions about suspicious intrusions even with missing feedback, and also (2) by making decisions without having to apply any aggregation method on consulted IDSs' feedback. Our contributions are summarized as follows:

- Proposing a cooperative intrusion detection model (based on stacked denoising autoencoders) that enables making decisions about suspicious intrusions even with partial IDS's feedback. This, in turn, accelerates the decision making in real-time environments.
- Designing a proactive multi-cloud cooperative IDS, which allows us to make decisions about suspicious intrusions proactively, i.e., without the need to apply aggregation methods on IDSs' feedback.
- Proposing an approach to extract robust features that yield a better performance in cooperative intrusion detection.
- Evaluating the effectiveness of the proposed solution using a real-life dataset, and comparing our results with those generated with other approaches.

Our model has been implemented in GPU-enabled TensorFlow and evaluated using a real-life dataset. The results show the effectiveness of the proposed approach in terms of enhancing the accuracy of the detection compared to the state-of-the-art deep architectures: Multi-Layer Perceptron (MLP) and Stacked Auto Encoders (SAE).

The remainder of this paper is organized as follows. In Section 2, the related work is introduced. The proposed cloud-based cooperative intrusion detection system is presented in Section 3. Experimental results show the effectiveness of the proposed approach in Section 4. Finally, Section 5 concludes the paper.

2. Background and related work

Cloud-based IDSs can be divided into two categories: signature-based and anomaly-based [11]. The former compares suspicious activity with known attack patterns. To make signature-based IDSs effective, the database of signatures should be updated repeatedly. On the other hand, anomaly-based IDSs raise alarms when unexpected behaviors have been detected. Anomaly-based IDSs are efficient in detecting unknown attacks. A database of known attacks is not required for this approach. However, the shortcoming of this solution lies in the relative high false positive rate compared with the signature-based technique [2]. In fact, IDSs may use both techniques to enhance their detection accuracy. However, the detection accuracy is limited by the amount of information held by the IDSs. Recent research has shown that the collaborative detection can enhance the detection rate up to 60% [2,12]. This section presents the state-of-the-art of the cloud-based cooperative IDSs.

In multi-cloud cooperative IDSs, Lo et al. [13] propose a detection approach that exchanges alerts among different nodes (i.e., hosts) whenever an attack is detected. They adopt a rule-based technique to detect SYN attacks by fetching the threshold for rule patterns through the initial rule establishment phase. The advantage of this approach is that it is can be used to distribute the detection overhead between the nodes. Also, Teng et al. [14] proposed an approach that aggregates two detectors: a feature detector and a statistical detector. The former uses SNORT to separate events based on transmission control protocols such as TCP. The later cooperates with the former by using packets from it to decide whether an event is an attack or not. A predefined threshold is set for this purpose. An attack is considered in cases where the rate of packets obtained exceeds the threshold.

Deep learning approaches for intrusion detection system were recently proposed in several works [15–18] [19–21]. However, these approaches have the same problem as a single IDS, they cannot detect all existing attacks [2,3] due to their limited knowledge about all attack patterns and implications.

Deep autoencoders have been used recently in many works. For example, Jia et al. [22] present a five-category classification of brain images based on deep stacked sparse autoencoder. They propose a method with deep autoencoders to detect the brain pathologies. Also, Zhang et al. [23] propose a new 7-layer SAE based deep neural network for cerebral microbleed detection. The results showed that this method is promising and gives better results than three state-of-the-art methods.

Man and Huh [24] and Singh et al. [25] designed a cloud-based cooperative IDS between different regions. The model allows alerts to be exchanged from multiple detectors. Also, knowledge are allowed to be exchanged between interconnected clouds. Ghribi [26] proposed a middle-ware IDS that allows a cooperation between different layers: Hybervisor-based, Network-based and VM-based IDS. If an intrusion is found in a layer, the propagation of such intrusion to the other layers could be prevented. Moreover, Chiba et al. [27] designed a network-based cooperative IDS, which is used to identify network intrusions in the cloud environment. This can be achieved through monitoring network traffic while maintaining and/or guaranteeing Quality of Services (QoS) and performance [27].

In a multi-cloud environment, Dermott et al. [1] proposed a cooperative intrusion detection in multi-cloud environments. The Dempster–Shafer theory of evidence is used to collect and aggregate the beliefs received by the monitoring entities. The received beliefs are used to make the final decision regarding a possible intrusion. This approach has a limitation: its centralized-based architecture, whereby a trusted third-party is responsible for aggregating feedback and managing cloud-based intrusion detection system.

A trust-based cooperative IDS was proposed for a non-cloud environment. For example, Fung and Zhu [2] designed trust-based cooperative IDS. Through cooperation, a local Intrusion Detection System (IDS) can detect new attacks that may be unknown to other IDSs. They use the diagnosis from different IDSs to perform intrusion detection. They present a system architecture of a collaborative IDS in which trustworthy feedback aggregation is a key component [2]. Also, Zhu et al. [28,29] designed an incentive-based communication protocol, which provides IDS nodes incentives to send feedbacks to their peers thus preventing malicious behaviors. The limitation of the existing trust-based cooperative IDS is that they consider a consultation request to be sent to a large number of IDSs in order to get a feedback. However, this approach causes extra overhead as some IDSs are needlessly consulted. We address this point by allowing an IDS to make a decision regarding suspicious intrusions without having to wait until receiving all the feedbacks from the consulted IDSs. Also, Abusitta et al. [3] propose a framework that allows cloud-based IDSs to distributively form trustworthy IDSs communities. For this purpose, they propose an algorithm based on the cooperative game theory: it allows a set of IDSs to cooperatively set up their coalition in such a way to allow their individual detection accuracy increase, despite the presence of untrusted IDSs.

In summary, There are considerable delays associated with the use of existing cooperative IDSs. These delays are mostly due to the computation complexity of using the aggregation algorithms such as Dempster–Shafer Theory (DST), and also the large geographic distances that might separate the IDSs. In fact, each IDS, after receiving feedback from consulted IDSs about a suspicious intrusion, is required to use a suitable feedback algorithm, in order to make a final decision about the suspicious intrusion. The aggregation technique is often costly in terms of computation time and depends on many factors such as the number of consulted IDSs, the IDSs expertise and trust levels. Moreover, due to uneven IDSs connections and Internet speeds and other unknown factors (e.g., busy IDSs, compromised IDSs), there is no guarantee that feedback will be received simultaneously. Thus, decisions on whether or not to rise an alarm about suspicious intrusions might be considerably delayed due to the missing feedback of a single IDS. Hence, the decisions generated by the cooperative IDS are ineffective in a real-time setting, making it unsustainable.

Overall, for a multi-cloud environment, a proactive cooperative IDS has yet to be designed. This proactive approach is useful in real-time environments, where fast decisions must be taken. In this work, the proactive aspect was achieved in two ways: (1) making decisions about suspicious intrusions proactively without having to apply aggregation methods on IDSs' feedback; and (2) making decisions about suspicious intrusions in the absence of complete IDS feedback.

3. The proposed proactive multi-cloud cooperative IDS

3.1. System model

In multi-cloud cooperative IDS, a cloud-based IDS can consult other cloud-based IDSs about suspicious intrusions and aggregate the received feedback to make a decision using an aggregation algorithm. Fig. 1 shows the high level architecture of the proposed cooperative IDS. As illustrated, CP4 has a list of several CPs which are open to cooperation. CP4 sends a consultation request to its whitelist for intrusion diagnosis. The recipient IDSs send their feedback (attack or not) to the CP4's IDS, which then uses an aggregation method such as the Dempster–Shafer Theory (DST) [30] for feedback aggregation.

In this paper, we are looking for a proactive approach in order to reduce delays associated with the architecture of multi-cloud cooperative IDS. These delays mostly come from the large

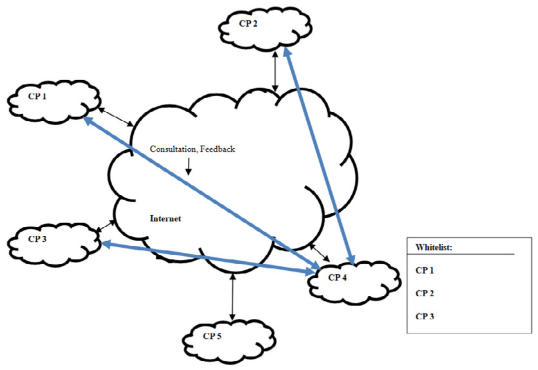


Fig. 1. Architecture of the proposed cooperative IDS.

geographic distances that might separate cloud-based IDSs and the computation complexity of using the aggregation algorithms, especially when the number of consulted IDSs is large [2,3]. To this end, we propose a proactive multi-cloud cooperative IDS that is based on machine learning approach. The proposed model exploits the historical feedback to predict the status of suspicious intrusions. In this solution, decisions can be taken proactively without having to apply an aggregation method on the received feedback an it eliminates waiting time to obtain all the feedbacks from the IDSs.

It is worth noting that the problem in this paper is about the detection under incomplete information. A Denoising Autoencoder (DA) is a good candidate for such a problem when it is used as a building block to train a deep neural network [9,10]. The power of DA lies in its ability to learn how to reconstruct IDSs feedback from partial feedback. This allows us to proactively make decisions about suspicious intrusions even in the absence of complete feedback from the IDSs.

The subsequent subsections will present the proposed model. We first present the concept of traditional autoencoders followed by the proposed feedback-based denoising autoencoders, where we will show how to train an autoencoder using unsupervised data to enable the extraction of the features that are robust to incomplete feedback. Then, we will introduce the stacked denoising autoencoders and fine-tuning process and show how to stack denoising autoencoders in order to build deep neural networks to be used to proactively make decisions about suspicious intrusions.

3.2. The traditional autoencoders

An autoencoder is an unsupervised learning approach that is used to learn efficient and reliable data codings [31]. It is used to pre-train each layer in a deep neural network in order to obtain better initial weights that lead to a better performing classification [8]. Researchers have seen that weights initialization using autoencoders can improves the performance of deep neural networks compared to random initialization [8].

The structure of an autoencoder is shown in Fig. 2. The dimensions for both input (IDSs' feedback) and output (reconstruction of IDSs' feedback) are the same as shown in the figure. Labels (i.e., +1) in Fig. 2 represent a bias vector. The use of biases in a neural network increases the capacity of the network to solve problems [32].

An autoencoder is used as a building block for deep networks [8]. In particular, it takes input vector (IDSs' feedback) $x \in [0,1]^d$, where d is the vector dimension, and maps it to a hidden representation $h \in [0,1]^{d'}$ using the following equation:

$$h = f_\theta(x) = Sigmoid(W * x + b) \qquad (1)$$

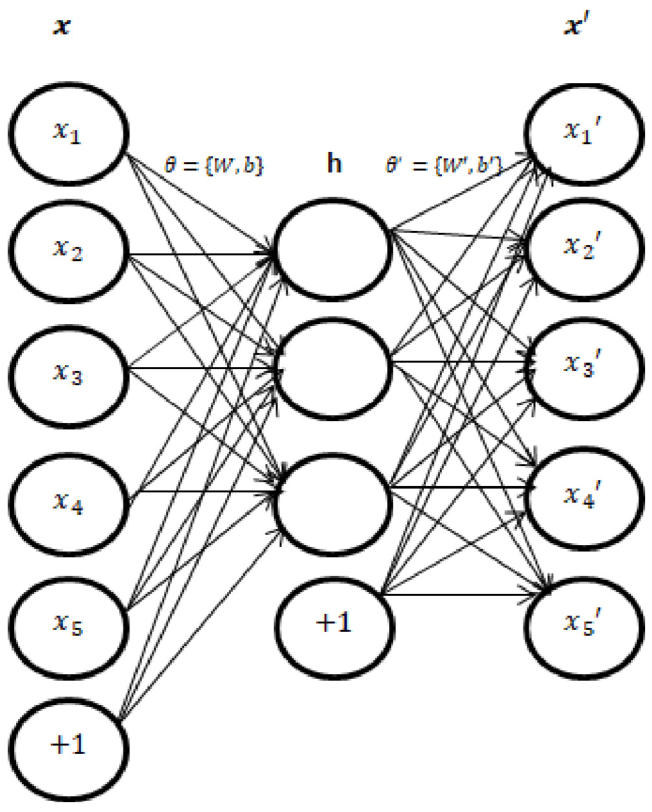


Fig. 2. Example of an autoencoder.

$\theta = \{W, b\}$, W is a weight matrix and b is a bias vector. After that, the resulting hidden layer representation h will be reconstructed to the output layer x' using a decoding function as follows:

$$x' = g_{\theta'}(h) = Sigmoid(W' * h + b') \tag{2}$$

$\theta' = \{W', b'\}$, W' and b' are a weight matrix and a bias vector of the reverse mapping, respectively. The weight matrix W' of the reverse mapping may optionally be constrained by $W' = W^T$, in which case the autoencoder is said to have tied weights [8,10]. Each training x is thus mapped to a corresponding h and a reconstruction x'.

The purpose of the model is to optimize the parameters (θ and θ') of the model, so that the reconstruction error between input and output can be minimized. The following optimization problem is used for this purpose:

$$\begin{aligned}\theta^*, \theta'^* &= arg \ \ min_{\theta,\theta'} \ \frac{1}{n}\sum_{i=1}^{n} L(x^{(i)}, x'^{(i)}) \\ &= \frac{1}{n}\sum_{i=1}^{n} L(x^{(i)}, g_{\theta'}(f_\theta(x^{(i)})))\end{aligned} \tag{3}$$

where L is a loss function. In an unsupervised learning model, a loss function is a measure of how close the reconstructed input is to the original input [32].

Since the input vector x is a binary vector (an IDS's feedback is either 0 or 1), a reconstruction cross-entropy is used as a loss function [33]. Thus, the loss function will be described as follows:

$$L(x, x') = -\sum_{i=1}^{d}[x_i log x'_i + (1 - x_i) log(1 - x'_i)] \tag{4}$$

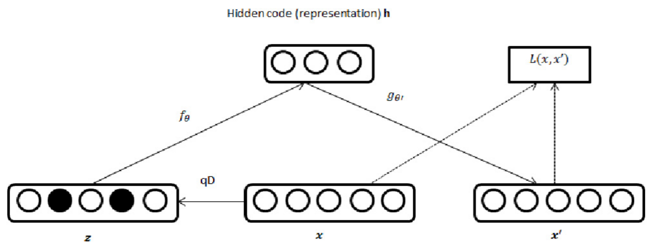


Fig. 3. IDS-based denoising autoencoder architecture.

3.3. The proposed IDS-based Denoising Autoencoders

In order to make an autoencoder robust to the incomplete input (IDSs' feedback) and also to prevent it from just learning the identity of the input, the autoencoder should be trained to reconstruct its IDS's feedback even if the feedback does not represent the whole IDSs' feedback (some feedback are not available). The autoencoder that deals with corrupted version of input is called a denoising autoencoder [10]. This can be achieved by adding noise to the initial input x before passing it to the hidden layer in order to reconstruct x, where x are IDSs' feedback. Thus, a partially corrupted version z will be obtained from x as follows:

$$z = qD(x) \tag{5}$$

Where qD is a corruption process [10]. In our model, we use Masking Noise (MN) for the corruption process as it is considered a useful method to represent incomplete IDS's feedback [9]. In MN noise, a fraction υ (selected at random) of each input x is forced to 0, while the others remain untouched. In fact, other noise can also be used (e.g., Gaussian noise). However, MN is more useful to simulate incomplete IDSs' feedback [9] since the noise will change only partial feedback.

The traditional autoencoder is then used to take corrupted data z and try to learn how to reconstruct x. This is done by allowing the input z to be mapped to a hidden representation as:

$$h = f_\theta(z) = Sigmoid(W' * z + b') \tag{6}$$

Note that we selected z as input instead of x as a traditional autoencoder was used. The result of the above equation h is then used to reconstruct x' as follows:

$$x' = g_{\theta'}(h) = Sigmoid(W * h + b) \tag{7}$$

The denoising autoencoder architecture is described in Fig. 3. As given in the traditional autoencoder, the parameters are trained to minimize the average reconstruction error:

$$\begin{aligned}\theta^*, \theta'^* &= arg \ \ min_{\theta,\theta'} \ \frac{1}{n}\sum_{i=1}^{n} L(z^{(i)}, x'^{(i)}) \\ &= \frac{1}{n}\sum_{i=1}^{n} L(z^{(i)}, g_{\theta'}(f_\theta(z^{(i)})))\end{aligned} \tag{8}$$

The above equation is then re-written by considering cross-entropy as loss function:

$$L(z, x') = -\sum_{i=1}^{d}[z_i log x'_i + (1 - z_i) log(1 - x'_i)] \tag{9}$$

The training algorithm of the proposed IDS-based denoising autoencoder is presented in Algorithm 1. As we can see in the algorithm, for the raw inputs x, randomly selected parts of them will be set to 0 as corrupted inputs z. The corrupted input z will be then encoded to the hidden code and then reconstructed to the output. However, at this point, x' is a deterministic function of z rather than x. The reconstruction is computed between z and x is denoted by $L(x, x')$ (Same as with the autoencoder). The parameters of the model are initialized randomly and then optimized by stochastic gradient descent algorithms.

Algorithm 1: IDS-based Training Denoising Autoencoder

```
procedure TRAINING-DA($x, l, e, b, \theta$)
    • $x$= [$x_1, x_2, ...x_n$]: Input IDSs' feedback
    • $l$: learning rate
    • $e$: Amount of epoches to be iterated
    • $b$: Amount of batches
    • $\theta$={$W, b, b_h$}
    for $i$ from 0 to $e$ do
        for $j$ from 0 to $b$ do
            • $z = CorruptInput(x, c)$: c is corruption level
            • $h = s(z * W + b)$
            • $x' = s(h * W^T + b_h)$
            • $L(x, x') = -\sum_{i=1}^{d}[x_i log x'_i + (1 - x_i) log(1 - x'_i)]$
            • $cost = mean(L(x, x'))$
            • g=compute the gradients of the cost w.r.t $\theta$
            for $\theta_i, g_i \in (\theta, g)$ do
                • $\theta_i = \theta_i - l * g_i$
            end
        end
    end
end procedure
```

3.4. The proposed IDS-based stacked denoising autoencoders

An autoencoder is used as a building block for unsupervised training of deep networks [8]. Such an architecture is called as Stacked AutoEncoder (SAE) [8]. The purpose of having an autoencoder as a building block to the deep network is to support the pre-training process [8], which is used to initialize the parameters of the deep network before applying supervised learning algorithms. Although Restricted Boltzmann Machine (RBM) (RBM) [34] can also be used as a building block for deep networks, SAE is considered much simpler and easier than RBM [8].

The initialization of deep networks parameters using SAE leads to a better classification accuracy compared to Multi-Layer Perceptron (MLP) [35], which does not use a pre-training process (all of the parameters are initialized randomly). In SAE, each layer's input is obtained from the previous layer's output. Fig. 4 shows the first step of a SAE. It trains an autoencoder on raw input x to learn h_1 by minimizing the reconstruction error $L(x, x')$. Once the parameters W_1 and W'_1 of the autoencoder are obtained using the gradient descent algorithm [36], a new layer can be added, as shown in Fig. 5. In Fig. 5, the hidden representation h_1 will be an input to train the second autoencoder by minimizing the reconstruction error $L(h_1, h'_1)$. This figure clearly shows that the error is calculated between the output h'_1 and the previous latent feature representation h_1.

Once the parameters W_2 and W'_2 of the second autoencoder are obtained using the gradient descent algorithm, the new hidden representation h_2 will become the raw input for the next autoencoder. These steps can be repeated until the last hidden layer (the output of the last autoencoder) is reached. To determine the number of layers required to build a IDS-based deep neural network, we keep adding layers until no improvement has been achieved in the test error or we start to overfit our training set [8–10,37]. The last hidden layer represents a "good" representation of data that can be used as input for any supervised learning algorithm [8,38].

The procedure to train a deep network using the proposed IDS-based denoising autoencoders as a building block (Fig. 6) is

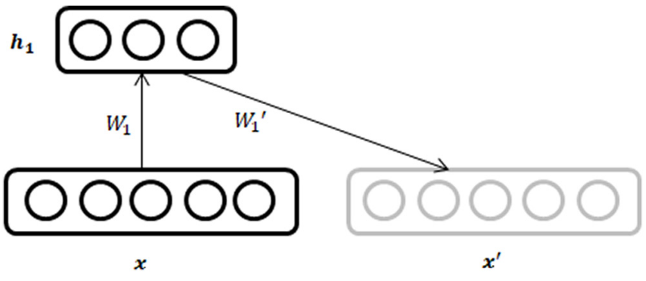


Fig. 4. Step 1 in Stacked Denoising Autoencoders.

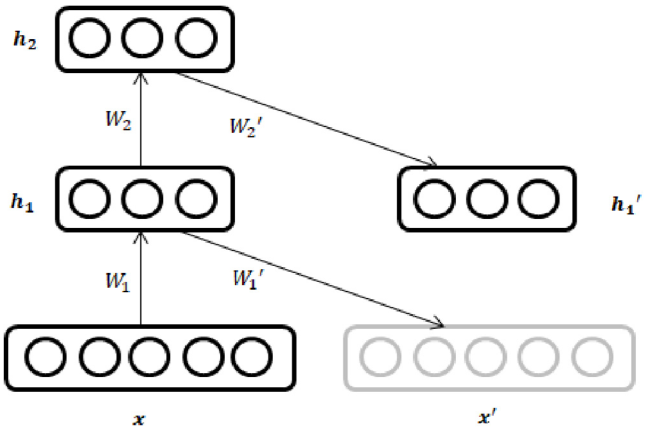


Fig. 5. Step 2 in Stacked Denoising Autoencoders.

similar [9,10]. Its only difference resides in the fact that each layer is trained, i.e., to minimize the criterion in eq. 8 instead of eq. 3. Note that the corruption process qD is used only during training, but not to propagate representations from the raw input to higher-level representations. Note also that when the layer k is trained, it receives as input the uncorrupted output from the previous layers. In Fig. 5, after training the first block of denoising autoencoder as shown in Fig. 3, the learned encoding function f_θ is used on clean input x as shown in Fig. 6 (left). The resulting representation is used to train the second block of denoising autoencoder as shown in Fig. 6 (right), to learn the second layer encoding function $f_\theta^{(2)}$. The process can be repeated for the next layers (Fig. 7).

The pre-training process is shown in Algorithm 2: once the mapping function f_θ is learnt using Algorithm 1, the function will be used on IDSs' (complete) feedback to generate the values of the second layer (first hidden layer). These values will then be used as inputs to train the next layer (using Algorithm 1) to generate the second mapping function $f_\theta^{(2)}$. This function will be used to generate the values of the third layer (second hidden layer). These values will also be used as inputs to train the next layer (using Algorithm 1) to generate the third mapping function $f_\theta^{(3)}$. The same steps can be applied for the whole set of hidden layers.

3.5. The proposed IDS-based fine-tuning and detection

Once the last hidden layer is trained as shown in the previous section, the parameters (generated from Algorithm 2) are used to initialize deep networks. The deep network is now ready to apply supervised machine learning such as SVM or a logistic regression. This can be done by adding a predictor to the last layer. In this work, we use a logistic regression solution as it can lead to better results in binary classifications [39]). To this end, a layer of logistic regression should be added, as shown in Fig. 8, to generate a

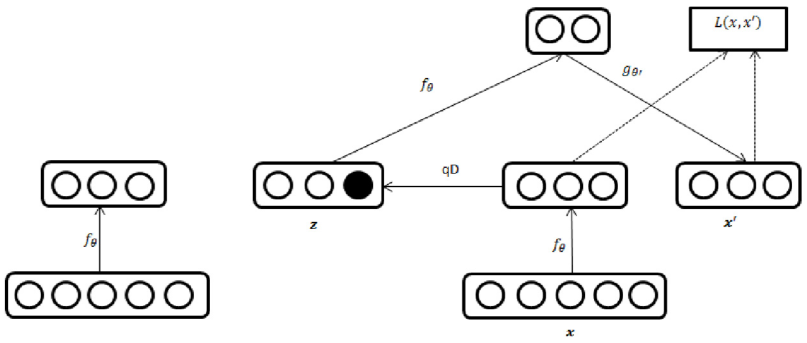


Fig. 6. Stacking denoising autoencoders. On the left, the encoding function f_θ, learnt in Fig. 2, is used on clean input x. On the right, The resulting representation is used to train a second block denoising autoencoder.

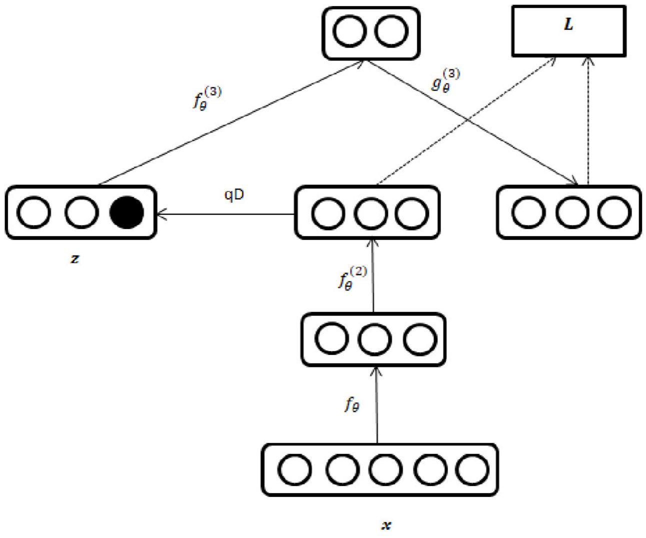


Fig. 7. The process is repeated for the third block denoising autoencoder.

Algorithm 2: IDS-based Unsupervised Pre-Training Algorithm

```
procedure PRE-RAINING($x$,$l$,$e$,$b$,$h$,$\Theta$)
    • $x$= $[x_1, x_2, ...x_n]$: Input IDSs' feedback
    • $h$=$[h_1, h_2, ...h_z] \in Z^l$
    • $\Theta$ = $[\theta_1, \theta_2, ...\theta_z]$,
    • Where $\theta_i$ = $\{W_i, b_i, b_{hi}\}$
    • $O$=$[O_1, O_2, ...O_l]$ is the output of each hidden layer, where $O_i$=$[o_{i,1}$,$o_{i,2}$, ... $o_{i,n}]$
    • $\theta_1$ = Training-DA ($x$, $l$, $e$, $b$, $\theta_1$)
for $i$ from 1 to $n$ do
        • $o_{1,i}$=$Sigmoid(x_i*W_1+b_i)$
    for $j$ from 2 to $l$ do
            • $\theta_j$ = Training-DA($O_{j-1}$,l,e,b,$\theta_j$)
        for $i$ from 0 to $n$ do
                • $o_{j,i}$=$\sigma(o_{j-1}*W_j+b_j)$
        end
    end
end
end procedure
```

deep neural network. The parameters of the all layers will then be fine-tuned to minimize the error to predict the supervised status (i.e., attack or not) using back-propagation algorithm. The fine-tuning algorithm is depicted in Algorithm 3.

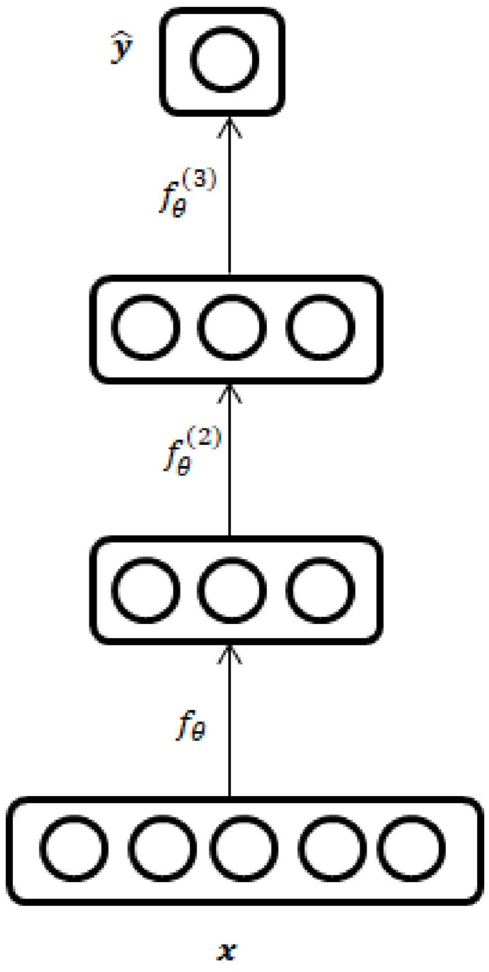


Fig. 8. The complete architecture of the proposed IDS-based deep neural network, once the last layer is added.

4. Experimental evaluation

This section first describes the setup used to perform the evaluation. Then, the performance of the proposed proactive multi-cloud cooperative intrusion detection system is examined.

4.1. Experimental setup

The proposed detection model was implemented using TensorFlow. The evaluation was performed using GPU-enabled TensorFlow running on a 64-bit Windows 8 with an Intel Xeon

Algorithm 3: IDS-based Fine Tuning Algorithm

```
procedure FineTuning($x$,$l$,$e$,$b$,$h$,$\Theta$)
    • $x$= $[x_1, x_2, ...x_n]$
    • $h$=$[h_1, h_2, ...h_z] \in Z^l$
    • $\Theta = [\theta_1, \theta_2, ...\theta_z]$, Where $\theta_i = \{W_i, b_i, b_{hi}\}$
    • $O$=$[O_1, O_2, ...O_l]$: output of each hidden layer, where $O_i$=$[o_{i,1}$,$o_{i,2}$, ... $o_{i,n}]$
    • $\theta_1$ = Training-DA ($x$, $l$, $e$, $b$, $\theta_1$)
    for epoch from 0 to e do
        • cost = $\frac{1}{|D|} = L(\theta = W, b, D)$
        • g = compute the gradient of cost w.r.t $\theta$
        for $\theta_i$,$g_i \in (\theta$,$g)$ do
            • $\theta_i = \theta_i - l * g_i$
        end
        if validationLoss < bestvalidationLoss then
            • bestEpoch=epoch
            • bestParameters=$\theta$
            • bestvalidationLoss=validationLoss
        end
    end
    return bestParameters
end procedure
```

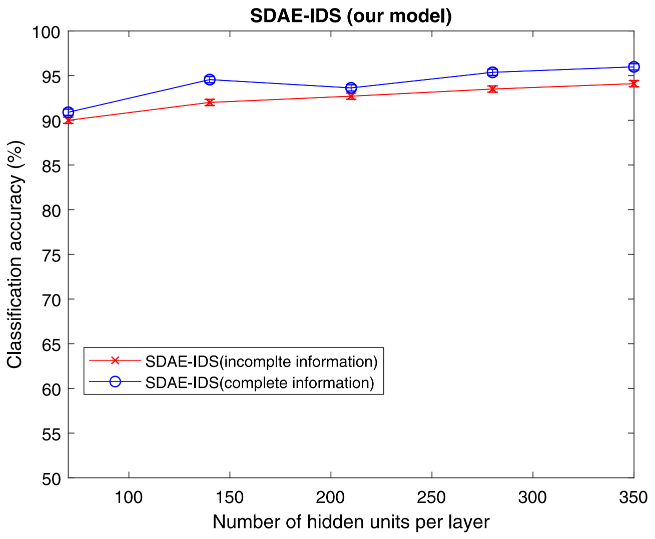


Fig. 9. Classification accuracy performance compare to having all the IDSs' feedback (complete information) — number of hidden layers = 3.

Table 1
Description of the dataset.

Category	Training data	Test data
DoS attacks	391458	223298
Probe attacks	4107	2377
R2L attacks	1126	5993
U2R attacks	52	39
Normal	97278	60593
Total	494021	292300

Table 2
Experimentation parameters.

Parameter	Considered values
Number of cloud-based IDSs N	70
Number of hidden layers h	{1, 2, 3}
Number of units per hidden layer	{70, 140, 210, 280, 350}
Corrupting noise level v	30%
Learning rate	0.05
Output layer	Binary logistic regression

3.60 GHz processor, 16 GB RAM. To evaluate this model, a dataset containing IDSs' feedback about suspicious intrusions was required. An IDS feedback about a given suspicious intrusion will be either 1 (if considered as an attack) or 0 (if not). This dataset was created based on the KDD Cup 99 dataset, where each 1 or 0 in the new dataset corresponds to the answer of an ID to a given row of the KDD Cup 99 dataset [40]. Recently, the KDD Cup '99 dataset has been used extensively in research about IDS [15–18] [19,20]. Table 1 shows the statistical summary of the dataset.

Once the dataset was created, it was used to train the model. Then, the ability of the proposed model in making decisions about suspicious intrusions was tested, even in the presence of partial/incomplete feedback. To represent partial/incomplete IDSs' feedback, some of the IDSs' feedbacks (selected randomly) were left blank. In this case, blanks indicate that some of IDSs' feedbacks have yet to be received, due to unexpected delays (busy IDSs, compromised IDSs, etc.).

The model (SDAE-IDS) was trained to create dataset using the 10-fold cross-validation model. Table 2 shows the parameters used for the experiment.

4.2. Experimental results

The accuracy of the proposed model was first examined and compared with having all the IDSs' feedback (complete information). This is important to evaluate the effectiveness of the proposed model in making decisions given partial or incomplete feedback. Fig. 9 shows that the average accuracy of the proposed model, with a variety of hidden units (ranging from 70 to 350), was slightly degraded (less than 1.5%). These results suggest that the proposed machine leaning-based approach can effectively make the right decisions about suspicious intrusions even in the absence of complete feedback from consulted IDSs.

Next, in Fig. 10, this model (i.e., SDAE-IDS) was compared with another approach, namely Stacked Auto Encoder-IDS (SAE-IDS). SAE-IDS uses traditional autoencoders (illustrated in Section 3.2) as a building block for the deep neural networks. As illustrated in Section 3, in our model (SDAE-IDS), denoising autoencoders are used as building blocks for the deep neural network. This figure illustrates the classification accuracy to make decisions regarding suspicious intrusions in the absence of complete feedback from the IDSs.

The study was conducted with different numbers of layers and hidden nodes. As shown in Fig. 10, our model yields increased accuracy compared to SAE-IDS. The study was done by considering three layers (the same number of layers is considered to study the deep neural network in [9]). More specifically, as for 1-hidden layer, Fig. 10(a) shows that the average accuracy obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) is 87.5%. This result is better than the results obtained using 1-hidden layer in SAE-IDS (72.50%) as shown in Fig. 10(b). Also, our model yields a improved accuracy compared to SAE-IDS using 2-hidden layers and 3-hidden layers. More particularly, Fig. 10(a) shows respectively that the 2-hidden layers' average accuracy and 3-hidden layers' average accuracy obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) are 91.5% and 92.5%. This result is better than the results obtained using SAE-IDS (75.50% for 2-hidden layers and 76.5% for 3-hidden layers) as shown in Fig. 10(b).

Clearly, for both approaches, SDAE-IDS and SAE-IDS, accuracy and the number of layers increase proportionally: this can be interpreted by the fact that each added layer allows for more

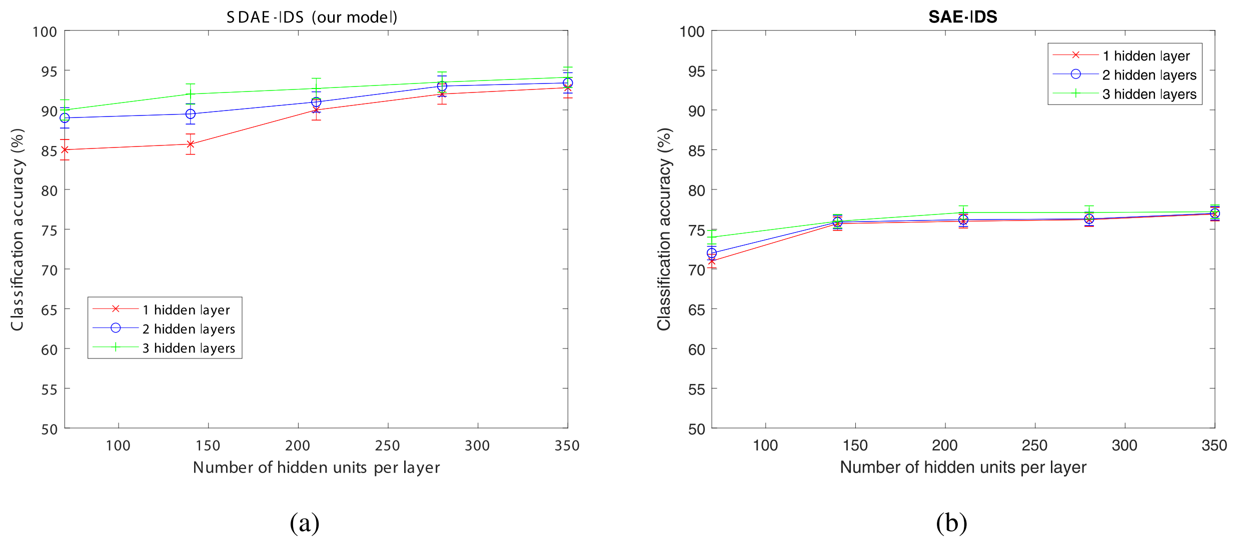


Fig. 10. Classification accuracy performance for SDAE-IDS (left) and SAE-IDS (right). Error bars show 95% confidence intervals.

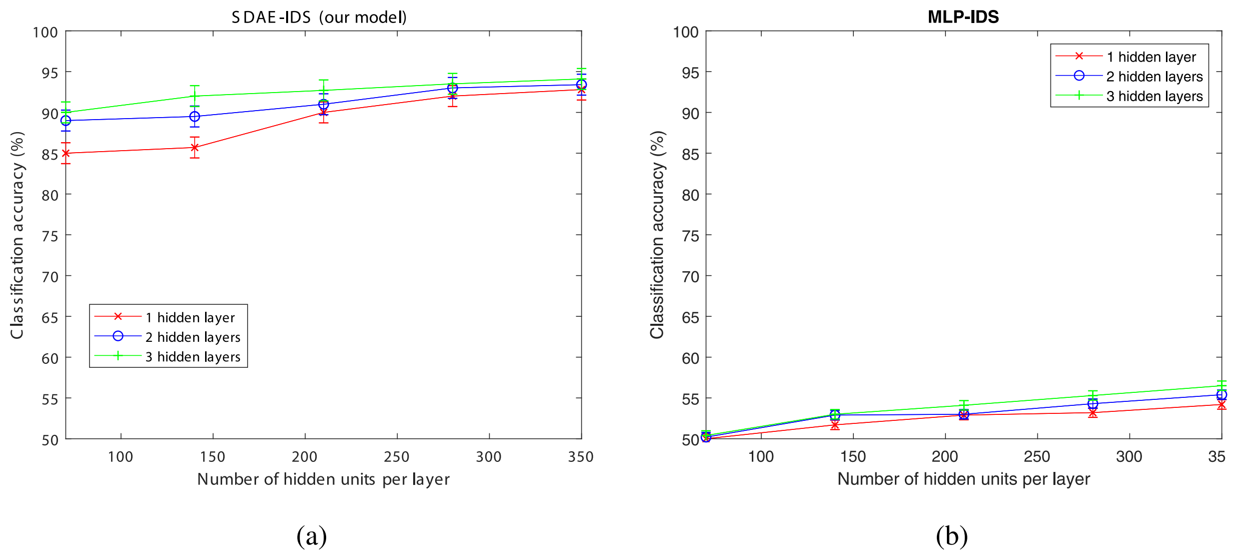


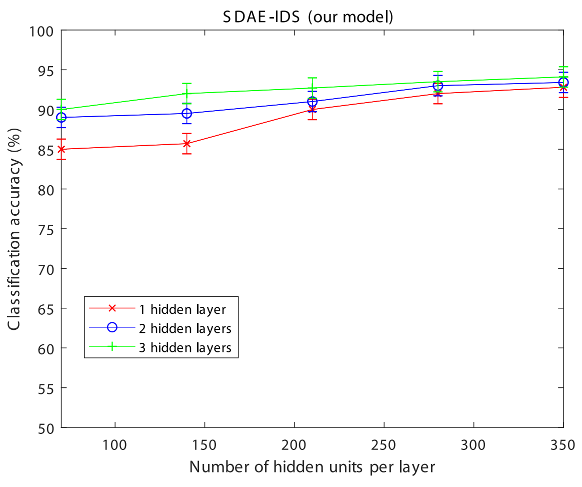
Fig. 11. Classification accuracy performance for SDAE-IDS (left) and MLP-IDS(right). Error bars show 95% confidence intervals.

representative data to be used during classification. Moreover, as the number of hidden units increases, the classification accuracy is enhanced. This is due to the fact that more hidden units allow more features to be captured from the data, hence enhancing the accuracy of the detection.

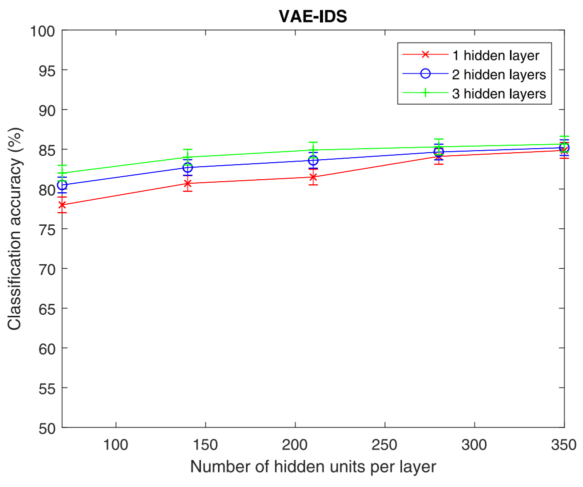
The reason why SDAE-IDS (our model) yields a better accuracy than SAE-IDS is that it uses denoising autoencoders as a building block for deep neural networks. Denoising autoencoders allow the deep neural network to extract robust features that lead to a better classification despite the incomplete feedback given as inputs to the deep neural network [10]. The denoising autoencoder learned how to reconstruct the feedback from corrupted input. This enables it to generate a "good" and useful representation despite corrupted input (missing feedback) thus enhancing the classification. This is not the case with SAE-IDS, as a basic autoencoder is used as a building block, which is unable to generate useful data from corrupted input.

Fig. 11 denotes the detection accuracy, comparing our model (i.e., SDAE-IDS) with multi-layers perceptron (MLP)-IDS. MLP-IDS uses a deep neural network without pre-training process. Furthermore, the study was conducted with different layers and hidden nodes. As for 1-hidden layer, Fig. 11(a) shows that the average accuracy obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) is 87.5%. This result is better than the results obtained using 1-hidden layer in MLP-IDS (52.20%) as shown in Fig. 11(b). Also our model yields improved accuracy compared to MLP-IDS using 2-hidden layers and 3-hidden layers. More particularly, Fig. 11(a) shows respectively that the 2- and 3-hidden layers' average accuracy obtained by the proposed model with different numbers of hidden units (ranging from 70 to 350) are 91.5%. and 92.5%. This result is superior to those obtained using MLP-IDS (53.20% for 2-hidden layers and 53.50% for 3-hidden layers) as shown in Fig. 11(b). The reason is that our model uses pre-training process which allows the deep network to have better initialization of parameters to be used then during backpropagation and fine tuning.

The proposed model (i.e., SDAE-IDS) was also compared with another approach, namely variation autoencoders-IDS (VAE-IDS). VAE-IDS uses a variation autoencoder as a building block to train deep neural networks [41]. The study was also conducted with different numbers of layers and hidden nodes. As shown in Fig. 12, our model yields increased accuracy compared to



(a)



(b)

Fig. 12. Classification accuracy performance for SDAE-IDS (left) and VAE-IDS (right). Error bars show 95% confidence intervals.

VAE-IDS. The study was done by considering three layers. More specifically, as for 1-hidden layer, Fig. 12(a) shows that the average accuracy obtained by the proposed model with different numbers of hidden units (ranging from 70 to 350) is 87.5%. This result is better than the results obtained using 1-hidden layer in VAE-IDS (72.50%) as shown in Fig. 12(b). Also, our model yields improved accuracy compared to VAE-IDS using 2-hidden layers and 3-hidden layers. Moreover, Fig. 12(a) shows respectively that the 2-hidden layers average accuracy, and 3-hidden layers average accuracy, obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350), are 91.5% and 92.5%. This result is better than the results obtained using VAE-IDS (75.50% for 2-hidden layers and 76.5% for 3-hidden layers), as shown in Fig. 12(b).

Figs. 13–15 denote the test classification error with different numbers of layers. As for 1-hidden layer, Fig. 13 shows that the average classification error obtained by the proposed model at different numbers of hidden units (ranging from 70 to 350) is 10.40%. This result is better than the results obtained using 1-hidden layer in SAE-IDS (24.80%) and MLP-IDS (46.50%). Also, our model yields enhanced accuracy compared to SAE-IDS and MLP-IDS using 2-hidden layers and 3-hidden layers. In particular, Figs. 14 and 15 respectively show that the 2- and 3-hidden layers' average accuracy obtained by the proposed model with different numbers of hidden units (ranging from 70 to 350) are 9.8%. and 7.20%. This result is superior to those obtained using SAE-IDS (24.70% for 2-hidden layers and 23.5% for 3-hidden layers) and MLP-IDS (46.10% for 2-hidden layers and 45.5% for 3-hidden layers). Moreover, Figs. 13, 14, and 15 indicate that the test classification error decreases as the number of hidden units increases. This is due to the fact that more hidden units allow for more features to be captured from the data, thus enhancing the accuracy of detection.

Figs. 16 and 17 compare SDAE-IDS (our model) with 3-hidden layers and two other denoising approaches [42] based on training with noisy input, namely SAE(1)-IDS and SAE(2)-IDS. SAE(1)-IDS is a 3-hidden-layers SAE-IDS where noisy inputs were only used for the pretraining [9]. SAE(2)-IDS is also 3-hidden-layers SAE-IDS where noisy inputs were used for both pretraining and fine-tuning [9]. These results demonstrate that our framework is also resilient to the increase in the percentage of noises. Figs. 16 and 17 respectively show that the accuracy and test classification error obtained by the proposed model at different percentages of corruption (from 0% to 40%) are 89.2% and 9.8%. These results are

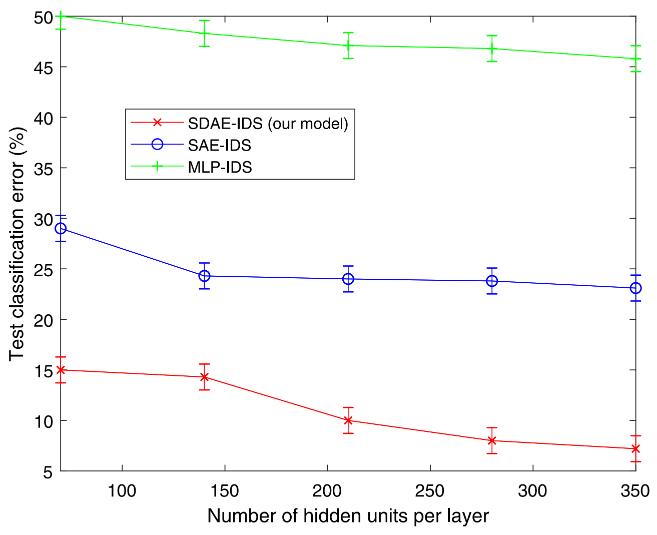


Fig. 13. Test classification error (%) — 1 hidden layer.

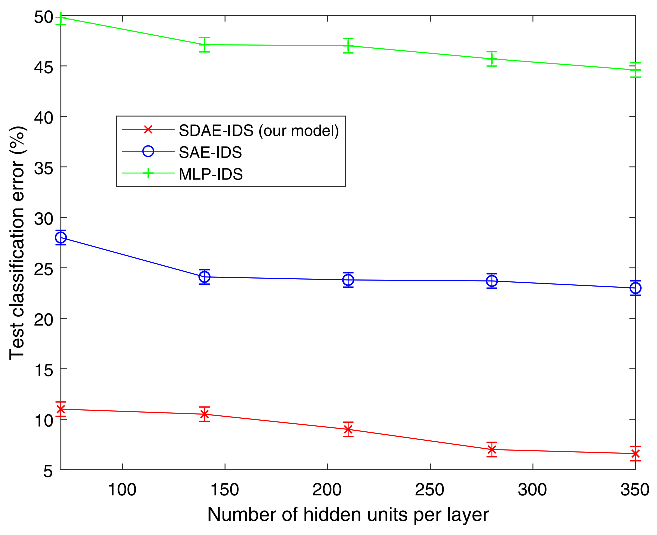


Fig. 14. Test classification error (%) — 2 hidden layers.

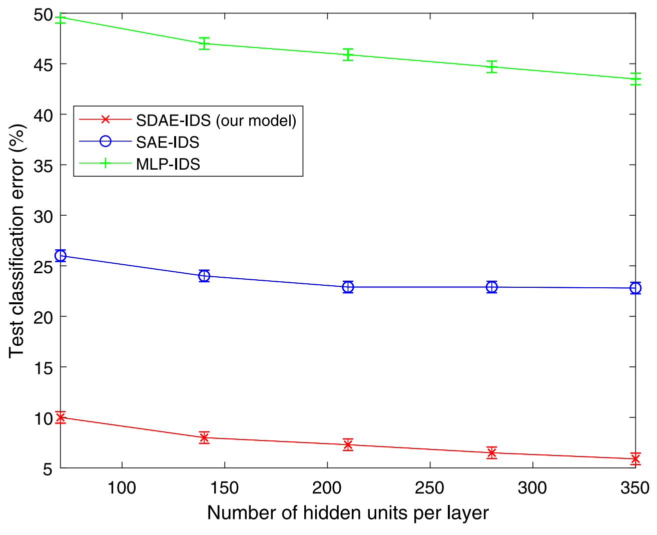


Fig. 15. Test classification error (%) — 3 hidden layers.

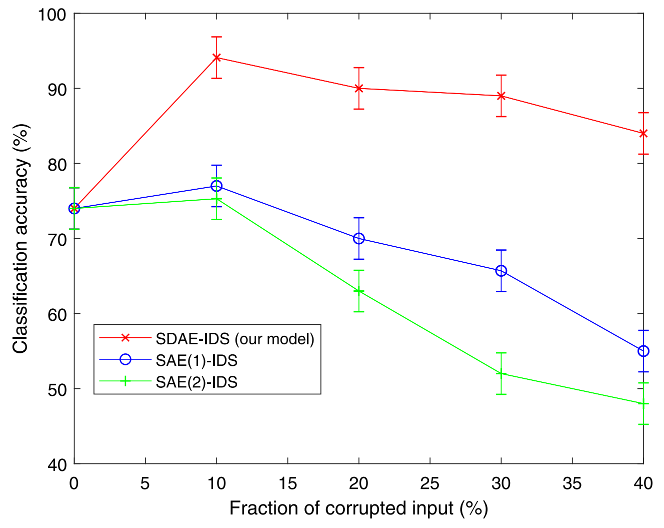


Fig. 16. SDAE-IDS vs. training with noisy input. Hidden layers have 350 units each.

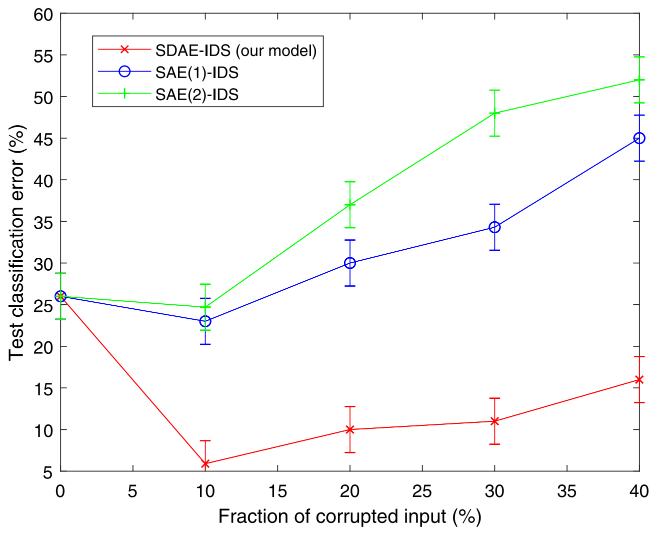


Fig. 17. SDAE-IDS vs. training with noisy input. Hidden layers have 350 units each.

better than those obtained using SAE(1)-IDS (69.7% for accuracy and 30.3% for classification accuracy) and SAE(2)-IDS (65.4% for the accuracy and 34.6% for test classification error). Note that when the percentage of corrupted inputs equals 0%, the three models (SAE(1)-IDS, SAE(2)-IDS and SDAE-IDS) yield the same results in terms of accuracy and error rate due to the fact that when 0% is applied, the three stacked models will be the same as SAE [9].

5. Conclusion

In this paper, we proposed a proactive multi-cloud cooperative IDS. The proposed model allows us to exploit the historical received feedback to produce learned models used for predicting the status (attack or not) of suspicious intrusions. The proposed model is based on stacked denoising autoencoders, where we use a denoising autoencoder as a building block for deep learning. The proposed IDS-based denoising autoencoder is used to learn how to reconstruct original IDSs' feedback given incomplete IDSs' feedback. This, in turn, allows us to learn how to extract features that are robust to incomplete feedback. Such robust features enables a useful representation of data, enabling a better performing cooperative intrusion detection. The proposed model allows making decisions regarding suspicious intrusions despite the existence of partial or incomplete IDSs' feedback. Also, the proposed model can make decisions regarding suspicious intrusions without having to apply an aggregation method on the consulted IDSs' feedback. Our model was implemented in GPU-enabled Tensor-Flow and evaluated using a real-life dataset. The results show the efficiency of the proposed approach in terms of enhancing the cooperative intrusion detection accuracy compared with the existing state-of-the-art deep architectures.

Acknowledgment

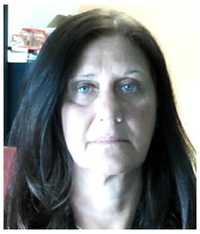
The financial support of the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged.

References

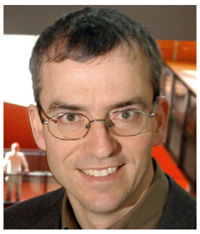
[1] Á. Dermott, Q. Shi, K. Kifayat, Collaborative intrusion detection in federated cloud environments, J. Comput. Sci. Appl. 3 (3A) (2015) 10–20.
[2] C.J. Fung, Q. Zhu, Facid: A trust-based collaborative decision framework for intrusion detection networks, Ad Hoc Netw. 53 (2016) 17–31.
[3] A. Abusitta, M. Bellaiche, M. Dagenais, A trust-based game theoretical model for cooperative intrusion detection in multi-cloud environments, in: 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), IEEE, 2018, pp. 1–8.
[4] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.
[5] G.E. Hinton, Learning multiple layers of representation, Trends Cogn. Sci. 11 (10) (2007) 428–434.
[6] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[7] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[8] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in Neural Information Processing Systems, 2007, pp. 153–160.
[9] 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, J. Mach. Learn. Res. 11 (Dec) (2010) 3371–3408.
[10] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 1096–1103.

[11] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, M. Rajarajan, A survey of intrusion detection techniques in cloud, J. Netw. Comput. Appl. 36 (1) (2013) 42–57.
[12] C.J. Fung, D.Y. Lam, R. Boutaba, Revmatch: An efficient and robust decision model for collaborative malware detection, in: Network Operations and Management Symposium, NOMS, 2014 IEEE, IEEE, 2014, pp. 1–9.
[13] C.-C. Lo, C.-C. Huang, J. Ku, A cooperative intrusion detection system framework for cloud computing networks, in: Parallel Processing Workshops, ICPPW, 2010 39th International Conference on, IEEE, 2010, pp. 280–284.
[14] S. Teng, C. Zheng, H. Zhu, D. Liu, W. Zhang, A cooperative intrusion detection model for cloud computing networks, Int. J. Secur. Appl. 8 (3) (2014) 107–118.
[15] N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Trans. Emerg. Top. Comput. Intell. 2 (1) (2018) 41–50.
[16] K. Alrawashdeh, C. Purdy, Toward an online anomaly intrusion detection system based on deep learning, in: Machine Learning and Applications, ICMLA, 2016 15th IEEE International Conference on, IEEE, 2016, pp. 195–200.
[17] J. Kim, N. Shin, S.Y. Jo, S.H. Kim, Method of intrusion detection using deep neural network, in: Big Data and Smart Computing, BigComp, 2017 IEEE International Conference on, IEEE, 2017, pp. 313–316.
[18] A. Javaid, Q. Niyaz, W. Sun, M. Alam, A deep learning approach for network intrusion detection system, in: Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies (Formerly BIONETICS), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26.
[19] S. Potluri, C. Diedrich, Accelerated deep neural networks for enhanced intrusion detection system, in: Emerging Technologies and Factory Automation, ETFA, 2016 IEEE 21st International Conference on, IEEE, 2016, pp. 1–8.
[20] C.G. Cordero, S. Hauke, M. Mühlhäuser, M. Fischer, Analyzing flow-based anomaly intrusion detection using replicator neural networks, in: Privacy, Security and Trust, PST, 2016 14th Annual Conference on, IEEE, 2016, pp. 317–324.
[21] T.A. Tang, L. Mhamdi, D. McLernon, S.A.R. Zaidi, M. Ghogho, Deep learning approach for network intrusion detection in software defined networking, in: Wireless Networks and Mobile Communications, WINCOM, 2016 International Conference on, IEEE, 2016, pp. 258–263.
[22] W. Jia, K. Muhammad, S.-H. Wang, Y.-D. Zhang, Five-category classification of pathological brain images based on deep stacked sparse autoencoder, Multimedia Tools Appl. (2017) 1–20.
[23] Y.-D. Zhang, Y. Zhang, X.-X. Hou, H. Chen, S.-H. Wang, Seven-layer deep neural network based on sparse autoencoder for voxelwise detection of cerebral microbleed, Multimedia Tools Appl. (2017) 1–18.
[24] N.D. Man, E.-N. Huh, A collaborative intrusion detection system framework for cloud computing, in: Proceedings of the International Conference on IT Convergence and Security 2011, Springer, 2012, pp. 91–109.
[25] D. Singh, D. Patel, B. Borisaniya, C. Modi, Collaborative ids framework for cloud, Int. J. Netw. Secur. 18 (4) (2016) 699–709.
[26] S. Ghribi, Distributed and cooperative intrusion detection in cloud networks, in: Proceedings of the Doctoral Symposium of the 17th International Middleware Conference, ACM, 2016, p. 7.
[27] Z. Chiba, N. Abghour, K. Moussaid, M. Rida et al, A cooperative and hybrid network intrusion detection framework in cloud computing based on snort and optimized back propagation neural network, Procedia Comput. Sci. 83 (2016) 1200–1206.
[28] Q. Zhu, C. Fung, R. Boutaba, T. Basar, A game-theoretical approach to incentive design in collaborative intrusion detection networks, in: Game Theory for Networks, 2009 GameNets' 09 International Conference on, IEEE, 2009, pp. 384–392.
[29] Q. Zhu, C. Fung, R. Boutaba, T. Basar, Guidex: A game-theoretic incentive-based mechanism for intrusion detection networks, IEEE J. Sel. Areas Commun. 30 (11) (2012) 2220–2230.
[30] G. Shafer, Dempster-shafer theory, Encyclopedia Artif. Intell. (1992) 330–331.
[31] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, D.-R. Liou, Autoencoder for words, Neurocomputing 139 (2014) 84–96.
[32] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, vol. 1, MIT press, Cambridge, 2016.
[33] A.J. Bell, T.J. Sejnowski, An information-maximization approach to blind separation and blind deconvolution, Neural Comput. 7 (6) (1995) 1129–1159.
[34] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (8) (2002) 1771–1800.
[35] S. Haykin, N. Network, A comprehensive foundation, Neural Netw. 2 (2004) (2004) 41.
[36] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.
[37] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[38] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, Y. Bengio, An empirical evaluation of deep architectures on problems with many factors of variation, in: Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 473–480.
[39] S. Dreiseitl, L. Ohno-Machado, Logistic regression and artificial neural network classification models: a methodology review, J. Biomed. Inform. 35 (5–6) (2002) 352–359.
[40] Kdd cup 1999 data, 2018. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. (Accessed 10 May 2018).
[41] M.E. Abbasnejad, A. Dick, A. van den Hengel, Infinite variational autoencoder for semi-supervised learning, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, 2017, pp. 781–790.
[42] A. Zee, Emergence of grandmother memory in feed forward networks: Learning with noise and forgetfulness, in: Connectionist Models and their Implications: Readings from Cognitive Science, 1988, p. 309.

Adel Abusitta is a Ph.D. student in Computer Engineering at Ecole Polytechnique de Montreal, Canada. He holds a M.Sc. degree in computer science from the University of Jordan, Jordan. The main topics of his current research activities are security in Cloud Computing, network security, and authorship analysis.



Martine Bellaiche is Assistant Professor in Department of Computer and Software engineering of Ecole Polytechnique of Montreal. She received a M.Sc. in Computer Science from University of Montreal in 1985 and the Ph.D. in Telecommunications from INRS in 2007. Her research interests are in Network Security with special focus on attack, in Network Sensor Security and cloud computing security.



Michel Dagenais is professor at Ecole Polytechnique de Montreal in the department of Computer and Software Engineering. He authored or co-authored over one hundred scientific publications, as well as numerous free documents and free software packages in the fields of operating systems, distributed systems and multicore systems, in particular in the area of tracing and monitoring Linux systems for performance analysis. Most of his research projects are in collaboration with industry and generate free software tools among the outcomes. The Linux Trace Toolkit next generation, developed under his supervision, is now used throughout the world and is part of several specialized and general purpose Linux distributions.

Talal Halabi is a Ph.D. candidate in Computer Engineering at Ecole Polytechnique of Montreal, Canada. He holds a Master of Research (MRes) degree in telecommunication and network engineering from the Lebanese University and Saint-Joseph University in Beirut. His current research activities are mainly focused on security quantification and evaluation in Cloud Computing, security satisfaction of Cloud consumers, Cloud Security-SLA development and standardization, optimal security-based Cloud resources allocation, and security of Cloud federations.