AutoIDS: Auto-encoder Based Method for Intrusion Detection System

Mohammed Gharib*, *Member, IEEE,* Bahram Mohammadi*, Shadi Hejareh Dastgerdi, and Mohammad Sabokrou, *Member, IEEE*

***Abstract*—Intrusion Detection System (IDS) is one of the most effective solutions for providing primary security services. IDSs are generally working based on attack signatures or by detecting anomalies. In this paper, we have presented AutoIDS, a novel yet efficient solution for IDS, based on a semi-supervised machine learning technique. AutoIDS can distinguish abnormal packet flows from normal ones by taking advantage of cascading two efficient detectors. These detectors are two encoder-decoder neural networks that are forced to provide a compressed and a sparse representation from the normal flows. In the test phase, failing these neural networks on providing compressed or sparse representation from an incoming packet flow, means such flow does not comply with the normal traffic and thus it is considered as an intrusion. For lowering the computational cost along with preserving the accuracy, a large number of flows are just processed by the first detector. In fact, the second detector is only used for difficult samples which the first detector is not confident about them. We have evaluated AutoIDS on the NSL-KDD benchmark as a widely-used and well-known dataset. The accuracy of AutoIDS is 90.17% showing its superiority compared to the other state-of-the-art methods.**

***Index Terms*—IDS, Security Services, Anomaly Detection, Machine Learning, Semi-supervised, Encoder-Decoder.**

I. Introduction

NOWADAYS, providing security services in different computer networks is an issue of paramount significance. The principal security services required by almost all of the communication networks, irrespective of their types, are confidentiality, authenticity, non-repudiation, integrity, and availability. Cryptography is an effective solution for providing security services [1], however, it is ineffective against availability attacks. IDS is an alternative solution to provide availability as a crucial security service. Hence, IDS and cryptography complement each other to cover all of the aforementioned security services.

Intrusion detection methods can be generally categorized into two groups: (1) signatures based, and (2) anomaly detection based [2]. The signature based methods can effectively detect network attacks which have been already known and their pattern is available, while they fail in detecting unknown and also zero-day intrusions. The latter solution, i.e., anomaly detection based, seems to be more effective due to its capability of detecting unknown and new attacks. Machine learning is one of the most promising techniques for anomaly detection [3]. Furthermore, by the recent exponential growth in the volume of the computer network data, this technique attracts even much more attention. Machine learning technique basically falls into three categories, supervised, semi-supervised and unsupervised.

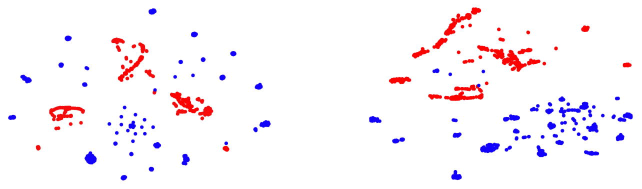

Fig. 1. Visualizing the original normal (blue) and anomalous (red) packet flows using t-SNE before and after representing by the sparse AE. (*Left*:) original samples, (*Right*:) latent space of the sparse AE. As can be seen in this figure, the sparse AE makes the original normal and abnormal packet flows more separable from each other.

The majority of the previously proposed methods for IDS are based on supervised learning [4] meaning that they have access to both normal and abnormal samples in training duration and they are able to efficiently learn the characteristics of merely the available normal and abnormal flows. Accordingly, such methods can accurately detect abnormal packet flows only if they are available in training procedure (or be similar to training samples), otherwise, they fail to properly detect anomalies. Broadly, the generalization of supervised methods for the IDS task is not enough to be able to detect new types of attacks. Training an intrusion detection method in a semi-supervised fashion, i.e., training a model just on the normal traffic, without knowing anything about the abnormalities, is more general especially for new (unseen) network attacks. Handling this problem leads to a generalized method that is able to act properly against new types of intrusions.

In this paper, we have proposed AutoIDS, a semi-supervised deep learning method to precisely detect the anomalous traffic. We have named it AutoIDS, since it exploits two Auto-Encoders (AEs) for detecting anomalies in communication networks in a cascade manner. The key idea is to make a decision about the types of incoming network flows in two separate phases, each phase with a different detector. In this case, incoming network traffic is investigated in different feature spaces enabling us to differentiate the process of making decision about simply detectable and more complex incoming packet flows. Thus, the accuracy of the detection process is increased. Firstly, the sparse AE is exploited to

* M. Gharib and B. Mohammadi contributed equally to this paper.

M. Gharib and M. Sabokrou are with the Institute for Research in Fundamental Sciences (IPM) (email: gharib, sabokro@ipm.ir).

B. Mohammadi is with the Computer Engineering Department, Sharif University of Technology (email: bmohammadi@alum.sharif.edu).

S. H. Dastgerdi is with the Computer Engineering Department, Iran University of Science and Technology (email: shadi_hejareh@iust.ac.ir).

detect the simple detectable flows. Then, the rest of packet flows are sent to a non-sparse AE. Thereby, distinguishing normal flows from complex anomalies are performed by AE with more accurate processing. From now on, for simplicity, we use AE instead of non-sparse AE, in short. Cascading several weak detectors and early rejection of the most samples for improving the complexity and accuracy are widely used in machine learning community [5], [6], especially for anomaly detection in videos [7]–[9]. Inspired by such researches, we have devised an effectual method for IDS.

In computer networks, the incoming network flows are passing through a gateway. Therefore, they can be monitored at this point. Since such network traffic is heavy, their processing must be performed as fast as possible. To this end, a considerable volume of incoming flows analyzed and detected in the first step, i.e., by the sparse AE, with low time complexity. The rest of the incoming network flows are then sent to the AE with a more complicated decision making process. Note that the number of packet flows forwarded to the second phase, form a portion of the entire incoming traffic. Consequently, in addition to increasing the accuracy of anomaly detection, the time complexity is reduced noticeably.

We have interestingly investigated that the sparsity value of a represented packet flow by the sparse AE as well as the calculated reconstruction error for an incoming flow through the AE which are trained merely on normal flows, are considered as two effective discriminative measures for hunting the abnormal traffic in computer networks. Proposing an efficient method to specify thresholds for both AE and sparse AE to distinguish between normal and anomalous traffic, is of crucial importance and accordingly forms part of our contribution. Additionally, to show the effectiveness of AutoIDS, we have investigated the power of the sparse AE to separate normal and abnormal flows using t-distributed Stochastic NEighbor (t-SNE) [10]. In fact, t-SNE is an algorithm for dimensionality reduction which is appropriate for visualizing high dimensional data. As an instance and by utilizing t-NSE, Fig. 1-left shows the feature vector of original normal and anomalous traffic, while Fig. 1-right shows the represented original packet flows by the sparse AE, trained only on normal flows. In this figure, there are 500 samples for each type of normal and abnormal traffic. As can be seen in Fig 1, sparse representation of the packet flows is more separable than the original version of them. However, it is not sufficiently discriminative for separating difficult samples and just works well for simple ones. To cope with this weakness, as explained earlier, the decision about difficult samples is taken by the AE, which works based on the reconstruction error.

The main contribution of this paper is proposing a novel, effective and accurate solution for detecting abnormal traffic with an acceptable time complexity. More specifically, our contributions are: (1) proposing a semi-supervised deep learning method to detect anomalies with higher performance, in terms of accuracy, compared to the other state-of-the-art solutions. To the best knowledge of us, this paper is one of the first deep learning based methods dealing with the intrusion detection as an semi-supervised approach, and (2) evaluating the proposed method under realistic circumstances and also showing its superiority.

The rest of this paper is organized as follows. In Section II, we have briefly reviewed the other proposed approaches. A detailed explanation of AutoIDS has been provided in Section III. Evaluation which includes experimental results alongside their analysis resides in Section IV. Finally, we have concluded this paper in Section V.

II. Related Work

In this section, we have explored the intrusion detection method based on machine learning technique. Generally, IDSs can be divided into two groups: (1) signature based, and (2) anomaly detection based. In signature-based solutions, the incoming network traffic is compared with a database of signatures. Therefore, this method needs a continuous update in order to add new intrusion patterns. Although this way of approaching the problem is efficient for detecting the known attacks [11], it is incapable of detecting new (unseen) intrusions. On the other hands, anomaly detection based methods are able to detect the unknown and zero-day network attacks. In contrary to the signature based solutions, anomaly detection based methods, which are broadly based on machine learning technique, are more robust against unknown attacks [12]. From the machine learning point of view, the anomaly detection based methods can be categorized into three approaches: (1) supervised, (2)semi-supervised, and (3) unsupervised [13]. In supervised learning, sufficient amount of labeled data from both normal traffic and the known intrusions are fed to the neural network. In this case, the system can detect such network attacks more accurately. However, labeling all of the network flows is a highly time-consuming and also expensive process. Semi-supervised learning is used to utilize both labeled and unlabeled data to decrease the cumbersome of labeling packet flows, especially when the system faces a big volume of data and diverse abnormal samples. Unsupervised learning aims to cluster the samples with no knowledge.

There are many algorithms based on supervised learning such as K-Nearest Neighbor (KNN) [14], neural network [15], Support Vector Machine (SVM) [16], and etc. Chand *et al.* [17] stacked a SVM, which is considered as an effective classifier for intrusion detection, followed by another ones such as BayesNet [18], AdaBoost [15], Logistic [19], IBK [20], J48 [21], RandomForest [22], JRip [23], and OneR [24] to improve the detection accuracy. Tao *et al.* [25] used the Genetic Algorithm (GA) alongside the SVM. Basically, GA can increase the true positive rate, decrease the error rate, and improve the classification time complexity of the SVM by selecting the best features. Ling *et al.* [26] introduced a random forest feature selection algorithm and proposed a multi-classifier method. The random forest extracts optimal features for training SVM, decision tree, NaiveBayes [27], and KNN classification algorithms. Then, deep learning is adopted to stack these classifiers. This proposed method improves the accuracy of intrusion detection in comparison with the majoring voting algorithms. Ashfaq *et al.* [28] proposed a fuzziness based semi-supervised learning approach to improve the performance of classifiers for IDSs, however, it utilizes

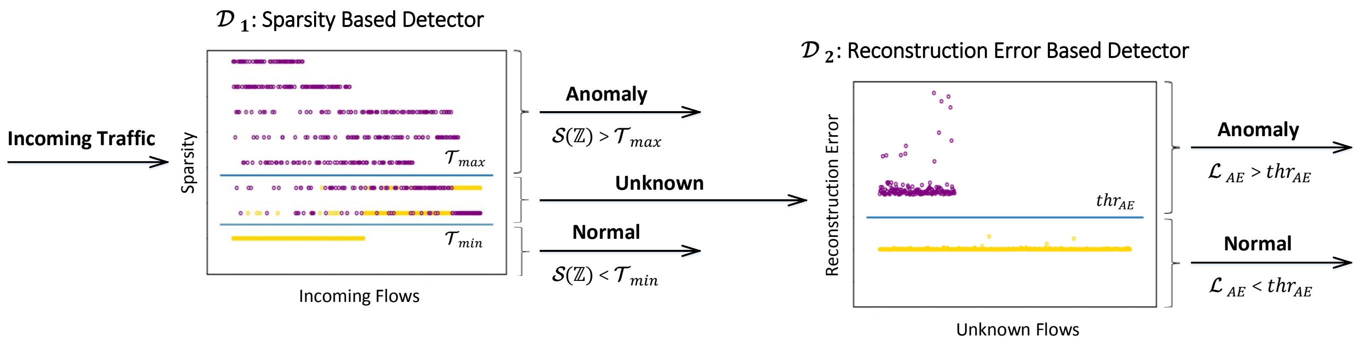
$\mathcal{D}_1$: Sparsity Based Detector
Incoming Traffic
Sparsity
$\mathcal{T}_{max}$
$\mathcal{T}_{min}$
Incoming Flows
Anomaly
$\mathcal{S}(\mathbb{Z}) > \mathcal{T}_{max}$
Unknown
Normal
$\mathcal{S}(\mathbb{Z}) < \mathcal{T}_{min}$
$\mathcal{D}_2$: Reconstruction Error Based Detector
Reconstruction Error
thr_{AE}
Unknown Flows
Anomaly
$\mathcal{L}_{AE} > thr_{AE}$
Normal
$\mathcal{L}_{AE} < thr_{AE}$

Fig. 2. The process of anomaly detection in AutoIDS. Incoming packet flows are processed in two phases. More complex flows which the sparse AE is not confident about their types, are sent to the next phase. At this point, the AE makes a decision about the type of incoming traffic whether it is normal or not. For AE, accuracy is preferred to speed while this is opposite for sparse AE. In the first step, τ_{min} and τ_{max} are the minimum and maximum threshold for the sparse AE, respectively. Also, $\mathcal{S}(\mathbb{Z})$ shows the sparsity value. In the second step, $\mathcal{L}_{\text{AE}}$ is the loss of AE which is calculated for each of incoming network flows and thr_{AE} is the threshold for separating the incoming traffic.

supervised learning. The proposed network has a single hidden layer, and the output is a fuzzy membership network. Fuzzy value is used for categorizing unlabeled samples. The classifier is retrained after incorporating each category separately into the original training set.

There are further literature works that have combined the supervised and unsupervised approaches to propose hybrid algorithms. Aljawarneh *et al.* [29] developed a new hybrid model to minimize the time complexity of determining the feature association impact scale. This hybrid algorithm consists of several classifiers including J48, Meta Pagging [30], RandomTree [31], REPTree [32], AdaBoostM1 [33], DecisionStump [34], and NaiveBayes.

Deep learning as one of the most popular branches of machine learning technique in the field of IDS, uses multiple hidden layers in order to extract features, automatically. However, machine learning based methods need an expert to manually extract features. Deep learning based solutions improve the detection accuracy compared to traditional methods [35]. Hodo *et al.* [13] presented a taxonomy including shallow and deep neural networks for IDSs to demonstrate the effect of feature selection and feature extraction on the performance of the anomaly detection procedure. Javaid *et al.* [36] proposed a deep learning method based on sparse AE and softmax-regression to implement effective and flexible Network Intrusion Detection System (NIDS). This technique consists of two steps, (1) a sparse AE is used for feature learning due to its ease of implementation and good performance, and (2) softmax-regression is used for the classification purpose. Farahnakian *et al.* [37] proposed an approach based on a stacked AE. The whole neural network includes four AEs in which the output of each AE is considered as an input for the next AE, in the next layer. The stacked AE is followed by a softmax layer classifying incoming network flows into normal and anomaly. Shone *et al.* [38] proposed a Non-symmetric Deep AE (NDAE) for unsupervised feature learning and introduced the stacked NDAE and the random forest for classification. This model is a combination of deep and shallow learning to exploit their strengths and reduce the analytical overhead. Papamartzivanos *et al.* [39] proposed a methodology that combines self-taught learning [40] with MAPE-K framework [41]. self-taught learning is utilized to transfer learning from unlabeled data to achieve high attack detection accuracy. Also, MAPE-K framework is used for delivering a self-adaptive IDS to identify the previously unseen intrusions via reconstructing of unlabeled data. The work of [42] inspired by [43], proposed a semi-supervised method based on deep generative method, where two deep neural networks are adversarially and jointly trained to generate abnormal flows and distinguish fake samples from the real ones.

III. AutoIDS Algorithm

In this section, we have introduced AutoIDS, the novel and effective approach to detect anomalies in different types of communication networks, with the aim of high accuracy and low computational cost. To cope with the supervised weaknesses, i.e., low generalization for detecting the unseen attacks and needed a numerous labeled samples, we have proposed an effective method with minimum supervision which is able to be efficiently trained by considering merely the normal samples. The key idea is to exploit a combination of an AE and a sparse AE in a cascade manner aiming to increase the accuracy and decrease the time complexity, at the same time.

AutoIDS consists of two detectors, $\mathcal{D}_1$ and $\mathcal{D}_2$ including sparse AE and AE, respectively. In $\mathcal{D}_1$, the criterion for distinguishing normal from anomalies is the sparsity. In fact, the sparse AE is trained only on normal traffic. Accordingly, it is optimized to provide a sparse representation from normal packet flows. Most probably anomalous traffic do not satisfy the imposed constraint and the sparse AE needs more active neurons to reconstruct the input, i.e., the sparsity value of their latent representations is not lower than a specific threshold. To reduce the overall complexity of AutoIDS, we have considered $\mathcal{D}_1$ as a lightweight detector, and it is allowed to work with high False Positive Rate (FPR). Flows which $\mathcal{D}_1$ are not certain about their types are sent to the next more accurate detector.

In $\mathcal{D}_2$, reconstruction error is used as the criterion for separating different types of flows. Analogous with the sparse AE, the AE is also trained just using normal traffic. Hence, it learns to minimize the reconstruction error. Accordingly, when the reconstruction error of an incoming flow is more than a predefined threshold, it does not satisfy the desired constraint and thus it is considered as an anomaly. It is worth mentioning that, considering two different detectors, i.e., $\mathcal{D}_1$ and $\mathcal{D}_2$, empowers the system to make a decision about different types of traffic in two individual feature spaces. In this way, AutoIDS is able to differentiate normal traffic form anomalies with more accuracy. Fig. 2 outlines our proposed solution for anomaly detection in computer networks.

We have divided this section into three parts. The classification manner employed by the sparse AE has been explained in the first part. Then, we have presented a detailed explanation about separating normal and anomalous traffic by the AE in the second part. Finally, AE and sparse AE are considered as a whole and the way of detecting anomalies by AutoIDS has been investigated.

A. $\mathcal{D}_1$: *Sparsity Based Detector*

Sparse AEs are originally used for unsupervised feature learning while they improve generality [44]. However, we have exploited the value of the sparsity in the latent representation like what has been also considered as an effective solution in [45]. The overall scheme of the sparse AE, which has been used in our proposed method, is depicted in Fig. 3. As it is clear in this figure, the sparse AE consists of two major parts, encoder and decoder. Furthermore, to the latent representation becomes over-complete, the number of neurons in the latent representation (m) should be considered more than the size of the input layer (n), i.e., $m > n$. The applied constraint on the sparse AE during the training phase, is sparsity. As explained previously, the sparse AE is merely trained on normal traffic. Therefore, the sparse AE parameters (θ_{SAE}) are optimized based the normal flows. The proposed spares AE for detecting the anomalous packet flows is trained based on Equation 1.

$$\mathcal{L}_{\text{SAE}} = \|X - \hat{X}\|_2^2 + \textstyle\sum_{j=1}^{m} KL(\rho\|\hat{\rho}_j)$$
$$\hat{X} = \sigma_2(W_2\sigma_1(W_1X + b_1) + b_2) \tag{1}$$

where $X \in \mathbb{R}^n$ is the characteristic set of each incoming flow, $\hat{X} \in \mathbb{R}^n$ is the characteristic set of the reconstructed version of each incoming flow, σ_1 and σ_2 are activation functions, W_1 and W_2 are encoder and decoder weight matrices, respectively, b_1 and b_2 are bias terms, ρ is the sparsity parameter, $\hat{\rho}_j$ is the expected value for $j-th$ unit in hidden layer and $W_1 \times X = \mathbb{Z} \in \mathbb{R}^m$ is the latent representation.

With respect to Equation 1, the sparse AE is trained to minimize $\mathcal{L}_{\text{SAE}}$ for normal flows and also produce $\mathbb{Z}$ whose sparsity value, i.e., $\mathcal{S}(\mathbb{Z})$, is lower than ρ, where $\mathcal{S}(\mathbb{Z} = \mathcal{D}_1(X))$ is the sparsity value of the representation of X using $\mathcal{D}_1$ network and can be calculated based on the Equation 2.

$$\mathcal{S}(\mathbb{Z}) = \frac{\#_{\neq 0}(\mathbb{Z})}{m} \tag{2}$$

Where $\#_{\neq 0}(\mathbb{Z})$ and m determine the total number of active neurons (non-zero elements) and total number of neurons

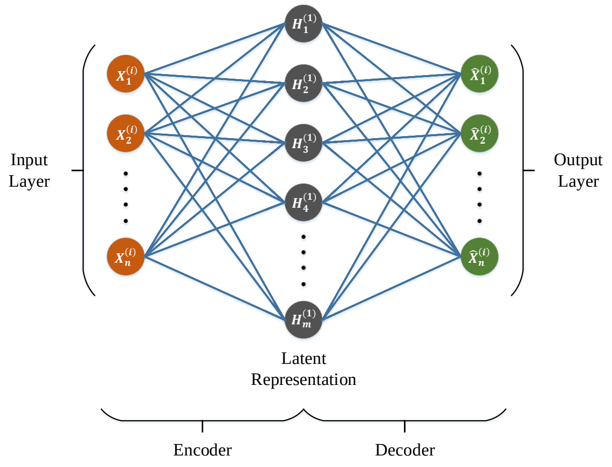

Fig. 3. The overall scheme of our proposed sparse AE. In this figure, $X_j^{(i)}$ and $X_j'^{(i)}$ are the $j-th$ feature of $i-th$ sample and its reconstructed version, respectively. $H_j^{(i)}$ shows the $j-th$ feature of the latent representation of $i-th$ sample. In AutoIDS $n = 122$ and $m = 140$.

(including active and inactive neurons) in $\mathbb{Z}$, respectively. As can be seen in Fig. 2, to speed up the decision making process of AutoIDS about the incoming traffic, two thresholds are determined in this phase, τ_{min} and τ_{max}. Flows with $\mathcal{S}(\mathbb{Z}) > \tau_{max}$ are detected as anomaly. On the contrary, when an incoming flow follows the concept of normal traffic and its sparsity value is lower than τ_{min}, it is detected as normal. Otherwise, i.e., $\tau_{min} < \mathcal{S}(\mathbb{Z}) < \tau_{max}$, the incoming packet flow is considered as an unknown flow and thus it needs further process due to lack of precision in $\mathcal{D}_1$. Although anomalies are detected by $\mathcal{D}_1$ are not complicated, including both normal and anomalous samples, their number is noticeable. In fact, a large volume of incoming traffic are detected by $\mathcal{D}_1$ with low time complexity.

B. $\mathcal{D}_2$: *Reconstruction Error Based Detector*

AEs are mainly used for dimensionality reduction and unsupervised feature learning [44]. However, as has also been investigated in [46], leveraging the reconstruction error is very useful. The overall architecture of our desired AE is represented in Fig. 4. Unlike the sparse AE, the whole structure of the AE is used to reconstruct the incoming flows. In $\mathcal{D}_2$, the AE is forced to reconstruct only the normal flows using a hidden layer with the lower number of neurons (k) compared to the number of input neurons (n) in the training phase ($k < n$). The loss function of AE is shown in Equation 3 called squared euclidean error. Note that the loss function of AE is similar to sparse AE, but the sparsity constraint no longer exists.

$$\mathcal{L}_{\text{AE}} = \|X - \hat{X}\|_2^2 \tag{3}$$

$\mathcal{L}_{AE}$ is minimized for normal traffic and θ_{AE}, i.e., the parameters vector of AE, is optimized on reconstructing the incoming normal flows with low reconstruction error. In fact,

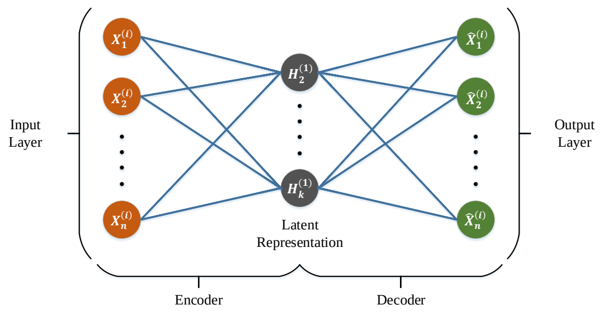

Fig. 4. The overall scheme of our proposed AE. In this figure, $X_j^{(i)}$ and $X_j'^{(i)}$ are the $j-th$ feature of $i-th$ sample and its reconstructed version, respectively. $H_j^{(i)}$ shows the $j-th$ feature of the latent representation of $i-th$ sample. In AutoIDS $n = 122$ and $k = 80$.

we have aimed to exploit the lack of ability of the AE to reconstruct incoming abnormal flows to distinguish different types of traffic. Reconstruction error is calculated for all of the incoming packet flows entering the AE. If it be lower than a predetermined threshold (thr_{AE}), the packet flow is considered as normal, otherwise, it is detected as anomaly.

C. Training AutoIDS

In AutoIDS, $\mathcal{D}_1$ and $\mathcal{D}_2$ learn the concept of normal traffic, individually. In other words, both of them are trained on all of the normal packet flows included by the training set and thus their learning process is not done according Fig. 2. $\mathcal{D}_1$ learns to provide a sparse representation from normal traffic. Consequently, the number of active neurons in the hidden layer should increase in case of facing anomalous samples. Also, $\mathcal{D}_2$ is trained to provide a compressed representation of normal flows. When an abnormal packet flow enters $\mathcal{D}_2$, it is not capable of properly providing the reconstructed version. Therefore, the reconstruction error increases which is the criterion for distinguishing different types of incoming traffic from each other.

D. Anomaly Detection

In this section, we have cohesively described the manner of detecting anomalies in AutoIDS. Suppose an incoming flow enters the computer network. At first, as it is shown in Fig. 2, the incoming flow enters the sparse AE. Formal representation of processing the incoming flow in the sparse AE is given as follows.

$$\mathcal{D}_1(x) = \begin{cases} Normal, & \text{if } \mathcal{S}(\mathbb{Z}) < \tau_{min}. \\ Anomaly, & \text{if } \mathcal{S}(\mathbb{Z}) > \tau_{max}. \\ Unknown, & \text{otherwise.} \end{cases}$$

A large amount of normal and anomalous flows, which are not complicated, are detected in this phase. Accordingly, Flows which their behavior are more complex, i.e., unknown flows, are send to the next detector for further processing irrespective of their actual types. This cascading method results in increasing accuracy, while the time complexity is reduced compared to each detector individually. The second phase is shown in below.

$$\mathcal{D}_2(x \in Unknown) = \begin{cases} Anomaly, & \text{if } \mathcal{L}_{\text{AE}} > thr_{\text{AE}}. \\ Normal, & \text{otherwise.} \end{cases}$$

The second phase is more complex and more precise. Hence, it is suitable for distinguishing anomalies from normal traffic in case that they are very similar to each other. The point should be noticed is the way of determining thr_{AE}. In fact, this threshold is specified based on Receiver Operating Characteristic (ROC) curve. The best threshold is the one where $recall$ is equal (roughly equal) to $1-fpr$. If we consider thr_{AE} as the best threshold, then:

$$\text{Recall}(thr_{\text{AE}}) \simeq 1 - \text{FPR}(thr_{\text{AE}})$$

IV. Evaluation

In this section, We have evaluated the performance of our proposed method, i.e., AutoIDS, on NSL-KDD[1] dataset [47]. We have compared the results of AutoIDS with the state-of-the-art intrusion detection methods which are based on anomaly detection. We have also expressed a comprehensive analysis of the obtained results confirming that our method is able to be performed as an effective method for intrusion detection.

A. Dataset

KDDCUP'99 dataset is constructed based on DARPA'98 IDS evaluation program [48]. This dataset is provided by Stolfo *et al.* [49]. A large number of duplicate records is the fundamental problem of this dataset. Therefore, classifiers had a bias toward the frequent records. Tavallaee *et al.* [47] improved KDDCUP'99 by removing the duplicate records, and named it NSL-KDD. Furthermore, the reasonable number of samples in both training and test sets has enabled researchers to use the entire dataset for the process of evaluation and not to select them randomly. Thus, they are able to fairly compare their results with other proposed methods. Each of flows in NSL-KDD includes 41 features. Three of these features are non-numeric values and the processing of such features is not possible. To solve this issue, a pre-processing for converting those three features to numerical type is needed (See subsection IV-B).

NSL-KDD is composed of two subsets: (1) KDDTrain$^+$ and, (2) KDDTest$^+$. The available attack categories in this dataset are (1) Denial of Service (DoS), (2) probing, (3) User to Root (U2R), and (4) Root to Local (R2L). KDDTrain$^+$ contains 24 attack types belonging to the above-mentioned abnormal classes. It is worth mentioning that, there are 17 attack types in KDDTest$^+$ which is not included by KDDTrain$^+$. In other words, the test process is performed on some malicious flows (network intrusions) which certainly has not been got involved in the training procedure. This characteristic leads to

[1]Available at https://www.unb.ca/cic/datasets/nsl.html

TABLE I
THE SPECIFICATION OF THE NSL-KDD DATASET.

		Number of Records	
Category	**Class**	**KDDTrain$^+$**	**KDDTest$^+$**
Anomaly	DoS	45927	7458
	Probing	11656	2421
	R2L	995	2754
	U2R	52	200
Normal	—	67343	9711
Total	—	125973	22544

examining the generalization of different proposed approaches that use NSL-KDD for the evaluation process. Table I shows the detailed information of the NSL-KDD dataset.

B. Pre-processing

As stated previously, since NSL-KDD contains non-numerical values (protocol, service and flag), a prepossessing is essential to process the entering flows. We have converted these three features to numerical values using one-hot encoding. For instance, three protocols are encoded as follows.

$$\text{one-hot encoded} = \begin{cases} (0,0,1), & \text{if the protocol is tcp.} \\ (0,1,0), & \text{if the protocol is udp.} \\ (1,0,0), & \text{if the protocol is icmp.} \end{cases}$$

Services and flags are encoded in the same way. There are three different protocols, 70 different services, 11 different flags and 38 other numeric features. Accordingly, there are 122 features in total. After converting all columns (features) to numerical values, standardization and also normalization is needed to be ready for the training phase. Features are standardized based on the Equation 4.

$$x(i) = \frac{x(i) - mean(x(i))}{standard\_deviation(x(i))} \qquad \forall\ i \in [1, 122] \tag{4}$$

Where $x(i)$ is the standardized form of $i-th$ feature. Standardization procedure makes the mean and the standard deviation of each feature equal to 0 and 1, respectively. Then, normalization is done using the Equation 5.

$$x(i) = \frac{x(i) - min(x(i))}{max(x(i)) - min(x(i))} \qquad \forall\ i \in [1, 122] \tag{5}$$

Where $x(i)$ is the normalized form of $i-th$ feature. Finally, $x(i)$ is fed to both encoder-decoder networks in the stage of training.

TABLE II
AE AND SPARSE AE SPECIFICATIONS.

Parameters	AE	Sparse AE
Hidden layers	1	1
Hidden layer size (neurons)	80	140
Encoder activation function	ReLU	ReLU
Decoder activation function	Sigmoid	Sigmoid
Loss function	MSE	MSE
Optimizer	Adam	Adam
Regularizer	–	10e-5

C. Implementation Details

We have implemented AutoIDS using Keras framework [2]. The evaluations are performed on GOOGLE COLAB[3]. The detailed information and learning parameters of $\mathcal{D}_1$ and $\mathcal{D}_2$ are provided in Table II.

AutoIDS has been evaluated based on five measures, accuracy (ACC), precision (PR), recall (RE), f-score (FS) and false positive rate (FPR). These measures have been calculated through the Equation 6.

$$\begin{aligned} ACC &= \frac{TP+TN}{TP+FN+TN+FP} \\ PR &= \frac{TP}{TP+FP} \\ RE &= \frac{TP}{TP+FN} \\ FS &= \frac{2*PR*RE}{PR+RE} \\ FPR &= \frac{FP}{FN+FP} \end{aligned} \tag{6}$$

Where True Positive (TP) and False Negative (FN) refers to those anomalous flows which are classified correctly as anomaly and mistakenly as normal, respectively. Similarly, the True Negative (TN) and False Positive (FP) are the normal flows which are classified as normal and abnormal, respectively.

D. Experimental Results

We have compared our proposed method, AutoIDS, with the other state-of-the-art solutions. Also, the performance of AutoIDS against changing the anomalous traffic ratio to the normal traffic has been evaluated. Finally, We have investigated the performance of each detector individually, in the second part.

Comparison with the previous methods: We have compared AutoIDS with methods reporting the result of 2-class performance and uses the same dataset for the test phase,

[2]https://keras.io/
[3]https://colab.research.google.com

TABLE III
AUTOIDS ACCURACY COMPARISON WHEN THE TRAINING PROCESS IS PERFORMED ON KDDTRAIN^{+} AND THE TEST PROCEDURE IS DONE ON KDDTEST^{+}.

Method	Supervised	*Accuracy*(%)
RNN-IDS [50]	✓	83.28
DCNN [51]	✓	85.00
Sparse AE and MLP [36]	✓	88.39
Random Tree [52]	✓	88.46
LSTM [51]	✓	89.00
Random Tree and NBTree [52]	✓	89.24
AE [53]		88.28
De-noising AE [53]		88.65
Ours (AutoIDS)		**90.17**

TABLE IV
DATA DISTRIBUTION OF TRAINING, VALIDATION AND TEST SETS.

	KDDTrain^{+}		
Class	**Training**	**Validation**	**Test**
Normal	53873	6735	6735
Anomaly	—	6735	6735

i.e., KDDTest^{+}. Table III shows that AutoIDS outperforms the state-of-the-art anomaly detection based methods for IDS, in terms of accuracy. Note that AutoIDS has a better performance even in comparison with the methods using supervised learning approach.

Regarding Table III and the characteristics of NSL-KDD dataset, AutoIDS is more efficient for detecting unseen network attacks. This superior performance has originated from the feature of generalization. To showcase the generality of AutoIDS, both training and test process are performed on KDDTrain^{+}. In this case, the test procedure is done on anomalies which are most probably available in the training set. Having learned all types of anomalous traffic, the neural network gets familiar with their concept and then distinguishes anomalies from normal flows more precisely. In this experiment, KDDTrain^{+} is divide into three subsets, training set, validation set and test set. The data distribution of these subsets is represented in Table IV. Note that flows are selected randomly for each subset.

Table V shows the accuracy of AutoIDS compared to the other state-of-the-art approaches in Table III which also report the accuracy of their proposed solution on KDDTrain^{+}. In fact, we need the other approaches evaluating their proposed solution using both KDDTrain^{+} and KDDTest^{+}, otherwise, the generalization is not understood properly. Although AutoIDS has a better performance compared to the other semi-supervised approaches [53], it is less accurate than supervised ones. Supervised methods learn the concept of available intrusions in the training set. Obviously, when they face same attack types in the test phase, separating incoming flows is done more accurately. Table III along with Table V confirms that AutoIDS outperforms the other solutions against new and unknown attacks indicating the generalization of this method.

TABLE V
AUTOIDS PERFORMANCE COMPARISON WITH THE OTHER STATE-OF-THE-ART SOLUTIONS WHEN BOTH TRAINING AND TEST PHASES ARE PERFORMED ON KDDTRAIN^{+}. NOTE THAT KDDTEST^{+} IS NOT USED.

Method	*ACC*(%)	*PR*(%)	*RE*(%)	*FS*(%)
AE [53]	93.62	91.39	96.33	93.80
De-noising AE [53]	94.35	94.26	94.43	94.35
Ours (AutoIDS) [53]	96.45	**95.56**	**97.43**	96.49
Sparse AE and MLP [36]	98.30	—	—	**98.84**
RNN-IDS [50]	**98.81**	—	—	—

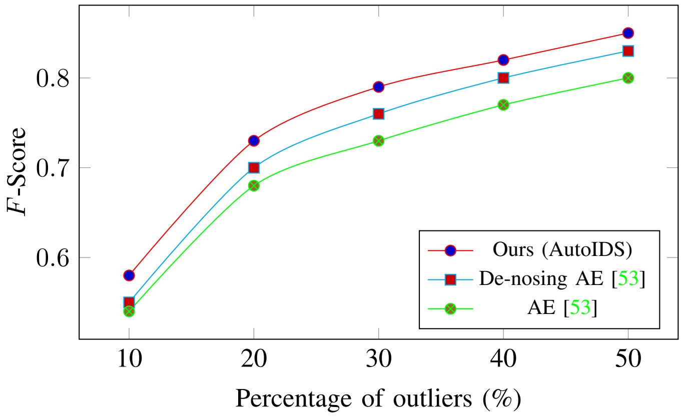

Fig. 5. Comparisons of f-scores for different percentages of anomalous flows involved in the experiment.

Generally, in the real world computer networks, abnormal traffic constitutes a lower portion of total traffic. Hereupon, we have evaluated AutoIDS using the f-score measure with different test datasets when NSL-KDD is used. The amount of anomalies is considered from 10 percent of the normal to 50 percent, with the step of 10. Fig. 5 shows that AutoIDS outperforms the other two semi-supervised solutions in [53].

Ablation Study: Both $\mathcal{D}_1$ and $\mathcal{D}_2$ can be used for anomaly detection separately, but exploiting both of them results in gaining more accuracy. Table VI shows the performance of AutoIDS when $\mathcal{D}_1$ and $\mathcal{D}_2$ work based on sparsity ($\mathcal{S}$) and reconstruction error ($\mathcal{R}$) respectively, i.e., $\mathcal{D}_1^{[\mathcal{S}]} \rightarrow \mathcal{D}_2^{[\mathcal{R}]}$. We have also reported the performance of each detector individually as an efficient detector when they both work based on reconstruction error, i.e., $\mathcal{D}_1^{[\mathcal{R}]}$ and $\mathcal{D}_2^{[\mathcal{R}]}$.

Table VI shows that AutoIDS ($\mathcal{D}_1^{[\mathcal{S}]} \rightarrow \mathcal{D}_2^{[\mathcal{R}]}$) outperforms $\mathcal{D}_1^{[\mathcal{R}]}$ and $\mathcal{D}_2^{[\mathcal{R}]}$ in terms of accuracy, recall and f-score. AutoIDS processes incoming traffic in two steps. We have concentrate on simple normal and anomalies in the step one, while the decision about more complex traffic is made in the next step. Hence, AutoIDS performs more precise compared to each detector separately.

The computational cost for processing a packet flow is also taken into account due to its significance for proposing an appropriate solution. In fact, the time of processing an incoming flow should not be the bottleneck as much as

TABLE VI
AUTOIDS PERFORMANCE COMPARISON WITH EACH DETECTOR INDIVIDUALLY WHEN THE TRAINING PROCESS IS PERFORMED ON KDDTRAIN$^+$ AND THE TEST PROCEDURE IS DONE ON KDDTEST$^+$

	AutoIDS ($\mathcal{D}_1^{[\mathcal{S}]} \to \mathcal{D}_2^{[\mathcal{R}]}$)	$\mathcal{D}_2^{[\mathcal{R}]}$	$\mathcal{D}_1^{[\mathcal{R}]}$
ACC (%)	**<u>90.17</u>**	89.14	87.50
PR (%)	90.80	91.31	**<u>91.72</u>**
RE (%)	**<u>92.05</u>**	89.43	85.79
FS (%)	**<u>91.42</u>**	90.36	88.65
FPR (%)	12.33	11.25	**<u>10.24</u>**

TABLE VII
AVERAGE TEST TIME COMPARISON PER EACH INCOMING FLOW FOR AUTOIDS AND EACH OF DETECTORS INDIVIDUALLY WHEN THE TRAINING PROCESS IS PERFORMED ON KDDTRAIN$^+$ AND THE TEST PROCEDURE IS DONE ON KDDTEST$^+$

AutoIDS ($\mathcal{D}_1^{[\mathcal{S}]} \to \mathcal{D}_2^{[\mathcal{R}]}$)	**<u>137</u>** μsec
$\mathcal{D}_2^{[\mathcal{R}]}$	195 μsec
$\mathcal{D}_1^{[\mathcal{R}]}$	202 μsec

possible. Once again we have compared these approaches, but this time in terms of time complexity in the test phase. Table VII confirms that AutoIDS, i.e., $\mathcal{D}_1^{[\mathcal{S}]} \to \mathcal{D}_2^{[\mathcal{R}]}$, can effectively detect anomalies and it is faster than each of detectors by a considerable margin. Since in AutoIDS all of the incoming flows are just encoded by $\mathcal{D}_1$ and also a large amount of normal and abnormal traffic are detected in the first step, our proposed method achieve lower time complexity in comparison with the other ones. In fact, all of incoming flows are encoded and then decoded when each detector is used individually and this way of approaching the problem makes them slower.

Table. VI and Table. VII confirm the superiority of the idea of composing AE and sparse AE which leads to improving the performance in terms of accuracy and computational cost.

V. CONCLUSION

In this paper, we have proposed a novel yet efficient semi-supervised method based on AEs which is called AutoIDS. AutoIDS takes advantage of cascading two encoder-decoder neural networks forced to provide a compressed and a sparse representation from normal traffic. These neural networks are known as AE and sparse AE, respectively. In case that these neural networks fail in providing the desire representations for an incoming packet flow, it does not follow the concept of normal traffic. The first detector, i.e., sparse AE, is faster while the second one, i.e., AE, makes a decision about the incoming packet flows more accurately. It is worth mentioning that, a large number of incoming packet flows are processed merely by the first detector. AutoIDS has been evaluated comprehensively on the NSL-KDD dataset. Results confirm the superiority of our method in comparison with the other state-of-the-art approaches.

REFERENCES

[1] W. Stallings, *Cryptography and Network Security, 4/E*. Pearson Education India, 2006.

[2] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.

[3] J. Cui, J. Long, E. Min, Q. Liu, and Q. Li, "Comparative study of cnn and rnn for deep learning based intrusion detection system," in *Cloud Computing and Security*, X. Sun, Z. Pan, and E. Bertino, Eds. Springer International Publishing, 2018, pp. 159–170.

[4] R. Chapaneri and S. Shah, "A comprehensive survey of machine learning-based network intrusion detection," in *Smart Intelligent Computing and Applications*, S. C. Satapathy, V. Bhateja, and S. Das, Eds. Springer Singapore, 2019, pp. 345–356.

[5] P. Viola, M. Jones *et al.*, "Rapid object detection using a boosted cascade of simple features," *CVPR (1)*, vol. 1, no. 511-518, p. 3, 2001.

[6] R. Polikar, "Ensemble learning," in *Ensemble machine learning*. Springer, 2012, pp. 1–34.

[7] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," *Computer Vision and Image Understanding*, vol. 172, pp. 88–97, 2018.

[8] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette, "Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1992–2004, 2017.

[9] M. Sabokrou, M. Fathy, Z. Moayed, and R. Klette, "Fast and accurate detection and localization of abnormal behavior in crowded scenes," *Machine Vision and Applications*, vol. 28, no. 8, pp. 965–985, 2017.

[10] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, pp. 2579–2605, 2008.

[11] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, 2019.

[12] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Communications Surveys and Tutorials*, 2018.

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

[14] Y. Liao and V. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection11an earlier version of this paper is to appear in the proceedings of the 11th usenix security symposium, san francisco, ca, august 2002," *Computers and Security*, vol. 21, no. 5, pp. 439 – 448, 2002.

[15] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, vol. 2. IEEE, 2002, pp. 1702–1707.

[16] S. Mukkamala and A. H. Sung, "Detecting denial of service attacks using support vector machines," in *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1231–1236.

[17] N. Chand, P. Mishra, C. R. Krishna, E. S. Pilli, and M. C. Govil, "A comparative analysis of svm and its stacking with other classification algorithm for intrusion detection," in *Advances in Computing, Communication, and Automation (ICACCA)(Spring), International Conference on*. IEEE, 2016, pp. 1–6.

[18] D. Heckerman, "Bayesian networks for data mining," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 79–119, Mar 1997.

[19] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip *et al.*, "Top 10 algorithms in data mining," *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008.

[20] C. Gates, J. J. McNutt, J. B. Kadane, and M. I. Kellner, "Scan detection on very large networks using logistic regression modeling," in *null*. IEEE, 2006, pp. 402–408.

[21] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[22] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[23] W. W. Cohen, “Fast effective rule induction,” in *Twelfth International Conference on Machine Learning*. Morgan Kaufmann, 1995, pp. 115–123.

[24] R. Holte, “Very simple classification rules perform well on most commonly used datasets,” *Machine Learning*, vol. 11, pp. 63–91, 1993.

[25] P. Tao, Z. Sun, and Z. Sun, “An improved intrusion detection algorithm based on ga and svm,” *IEEE Access*, vol. 6, pp. 13 624–13 631, 2018.

[26] J. Ling and C. Wu, “Feature selection and deep learning based approach for network intrusion detection,” in *3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)*. Atlantis Press, 2019.

[27] P. Domingos and M. Pazzani, “On the optimality of the simple bayesian classifier under zero-one loss,” *Machine learning*, vol. 29, no. 2-3, pp. 103–130, 1997.

[28] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, “Fuzziness based semi-supervised learning approach for intrusion detection system,” *Information Sciences*, vol. 378, pp. 484–497, 2017.

[29] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, “Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model,” *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.

[30] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[31] Tin Kam Ho, “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 278–282.

[32] J. R. Quinlan, “Simplifying decision trees,” *International journal of man-machine studies*, vol. 27, no. 3, pp. 221–234, 1987.

[33] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” in *Thirteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 1996, pp. 148–156.

[34] C. Sammut and G. I. Webb, Eds., *Decision Stump*. Springer US, 2010, pp. 262–263.

[35] B. Dong and X. Wang, “Comparison deep learning method to traditional methods using for network intrusion detection,” in *Proc. IEEE ICCSN*, 2016, pp. 581–585.

[36] A. Javaid, Q. Niyaz, W. Sun, and 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)*, ser. BICT’15. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26.

[37] F. Farahnakian and J. Heikkonen, “A deep auto-encoder based approach for intrusion detection system,” in *Advanced Communication Technology (ICACT), 2018 20th International Conference on*. IEEE, 2018, pp. 178–183.

[38] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[39] D. Papamartzivanos, F. Gmez Mrmol, and G. Kambourakis, “Introducing deep learning self-adaptive misuse network intrusion detection systems,” *IEEE Access*, vol. 7, pp. 13 546–13 560, 2019.

[40] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, “Self-taught learning: transfer learning from unlabeled data,” in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 759–766.

[41] J. O. Kephart and D. M. Chess, “The vision of autonomic computing,” *Computer*, no. 1, pp. 41–50, 2003.

[42] B. Mohammadi and M. Sabokrou, “End-to-end adversarial learning for intrusion detection in computer networks,” *LCN*, 2019.

[43] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, “Adversarially learned one-class classifier for novelty detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3379–3388.

[44] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun, “Learning convolutional feature hierarchies for visual recognition,” in *Advances in neural information processing systems*, 2010, pp. 1090–1098.

[45] M. Sabokrou, M. Fathy, and M. Hoseini, “Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder,” *Electronics Letters*, vol. 52, no. 13, pp. 1122–1124, 2016.

[46] M. Sabokrou, M. Pourreza, M. Fayyaz, R. Entezari, M. Fathy, J. Gall, and E. Adeli, “Avid: Adversarial visual irregularity detection,” *ACCV2018*, 2018.

[47] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[48] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, “Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation,” in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX’00*, vol. 2, 2000, pp. 12–26.

[49] S. J. Stolfo, W. Fan, A. Prodromidis, P. K. Chan, and W. Lee, “Cost-sensitive modeling for fraud and intrusion detection: Results from the jam project,” in *In Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, 2000.

[50] C. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[51] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, “Enhanced network anomaly detection based on deep neural networks,” *IEEE Access*, vol. 6, pp. 48 231–48 246, 2018.

[52] J. Kevric, S. Jukic, and A. Subasi, “An effective combining classifier approach using tree algorithms for network intrusion detection,” *Neural Computing and Applications*, vol. 28, no. 1, pp. 1051–1058, 2017.

[53] R. C. Aygun and A. G. Yavuz, “Network anomaly detection with stochastically improved autoencoder based models,” in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2017, pp. 193–198.