



REVUE ALGERIENNE DES SCIENCES

Section -A-

Sciences de la Nature et de la Vie

Sciences techniques

<http://univ-eltarf.dz/fr/>



Detecting IoT Botnet Attacks Using Deep Learning Approach

Labiodyasmine^a, Abdelaziz Amara Korba^a, NaciraGhoualmi^{a,*}

^aComputer science department, BadjiMokhtarUniversity - Annaba, Algeria.

Informations	Abstract
Keywords: Deep learning, IoT, Autoencoders network security, intrusion detection systems * Correspondence: yasminelabiod769@gmail.com Received, Revised, Accepted.	Recently, Network Intrusion Detection (NIDS) has emerged as an essential part of cyber security, with a focus on protection Internet of Things (IoT) systems from unauthorized use and attacks. Nevertheless, NIDS techniques are facing several challenges due to the emergence of new numerous sophisticated attacks. Using Deep Learning approaches, the main objective of this research is to automatically learn useful feature representations, extracted from large amounts of unlabeled raw network traffic data. Therefore, we propose an effective Network Intrusion Detection System STL-NIDS based on the self-taught learning (STL) framework. The proposed model is build using the deep AutoEncoder neural network, which is an effective learning technique for reconstructing a new feature representation in an unsupervised manner. After the pre-training stage, the new features are fed into the SVM algorithm to improve its detection capability for intrusion and classification accuracy. The proposed method is validated experimentally and a comparison between competitive methods using IoT Bot dataset is conducted. Hence, results of the evaluation using several appropriate metrics show that the proposed method outperforms the related works' results. Moreover, it is proven that our model is effective for large-scale and real-world network environments.

1. Introduction

The incredible growth of insecure Internet of Things (IoT) devices with high computation power has led to a rapid development of highly sophisticated and targeted malicious activities and attacks [1]. A botnet is one of the most dangerous malware in IoT environment. The botnet is a robot network of infected machines (compromised machine) or bots, also called zombies, that has a command-and-control infrastructure which is used for various malicious activities such as: email spam delivery, password cracking, key logging, crypto currency mining, and distributed denial-of-service (DDoS) attacks [2]. Bots can automatically scan entire network ranges and propagate themselves using weak passwords and known vulnerabilities on other compromised machines. Once a machine is infected, a small program is installed for future activation by the botmaster. This program at a certain time can instruct the bots in the network to execute actions such as sending requests to a target website with the intent of rendering it unable to serve requests by legitimate users, brute force and resulting in DDoS [3]. Thenceforth, a secured IoT network is necessary and requires immediate attention for the protection from different

types of attacks. Systems such as Anti-Virus and Firewall have been developed to establish various security solutions and protect the IoT network. However, weaknesses caused by anti-viruses architectures and Firewall are not updated and new (unknown or unseen) attacks or botnet can be created. For this reason, significant research has been focused on developing Intrusion Detection Systems (IDSs) to improve software and system security in IoT [4]. Thus, Intrusion Detection Systems (IDSs) play a vital role in the IoT network security. The IDS approaches can be categorized into two main classes: Host-based Intrusion Detection Systems (HIDSs) and Network Intrusion Detection Systems (NIDSs). Where the HIDSs detect intrusions in an individual host. While, the NIDSs detect attacks by observing various network activities, and help determine and identify unauthorized usage, duplication, alteration as well as any destruction of data. Depending on the detection techniques, NIDSs techniques can be examined in two separate categories: misuse-based detection and anomaly-based detection. Misuse-based detection discovers attacks based on the patterns extracted from known intrusions. While anomaly-based detection are designed to identify and capture attacks

based on the deviations from the established profiles of normal behavior.

Behaviors that exceed thresholds of the deviations are detected as attacks. The misuse detection techniques has low false positive rate but cannot detect new types of attacks. Therefore, the NIDSs based anomaly detections are more suitable than misuse detection systems. Anomaly based detection are efficient for detecting unknown or novel types of attacks without any prior knowledge and under a basic assumption that attacks deviate from normal activities [5]. For this reason, we focus in our study on NIDS based anomaly detection. The NIDSs based anomaly detection has a relationship with different attacks threaten IoT network security, such as: DDoS such as SYN Flood and Scan Flood, Information Theft, and Probing attacks. In addition, the NIDS has been successfully applied to various fields such as: computer network security, automatic speech recognition, credit card fraud detection, natural language processing, environmental disaster analysis, military applications, health care systems [6], and cyber-physical systems [7].

In recent years, the NIDSs based anomaly can be trained used machine learning models to separate abnormal and normal features. The schemes are able to detect patterns of known and unknown attacks in unsupervised, supervised, [8] or semi-supervised manner [9]. In supervised training, the detection model is trained with using both anomalous and normal features. The goal of this type of training is to label the test features as normal or anomalous by using the trained model. There are various studies which applied supervised techniques such as Bayesian networks, Random Forest, artificial neural networks (ANN), support vector machines (SVM), and Random Forest (RF) for anomaly detection [10]. Compared to misuse based approaches, the supervised techniques perform better performance in anomaly detection, but these methods are not very successful on the detection of zero-day attacks [11]. On the other hand, semi-supervised or unsupervised techniques are more suitable for network intrusion detection issues. In semi-supervised training, the model is obtained by training it with normal features only. Then this model is used to classify features which do not fit the model as anomalous [12]. In addition to afore mentioned techniques, in unsupervised training, the model is trained using completely unlabeled features. Thereafter, in the test stage, the trained model tries to cluster the normal features and anomalous into separate clusters [13].

Deep learning based unsupervised techniques have gained interest recently as they can efficiently learn a model and improve the NIDS performance. With deep

learning approaches, we can expect to tackle issues in anomaly-based network intrusion detection, such as, ability to adapt to dynamic network environments, capabilities to extract automated complex data representation at high levels of abstraction, unavailability of labeled features and high intrusion detection rate. The most popular deep learning based unsupervised techniques include, convolutional neural network [14], long short term memory recurrent neural network [15], deep belief networks with restricted boltzmann machine and autoencoders (AEs) neural network [16] have been used for network intrusion detection system. Hence, the AEs as the automatic feature learning models can provide more discriminative features in contrast to other feature engineering approaches and they achieve a promising performance in contextual categorization by reducing the testing and training times. On the basis of a self-taught learning framework and inspired by the success of deep autoencoder based unsupervised techniques in number of challenging classification issues [17] [18], we propose a deep learning based Intrusion Detection System STL-NIDS (a self-taught learning based intrusion detection system). Our approach consists of three autoencoders for good data representation and SVM for the classification task. The output of first deep autoencoder in the current layer is used as the input of the second autoencoder in the next layer. In addition, training an autoencoder is started when training the previous one is completed. Then, the trained parameters of the second autoencoder will be saved to train the last autoencoder. After that, the output features have passed to the support vector machines (SVM) classifiers. The SVM classifies the attack classes from the input dataset. Since, a manual search optimization method was studied to find suitable hidden layer number, weight decay parameter μ , and neuron number in each layer.

The proposed models were trained in an unsupervised manner using the up-to-date NIDS benchmarking dataset Bot IoT-dataset [19]. The Bot-IoT dataset is a common benchmark for network intrusion detection consisting of real network data in IoT. In addition we compared the obtained results with the results of the most successful previous studies which applied basic single SVM for the classification task. We also compared our model to random forest and Naïve Bayes algorithms with a similar evaluation techniques [20], [21], and [22].

In summary, the major contributions of this paper are the following:

1) We develop a novel deep learning approach STL-NIDS based on the STL framework by combining deep autoencoder and SVM for network intrusion detection.

We study the potential of our approach to achieve effective representation and dimensionality reduction for the improvement of the classification results of traditional supervised machine learning techniques.

2) Our model is pre-trained using an unsupervised learning algorithm to avoid overfitting and decrease the amount of training and testing times of SVM.

3) The performance of the proposed DAE is evaluated using Bot-IoT dataset and compared with single SVM and other classification algorithm on binary classification. Moreover, a series of preliminary experiments is conducted to explore the performance of DAE based on different number of hidden layers and units. To the best of our knowledge, we are the first to apply autoencoders to IoT bot dataset for network intrusion detection.

The rest of this paper is organized as follows. Section 2 briefly describes some previous relevant works on attacks intrusion detection using machine learning models and deep learning approaches. Section 3 describes our proposed methodology for NIDS implementation. The experimental results are demonstrated in Section 5 and the conclusions follow in Section 6.

2. Related works

Several studies have been conducted to develop tools for the detection of different attacks types in the IoT infrastructure. The network intrusion detection has become the most important part in IoT security. Various machine learning algorithms and approaches are applied in IDS to distinguish between normal traffic and attacks or anomalies. These approaches include naïve Bayes network [23], k-nearest neighbor (K-NN) [24], decision tree [25], SOM [26], artificial neural network (ANN) [27], and SVM [28]. In [27] authors compared the performance of ANN using a set of classifiers such as: ZeroR, OneR, JRip, RF, PART, J48 and SVM on IoT botnet dataset. The results showed that the detection results of ANN are better than those of RF. Authors in [24] tested several classifiers for IoT botnet attacks detection, namely decision tree (DT), probabilistic neural network (PNN), k-nearest neighbor (kNN), support vector machine (SVM) and discriminant classifiers; the PNN classifier gave a better accuracy of 98.8%. In [28] [29], authors employed an efficient machine learning techniques based on ANN and SVM classifiers with feature augmentation to classify the input network features into two specific class: normal and abnormal traffic. They used in their approach both ISOT [30] and ISCX [31] which are the two well-known datasets in the literature of botnet detection. The proposed approach

increases the quality of intrusion detection rate of the ANN classifier, and reduces the required training time. However, the weakness of this approach is the insufficiency of its detection accuracy. In addition, the time factor was not considered. In [32], authors presented a new method for the classification of network traffic in IoT applications. The proposed methodology is based on the insertion of new neurons into the hidden layer. The classification rate is 93%. In [33], researchers proposed an automatic system for the detection of DDoS attacks in IoT based on the N-Bot dataset by merging the results of the fuzzy and genetic algorithms. In [34] researchers proposed a bot host detection system (BHDS) for detecting bot infected hosts and their malicious network traffics in IoT environment. The proposed BHDS relies on the profile of each host including their behavior during a period of time. A machine learning approach was used by combining the SVM classifier with neural network machine learning classification techniques and random weights which they plays an important role in the detection rate of BHDS because it decreases the computational cost. The model was evaluated on the IoT botnet dataset but the performance of the model was studied on the binary classification task only. In [35][36], many unsupervised learning algorithms (clustering) like K-means and BIRCH (balanced iterative reducing and clustering using hierarchies) were combined with SVM and a neural network (NN). The combination aims to enhance the intrusion detection system performances and reduce the time training. Recently, in [37] unsupervised learning techniques has been used recently with SVM to detect DDoS attacks in IoT networks with non-stationary traffic. Consequently, this leads to an increase in the detection rate and a decrease in the processing time SVM machine learning algorithm. In [38], authors presented a network attack detection system using a large-scale clusters with SVM. The purpose of this proposition is offering the power to process high-dimensional network traffic data detection and to improve the detection rate. The KDD CUP 99 data was used as train and test dataset to evaluate the performance of the proposed approach. The performance of their model was compared against shallow and traditional machine learning approaches.

Approaches mentioned above need a prior artistic step of extracting the characteristics of the network traffic before the main recognition step. Hence, they are not applicable in real time that auto encoder AE. According to our knowledge, this is the first work using deep autoencoder neural networks for detecting DDoS attacks in IoT using the Bot IoT dataset.

3. Proposed Methodology: STL-NIDS:

In this section, we describe how the network intrusion detection problem is addressed through our Self-taught Learning. The Self-taught Learning (STL) is a deep learning techniques that consists of two stages for the classification [39]. The first stage, a good feature representation is learnt from a large collection of unlabeled data, x , termed as Unsupervised Feature Learning (UFL). In the second stage this learnt representation is applied to labeled data, X_L , and used for the classification task. Although the unlabeled and labeled data may come from different distributions, there must be relevance among them. The data in the first stage is obtained from IoT bot dataset without label, x_u . However, the data in the second stage is combined with labeled data, x_l which can be described as follow : we are given a labeled training set of m records $\{(x_l^{(1)}, y^{(1)}), (x_l^{(2)}, y^{(2)}), \dots, (x_l^{(m)}, y^{(m)})\}$ where input feature vector $x_l^{(i)} \in \mathbb{R}^n$, (The subscript "l" indicates that it is a labelled record), $y^{(i)} \in \{0,1\}$ are the corresponding labels for binary classification. There are different approaches used for UFL, such as Sparse Autoencoder, Restricted Boltzmann Machine (RBM) [40], Gaussian Mixtures and K-Means Clustering [41]. For our model, we used a deep autoencoder based feature learning due to its relatively easier implementation and good performance [42]. Autoencoder neural network is an unsupervised learning algorithm that applies back propagation to train, it tries to copy its input to its output. It can be used for dimensionality reduction and features learning instead of PCA to achieve a significantly nonlinear generalization. Its output and input layer have the same number of neurons. As shown in Figure 2(a), the output and input layers contain K units, and the hidden layer contains N units. The input values x_i in the input layer is similar to the output values $\hat{x}_i$ in the output layer. Internally, dimensionality reduction and Feature extraction process in deep autoencoder involves two steps: encoding and decoding. The encoding function $h = f(x)$ and decoding function $r = g(x)$ produce a reconstruction of x .

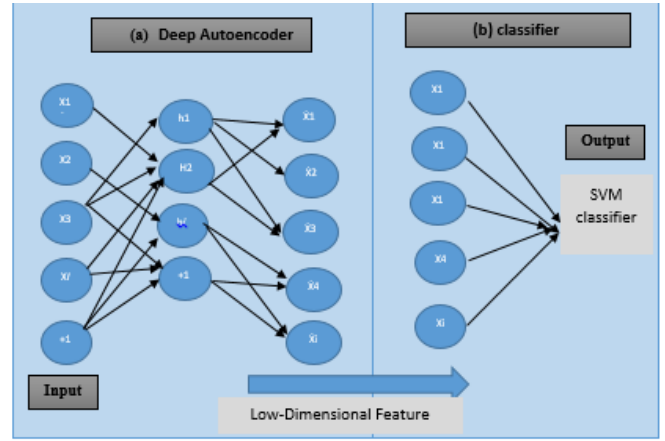


Figure 1: The Self-taught learning stages.

In the autoencoder, we expect the output sample $\hat{x}$ is the same as the input sample x . The encoder encodes the input feature x into a new expression and it is used to find a latent layer h that can represent the input feature. Then it transforms an input $x \in \mathbb{R}^n$ into a hidden layer representation $h \in \mathbb{R}^{n'}$ with the deterministic mapping:

$$h = f(x) = s(Wx + b) \quad (1)$$

Where, $x = (x_1, x_2, x_3, x_4, \dots, x_i)$ is the high-dimensional input data vector, b is the bias value of the hidden layer, $h = (h_1, h_2, h_3, \dots, h_k)$ is the low-dimensional vector output from the hidden layer, w is the weight matrix between input layer and hidden layer ($d \times d'$) and s is the activation function of hidden layer.

The decoder layer is the opposite of the encoder layer, it decodes the input (latent vectors) produced by encoder into a reconstruction of the original input (output). A transformation g maps back the resulting hidden representation h to a d -dimensional reconstruction of x denoted by r :

$$r = g(x) = s(W'h + b') \quad (2)$$

Where $r = (\hat{x}_1, \hat{x}_2, \hat{x}_3, \dots, \hat{x}_m)$ is the reconstruction vector of the input data, w is a $d \times d$ weight matrix, s is the average activation value of hidden layers over all training inputs, and b is the corresponding bias vector.

The back propagation algorithm is used our deep autoencoder to obtain the optimal values for its bias vectors $b \in \mathbb{R}^{K \times 1}$ and $b' \in \mathbb{R}^{N \times 1}$ and weight matrices $W \in \mathbb{R}^{K \times N}$ and $W' \in \mathbb{R}^{N \times K}$ witch attempt to reconstruct and learn its input value $\hat{x}_i$ to be equal to its output values $\hat{x}_i$. In other hands, a nonlinear activation functions is learned to make the output values similar to the input values; such as softmax, sigmoid, tanh and rectified linear functions [43]. Hence the deep autoencoder can extract more useful features when it uses an optimal

non-linear activation function. In our model, the activation function is chosen to be the sigmoid function, and its output range is [0,1]. This function is used for the activation (W) of the nodes in the output and hidden layers based on a series of preliminary experiments.

$$\text{Sigmoid function: } S(x) = \frac{1}{1+e^{-x}} \quad (3)$$

To improve the performance of our STL-NIDS, our deep autoencoder also applies back propagation to minimize the loss function (reconstruction error) or the difference between the input x and reconstruction r (output) [44]. The first term is the average sum-of-square errors for all m input data. Therefore, the optimal parameters of the decoder and encoder networks can be trained together to minimize the objective function:

$$(\theta^*, \Phi^*) = \operatorname{argmin}_{\theta, \Phi} \frac{1}{N} \sum_{i=1}^N L(x_i, r_i)$$

$$(\theta^*, \Phi^*) = \operatorname{argmin}_{\theta, \Phi} \frac{1}{N} \sum_{i=1}^N L(x_i, g(f(x_i)))$$

$$(\theta^*, \Phi^*) = \operatorname{argmin}_{\theta, \Phi}$$

$$\frac{1}{N} \sum_{i=1}^N L(x_i, \mu(s(\sum(W'h + b'))(s(\sum(Wx + b))))(4)$$

Where L represents the loss function and n represents the number of training features. According to the distribution assumptions on the input, the loss function can take many forms. The conventional one is mean squared error (RMSE):

$$L(x, r) = ||x - r||^2 \quad (5)$$

If the interpretation of r and x is either vector of number probabilities or number vector, the cross-entropy can be an alternative:

$$L(x, r) = -\sum_{k=1}^d [x_k \log r_k + (1-x_k) \log (1-r_k)] \quad (6)$$

The second term is a weight decay parameter (μ) used for tuning the weights between the output and hidden layers to improve performance and classification while helping avoid and check over fitting.

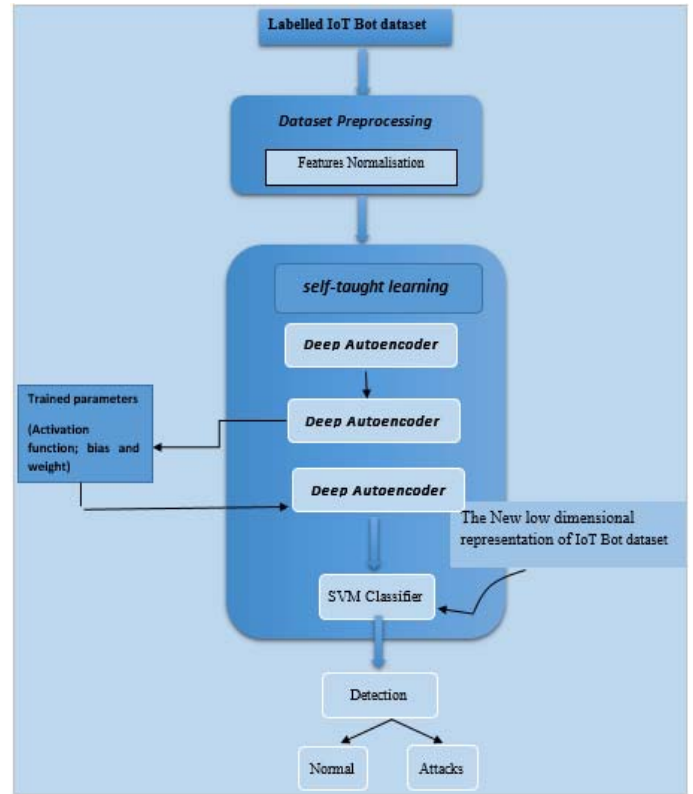


Figure 2 : Block diagram of the proposed STL-NIDS

3.1 Training Process:

Our model is performed after several performance tests. We start through the encoder layer creation, a batchnormalization step is applied after each hidden layer to decrease the number of feature maps. A stochastic gradient descent is used with a momentum value of 0.9. L2 regularization method is also applied for weight and biases with a threshold equal to 0.000001 and 0.002 respectively. Finally, a low learning rate is fixed at 0.0001 to train our deep autoencoder neural network. We use two layers of hidden layer and one latent layer using each one the sigmoid activation function. Stride parameters are fixed at 8 and 2 for hidden layer and latent layers respectively. For the dense layer, we use the widely used sigmoid function. The table 1 summarized the above Multi-layer parameters.

Model type	Multi-layer Parameters
Layers	5
Total parameters	1000001
Hiddenlayers	3
Epoche	1000

Batchsize	300
Learning rate	[0.05-0.1]
Weightdecayparameter μ	0.000001
bias	0.002
Latent vector dimension	2
Activation function	Sigmoid

Tab 1. The autoencoder Model and training information

After learning the optimal values for weight W and bias b , batch size and epoch by applying deep autoencoder on unlabeled data xu , we evaluate the feature representation $a = h$ for labeled data (xl, y) . We used this new feature representation, h , with the label vector, y in SVM for the classification task. For our study, we apply the STL based deep autoencoder for good data representation because of its unpretentious and simple implementation and its capability to learn the original expressions and structures of data. In addition, the wide range of STL applications has been successfully applied to various fields such as: image identification and SVM for classification tasks and detection different types of intrusion because combining efficacy and solid classifiers, such as SVM, with deep autoencoder leads to improved performance in intrusion detection. Furthermore, the features extracted from the DAE algorithm are passed to the SVM classifier for intrusion detection. Figure 2 shows an architectural diagram of the proposed STL. The SVM classifier is employed at the second stage in our STL-NIDS to determine the class of network traffic features and distinguish between the normal and abnormal network traffic. It takes the output of the last autoencoder as input and then tries to classify the features. The performance accuracy rate of our method is better than that of SVM alone, and the training and testing times of SVM are reduced.

3.2 SVM:

Support vector machine (SVM) is the most binary classifier used in STL classification process. It has gained popularity in machine learning community. SVM works in combination with a RBF kernel (Gaussian kernel) that automatically realizes a non-linear mapping to a feature space. SVM was originally developed for offering good generalization performance to new data and solving the optimization problem in classification fields. The SVM process was divided into two phases, namely the optimal phase and the decision phase. In the first phase, having a

set of labeled vectors $L = \{x_i, y_i\} | 1 < i < l$ where $x_i \in R^n$, and $y_i \in \{-1, +1\}$ represents two distinct categories. The SVM constructs a hyper plane or a set of hyper planes in a space of high or infinite dimension; it separates a given set of binary labeled training data with a hyper-plane that is maximally distant from these binary classes (known as the maximal margin hyper-plane). Finding h requires minimizing $\|w\|^2$, where w is a vector normal to h .

$$\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j, y_i y_j, x_i x_j, w = \sum_{i=1}^n \alpha_i y_i x_i \quad \text{With: } \alpha_i > 0 \quad \text{and} \quad \sum_{j=1}^n \alpha_j y_j = 0 \quad (7)$$

where α_i and α_j are lagrange multipliers. After that, in the second phase and based on the hyper-plane, a classification of the vector was done according to their position in the n -dimensional space. The decision function was based on the following relation:

$$F(x) = \min_i y_i \{ \langle w, \varphi(x^2) \rangle - b \} \quad (8)$$

The margin $f(x_i)$ is maximized, when the hyperplane (w, b) input the data x_i with the corresponding label y_i . The quantity $(w, \varphi(x_i) - b)$ corresponds to the distance between x_i and the decision boundary. While the SVM consumes much time for training and testing, numerous studies are addressed in SVM to reduce the required processing time for classification tasks. The computational complexity and storage requirements of the SVM with RBF kernel depend on both input dimensionality (d) and the number of support vectors (n_{SV}). In this study we was stressed on reducing the features dimensionality based on deep autoencoder to recude the SVM training and testing time

4. Experimental results

In order to evaluate our STL-NIDSbased approach for the classification of different types of attacks in IoT, we use the Bot-IoT dataset. The implementation of the proposed work was done with Deeplearning4j, the first commercial-grade, open-source, distributed deep-learning library written for Java and Scala. In the sub-sections that follow, the details of the experiments and their results are represented.

4.1 Dataset Used:

The Bot-IoT dataset was recommended in Sunday 05/22/2019 by NickolaosKoroniotis [19]. This dataset is a collections of varied data types of network attacks in Internet of Things environment. It incorporates legitimate and simulated IoT network traffic, along with various types of attacks. The Bot-IoT dataset was

Categor y	Type of Category	Sub- category	Number of Packets
Denial of Service (DoS)	DDoS attacks	TCP	19547603
		UDP	18965106
		HTTP	19771
	DoSattacks	TCP	12315997
		UDP	20659491
		HTTP	29706
Information theft	Key logging	-	1469
	Data theft	-	118
Information gathering	Service scanning	nmap, hping3	1463364
	OS Fingerprinting	nmap, xprobe2	358275

developed on a realistic testbed environment precisely at the Research Cyber Range lab of UNSW Canberra, and has been labeled, with the label features indicated an attack flow, the attacks category and subcategory for possible multiclass classification purposes. The researchers used four Kali Linux VMs to launch cyber-attacks in parallel for implementing different botnet scenarios. In addition, the Bot-IoT dataset contains approximately 1GB of pcap files. Cyber-attacks and their tools considered in the Bot-IoT dataset are described as follows:

Probing attacks: also so-called, fingerprinting are malicious activities that gather information about victims through scanning remote systems. Depending on the actions performed, probing attacks goal can be examined in two separate categories: OS fingerprinting and port scanning: in the port scanning category, the researchers used the Nmap and Hping3 tools in order to perform a number of different types of port scans. On the other hand they used the Nmap and Xprobe2 tools to launch different types of OS fingerprint scans.

Denial of Service attacks: A DDoS attack is exemplified by the direct attempt of attackers to prevent legitimate users from using a specific service. Also, DDoS attacks is distributed in the way that the attacker is using multiple computers as attack platforms to launch DDoS attacks on one or more targets (the victim or related network) with

the help of Control & Scan server technology. In addition, several types of DDoS attacks namely, UDP-flood, TCP-flood, and HTTP flood, was been performed in the Bot-IoT dataset. In addition, the researchers used the Hping3 tool for both DDoS, DoS for TCP and UDP. The hping3 indicates the type of attacks such as: SYN TCP attack, the type of the packets is sent as fast as possible, it specifies the packet body size, and indicates the targeted port. In other hand, for HTTP DDoS and DoS attacks, the researchers used the Golden-eye tool, to indicate the targeting Port number and IP address of Ubuntu server, the method type such as post, number of sockets, and number of concurrent workers.

Tab 2: Statistics of attacks in Bot-IoT dataset

Information Theft: is a group of attacks where an attackers seeks to compromise the security of a machine in order to obtain sensitive data. Based on the target of the attack, the information theft can be split into two major subcategories. The first subcategory is Key logging and the second one is Data theft. The researcher used Metasploit framework to exploit weaknesses in the target compromised devices in both keylogging activities and Data theft. The aim of this framework is established an ssh connection and access the logkeys software, record a user's keystrokes, potentially stealing sensitive credentials in the compromised host and then download the recordings.

4.2 Features Normalization

Several of the features of the Bot-IoT dataset have very large ranges between the maximum and minimum values, such as the difference between the maximum and minimum values in "flow duration" [0, 785673], where the minimum is 0 and the maximum value is 78, 5673. This large difference also exists in other feature values, such as Stime (Record start time) drate (Destination-to-source packets per second) and srate feature (Total packets per second in transaction). Thereby making the feature values incomparable and unsuitable for processing. Hence, these features are normalized by using standardization data process based on calculating the mean absolute difference for mapping all feature values to the range [0, 1] according to:

$$S_f = \frac{1}{N} [(X_{if} - M_f) + (X_{if} - M_f) + (X_{if} - M_f) + (X_{if} - M_f)] \quad (9)$$

Where N denotes the total number of features, x_i denotes the maximum value from all data points for each feature and m_i denotes the minimum value from all data points. After that, we calculate the standardized measurement (Z score function) of the features

according to the following equation: $Z_i = X_i + M_f$

$S_f(10)$

4.3 Evaluation Metrics

The following parameters were calculated for evaluating the performance of the proposed intrusion detection system. Through these metrics, the proposed NIDS can decide which technique is best suited for this study. These metrics were defined as follows:

Accuracy: is the proportion of correctly predicted events. This metric is one of the machines learning metrics for evaluating classification model. The accuracy value is given in the following equation

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FN+FP} \quad (11)$$

Precision: is the number of positive predictions divided by the total number of positive class values predicted. It is a measure of the number of true positives the model claims compared to the number of positives it claims. The precision value is given in the following equation:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (12)$$

Recall: is the number of positive predictions in the model divided by the number of positive class values in the testdata. The recall value is given in the following equation:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (13)$$

The True Positive (TP) indicates the correct predictions of attacks and False Positive (FP) refers to the normal traffic incorrectly classified as attack. True Negative (TN) refers to the normal data correctly classified as normal data (the number of true normal traffic). False Negative (FN) refers to the attack traffic incorrectly classified as normal traffic (the number of false normal traffic).

4.4 Performance evaluation: impact of the low-dimensional features and different hidden layers and hyper parameter on SVM classifier:

Our experiments aim to evaluate the performance efficiency of the proposed approach. In addition, we stressed on the effectiveness of the low-dimensional features extracted by our approach for binary classification based on IoT bot dataset. Furthermore, the training and testing times were calculated to evaluate

the efficiency of our approach. In addition, we also stressed on addressing network intrusion detection system requirements that have lower and faster computational costs by reducing computational complexity and storage requirements of SVM classifier.

4.4.1 Impact of the low-dimensional features on The Binary Classification

The training and testing process use a training and testing data separately when we evaluate the performance efficiency of the low-dimensional features extracted by our approach for two-category classification (binary classification). Figure 3 shows the experimental results. Our STL-IDS performs better than single SVM. However, STL-IDS performs better in all performance metrics compared with single SVM. The experimental results also show that the proposed approach STL-IDS reduces training and testing times of SVM as shown in table 3.

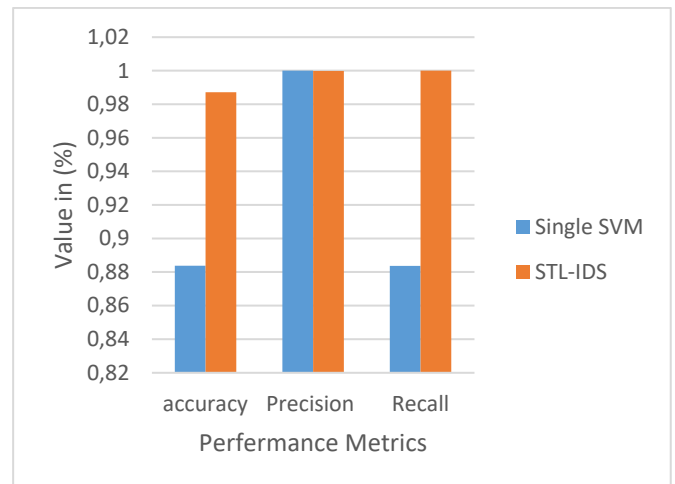


Figure3: Performance metrics values for STL-IDS and single SVM for binary classification.

Methods	Single SVM	STL-IDS
Training Time	1270.48	1009.56
Testing time	1070.48	90450
N(SV)	2756	2723

Tab 3: The performance of our model STL-NIDS compared to single SVM.

4.4.2 The effect of hyper-parameters and different numbers hidden layer number and hidden neurons setting in our Model efficiency

The number of hidden layer and deep autoencoder hyper parameters are the main parameters influencing the classification performance and time training speed.

Thus, developing an efficient deep model for NIDS involves a crucial challenging issue called hyper-parameters optimization. The optimization of our STL-NIDSbased approach was performed over key hyper-parameters and their values are given in Table 4. In addition, how to investigate the effect of hidden layer number and the Deep autoencoder and SVM hyper parameter on the performance of our deep learning model and decrease the testing and training time is another challenge in NIDS.

In order to tune the hyper parameters for all STL-NIDSmodels, in this paper, we used 900,001 instances for training and the remaining 100,000 instances for validation. After that, we selected the best value of hyper parameters based on manual search optimization method. When using Manual Search, we choose some model hyper parameters based on our experience. Then, we train the model, evaluate its accuracy and start the process again. This loop is repeated until a satisfactory accuracy is scored. Moreover, we investigate the effect of both number of hidden neurons and hidden layers size on the STL-NIDSperformance. This method optimize some of our DAE and SVM parameters such as: how many neurons to use in each layer and which hyper parameter and optimizer to use. The final model is trained with all 1 0000,001 instances. Moreover, the STL-NIDSmodel achieves the best result of accuracy when batch size and epochs of pre-training are 300 and 1000, respectively. Moreover, the model got highest accuracy for SVM Classifier with 0.50 bias, $\mu=0.000001$, 300 batch size and 1000 epochs of fine-tuning. In another hand, we investigate an impact the number of hidden neurons and hidden layers on the performance of STL-NIDS.

Hyper parametername	Values	The Best Value
training DAE batch size	20,4060,80,120,300	300
Pre-training DAE epoch size	10, 500, ,800 1000	1000
Weightdecayparameter μ	0.02,0.002,0.04,0.00001,0.0005,0.00003,0.000005	0.000001
training SVM batch size	20, 40 60 80,120,300	300
training SVM epoch size	10, 500, ,800 1000	1000

Tab 4: The tested values of hyper parameters for our STL-NIDS for binary classification.

The STL-NIDS-with five hidden layers and 32 hidden neurons at each layer is superior to other deep networks in testing accuracy for our model based intrusion detection system. Table 5 shows test classification accuracy of STL-NIDSwith different numbers of hidden neurons and layers.

Number of hiddenlayers	Neurons	Testingaccuracy (%)
3	(32,32,32)	97,701
	(64,64,64)	98,323
	(100,100,100)	98,567
5	(32,32,32, 32,32)	98,718
	(64,64,64,64,64)	97,656
	(100,100,100)	98,326
7	(32,32,32,32,32,32,32)	97,786
	(64,64,64,64,64,64,64)	94,678
	(100,100,100,100,100,100,100)	89,767
9	(32,32,32,32,32,32,32,32,32)	91,555
	(64,64,64,64,64,64,64,64,64)	92,432

Tab 5: The performance of the STL-NIDS with different number of hidden neurons and hidden layer.

4.5 Comparison

In order to investigate the effectiveness of our STL-IDS model on intrusion detection performance, we compared our approach to the traditional classification methods. Table 6 shows that the performance of STL-IDS is better than other benchmark algorithms. The outperformance can be noted in terms of accuracy and precision and recall. Due to the fact that deep autoencoder can learn a set of features with better classification capability and dimensionality reduction. Another reason for achieving the better accuracy by deep autoencoder is unsupervised pre-training task. Unsupervised pre-training gives substantially higher test classification accuracy than no pre-training.

Table 6: The performance of our model STL-NIDS compared to machine learning algorithms.

5. Conclusion and Future Work

In this paper, we proposed an unsupervised features learning based approach for developing an efficient and flexible network intrusion detection system (STL-NIDS) in IoT. A deep autoencoder DAE and SVM classifier based NIDS has been carried out. The deep autoencoder is one of the most interesting models to extract features from the high-dimensional data in the context of deep learning. The proposed approach offers a solution to combine the advantages of unsupervised and supervised learning in the context of intrusion detection. In addition, we used the benchmark IoT network intrusion dataset-Bot-IoT to evaluate our intrusion detection accuracy. In all the cases, we observed that the proposed STL-NIDS performed very well compared to single SVM implemented in NIDSs for the normal/anomaly detection when evaluated on the test data. The proposed approach achieved detection accuracy 98.71% on the total 10 test dataset. To the best of our knowledge this is the only framework which has the capability to collect network-level activities in a distributed manner using a combination of DAEs and SVM classifier to detect attack more accurately using Bot-IoT dataset. In future, we plan to implement a real-time NIDS for actual IoT networks using deep learning technique. Additionally, on-the-go feature learning on raw network traffic headers instead of extracted features can be another high impact research in this area.

Notes

1. <https://scikit-learn.org/>
2. [https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbpkkoE?path=%2FCSV%2FTraning%20and%20Testing%20Tets%20\(5%25%20of%20the%20entier%20dataset\)](https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbpkkoE?path=%2FCSV%2FTraning%20and%20Testing%20Tets%20(5%25%20of%20the%20entier%20dataset))
3. <https://deeplearning4j.org/>

References

- [1] Yan, Z., Zhang, P., and Vasilakos, A. V. (2014). A survey on trust management for Internet of Things. *Journal of Network and Computer Applications*, 42, 120–134.
- [2] Brian Lam and Cynthia Larose, "How did the internet of things allow the latest attack on the internet?" <https://www.privacyandsecuritymatters.com/2016/10/how-did-the-internet-of-things-allow-the-latest-attack-on-the-internet/>, 2016
- [3] Bertino, E., & Islam, N. (2017). Botnets and internet of things security. *Computer*, 50(2), 76–79.

Methods	Support Vector Machine	Bayes Naive	Random Forest	STL-IDS
Accuracy	0.88372702	0.96579	0.97579	0.987123
Precision	1	0.9111998	0.941199	0.99989
Recall	0.8837119	0.947890	0.977890	1
Training time	1270.48	3034.43	2885.34	1009.56

- [4] Yousefi-Azar, M., Varadharajan, V., Hamey, L., & Tupakula, U. (2017, May). Autoencoder-based feature learning for cyber security applications. In *2017 International joint conference on neural networks (IJCNN)* (pp. 3854–3861). IEEE.
- [5] Farahnakian, F., & Heikkonen, J. (2018, February). A deep autoencoder based approach for intrusion detection system. In *2018 20th International Conference on Advanced Communication Technology (ICACT)* (pp. 178–183). IEEE.
- [6] Nespoli, P., & Gómez Mármol, F. (2018, April). e-Health Wireless IDS with SIEM integration. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC18)*, Barcelona, Spain (pp. 15–18).
- [7] Attia, M., Senouci, S. M., Sedjelmaci, H., Aglzim, E. H., & Chrenko, D. (2018). An efficient Intrusion Detection System against cyber-physical attacks in the smart grid. *Computers & Electrical Engineering*, 68, 499–512.
- [8] Aygun, R. C., & Yavuz, A. G. (2017, June). Network anomaly detection with stochastically improved autoencoder based models. In *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)* (pp. 193–198). IEEE.
- [9] Aamir, M., & Zaidi, S. M. A. (2019). Clustering based semi-supervised machine learning for DDoS attack classification. *Journal of King Saud University-Computer and Information Sciences*.
- [10] Abraham, B., Mandya, A., Bapat, R., Alali, F., Brown, D. E., & Veeraraghavan, M. (2018, July). A Comparison of Machine Learning Approaches to Detect Botnet Traffic. In *2018 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE.
- [11] Anthi, E., Williams, L., & Burnap, P. (2018). Pulse: an adaptive intrusion detection for the internet of things.
- [12] Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016, May). 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)* (pp. 21–26).
- [13] Camacho, J., Macia-Fernandez, G., Fuentes-Garcia, N. M., & Saccetti, E. (2019). Semi-supervised multivariate statistical network monitoring for learning security threats. *IEEE Transactions on Information Forensics and Security*, 14(8), 2179–2189.
- [14] Benzebouchi, N. E., Azizi, N., & Ayadi, K. (2019). A computer-aided diagnosis system for breast cancer using deep convolutional neural networks. In *Computational Intelligence in Data Mining* (pp. 583–593). Springer, Singapore.
- [15] Tran, D., Mac, H., Tong, V., Tran, H. A., & Nguyen, L. G. (2018). A LSTM based framework for handling multiclass imbalance in DGA botnet detection. *Neurocomputing*, 275, 2401–2413.
- [16] Oskan, S., Yildirim, E. N., Karatas, G., & Cuhaci, L. (2019, April). Intrusion Detection Systems with Deep Learning: A Systematic Mapping Study. In *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)* (pp. 1–4). IEEE.
- [17] Al-Qatf, M., Lasheng, Y., Al-Habib, M., & Al-Sabahi, K. (2018). Deep learning approach combining sparse autoencoder with SVM for network intrusion detection. *IEEE Access*, 6, 52843–52856.
- [18] Kameoka, H., Kaneko, T., Tanaka, K., & Hojo, N. (2019). ACVAE-VC: Non-parallel voice conversion with auxiliary classifier variational

- autoencoder. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(9), 1432-1443.
- [19] Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100, 779-796.
- [20] Ahmad, I., Basher, M., Iqbal, M. J., & Rahim, A. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE Access*, 6, 33789-33795.
- [21] Debashi, M., & Vickers, P. (2018). Sonification of network traffic for detecting and learning about botnet behavior. *IEEE Access*, 6, 33826-33839.
- [22] Gao, X., Shan, C., Hu, C., Niu, Z., & Liu, Z. (2019). An Adaptive Ensemble Machine Learning Model for Intrusion Detection. *IEEE Access*, 7, 82512-82521.
- [23] Aljawarneh, S., Aldwairi, M., & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, 25, 152-160.
- [24] Aburomman, A. A., & Reaz, M. B. I. (2016). A novel SVM-kNN-PSO ensemble method for intrusion detection system. *Applied Soft Computing*, 38, 360-372.
- [25] Alauthaman, M., Aslam, N., Zhang, L., Alasem, R., & Hossain, M. A. (2018). A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks. *Neural Computing and Applications*, 29(11), 991-1004.
- [26] Qu, X., & Li, M. (2019). Detection of DOS Flooding Attacks with an Improved Growing Hierarchical SOM.
- [27] Chawathe, S. S. (2018, November). Monitoring IoT networks for botnet activity. In 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA) (pp. 1-8). IEEE.
- [28] Marir, N., Wang, H., Feng, G., Li, B., & Jia, M. (2018). Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. *IEEE Access*, 6, 59657-59671.
- [29] Homayoun, S., Ahmadzadeh, M., Hashemi, S., Dehghantanha, A., & Khayami, R. (2018). BotShark: A deep learning approach for botnet traffic detection. In *Cyber Threat Intelligence* (pp. 137-153). Springer, Cham.
- [30] Isot botnet dataset, jan 2017. <http://www.uvic.ca/engineering/ece/isot/datasets/>.
- [31] Unb iscx botnet dataset, jan 2017. <http://www.unb.ca/research/isxc/dataset/ISCX-botnet-dataset.html#Botnet%20Data%20set>.
- [32] Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2017). Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access*, 5, 18042-18050.
- [33] Carl.Ls, Robert.W, David.L, and W. Strayer. Using machine learning techniques to identify botnet traffic. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*. IEEE, nov 2006. <https://doi.org/10.1109/lcn.2006.322210>.
- [34] Junjie.Z, Roberto.P, Wenk. L, Xiapu Luo, and Unum Sarfraz. Building a scalable system for stealthy p2p-botnet detection. *IEEE Transactions on Information Forensics and Security*, 9 (1): 27-38, jan 2014. <https://doi.org/10.1109/tifs.2013.2290197>.
- [35] Aamir, M., & Zaidi, S. M. A. (2019). Clustering based semi-supervised machine learning for DDoS attack classification. *Journal of King Saud University-Computer and Information Sciences*.
- [36] A. Dainotti, A. Pescapé, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35-40, Feb. 2012
- [37] Hoang, T. M., Nguyen, N. M., & Duong, T. Q. (2019). Detection of eavesdropping attack in UAV-aided wireless systems: Unsupervised learning with one-class SVM and k-means clustering. *IEEE Wireless Communications Letters*.
- [38] Thaseen, I. S., & Kumar, C. A. (2017). Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *Journal of King Saud University-Computer and Information Sciences*, 29(4), 462-472
- [39] Wang, X., Du, Y., Lin, S., Cui, P., Shen, Y., & Yang, Y. (2019). adVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection. *Knowledge-Based Systems*, 105187.
- [40] Benzebouchi, N. E., Azizi, N., Ashour, A. S., Dey, N., & Sherratt, R. S. (2019). Multi-modal classifier fusion with feature cooperation for glaucoma diagnosis. *Journal of Experimental & Theoretical Artificial Intelligence*, 31(6), 841-874.
- [41] Bettge, A., Roscher, R., & Wenzel, S. (2017). Deep self-taught learning for remote sensing image classification. *arXiv preprint arXiv:1710.07096*.
- [42] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data withneural networks, *Science* 313 (5786) (2006) 504-507.[26]
- [43] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training ofdeep networks, in: *Advances in Neural Information Processing Systems*, 2007,pp. 153-160.
- [44] E. de la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, and A. MartínezÁlvarez, , "Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical selforganising maps," *Knowledge-Based Syst.*, vol. 71, pp. 322-338, 2014.