

Received May 12, 2020, accepted June 1, 2020, date of publication June 10, 2020, date of current version June 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3001350

# Anomaly-Based Intrusion Detection From Network Flow Features Using Variational Autoencoder

SULTAN ZAVRAK<sup>1,2</sup> AND MURAT İSKEFIYELI<sup>3</sup>, (Associate Member, IEEE)

<sup>1</sup>Department of Computer and Information Engineering, Sakarya University, 54187 Sakarya, Turkey

<sup>2</sup>Department of Computer Engineering, Düzce University, 81620 Düzce, Turkey

<sup>3</sup>Department of Computer Engineering, Sakarya University, 54187 Sakarya, Turkey

Corresponding author: Sultan Zavrak (sultanzavrak@duzce.edu.tr)

The work of Sultan Zavrak was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) through the 2211/C Ph.D. Scholarship Programme for Priority Areas.

**ABSTRACT** The rapid increase in network traffic has recently led to the importance of flow-based intrusion detection systems processing a small amount of traffic data. Furthermore, anomaly-based methods, which can identify unknown attacks are also integrated into these systems. In this study, the focus is concentrated on the detection of anomalous network traffic (or intrusions) from flow-based data using unsupervised deep learning methods with semi-supervised learning approach. More specifically, Autoencoder and Variational Autoencoder methods were employed to identify unknown attacks using flow features. In the experiments carried out, the flow-based features extracted out of network traffic data, including typical and different types of attacks, were used. The Receiver Operating Characteristics (ROC) and the area under ROC curve, resulting from these methods were calculated and compared with One-Class Support Vector Machine. The ROC curves were examined in detail to analyze the performance of the methods in various threshold values. The experimental results show that Variational Autoencoder performs, for the most part, better than Autoencoder and One-Class Support Vector Machine.


**INDEX TERMS** Flow anomaly detection, intrusion detection, deep learning, variational autoencoder, semi-supervised learning.

## I. INTRODUCTION

With the rapid growth of the Internet, cyber-attacks on networks and computer systems have also increased expeditiously. As a precaution against these attacks, Intrusion Detection Systems (IDS) are deployed in networking systems. Although IDSs have gained considerable significance in counteracting increased network attacks, payload-based IDSs lack the scalability due to high-speed of networks and increase in network traffic. Consequently, flow-based detection approaches have recently been a potential candidate for IDS. Flow-based IDSs are preferred over traditional IDSs that are based on deep packet inspection, for two reasons: firstly, they process noticeably smaller amount of data, and secondly, the flow-data is easily gathered out of network forwarding devices that utilize standard protocols (such as Cisco NetFlow, IETF IPFIX), without installing additional software and data is gathered from each computer

on the network [1]. Intrusion detection approaches are basically divided into two categories [1]: misuse-based and anomaly-based. Misuse-based techniques work in a way that the signatures of previously known attacks are compared. This technique, which works well against known attacks, is insufficient in the detection of the unknown attacks. Conversely, anomaly-based intrusion detection methods have the ability to identify unknown or zero-day attacks. Because of the extreme change in normal behavior of the network, it is hard to obtain the exact definition of normal behavior. Therefore, false alarms of anomaly-based techniques are higher than that of others.

Recently, deep learning methods have also been applied in network intrusion detection systems, as it has been seen that the deep learning methods could successfully solve many problems faced in the research areas such as text classification, object recognition and image classification etc. In this active research area, the studies using deep learning approaches mostly focus on dimensionality reduction [2]–[6] and anomaly-based intrusion detection [7]–[10].

The associate editor coordinating the review of this manuscript and approving it for publication was Kim-Kwang Raymond Choo .

The KDDCUP99 dataset [11], which includes packet-based features, too, is used to evaluate the methods in the studies [6], [12]. The NSL-KDD dataset [13], which is a revised version of KDDCUP99, is used for evaluating the methods proposed in the studies [3], [4], [6]–[8]. Main drawbacks of these studies could be listed as follows: a) These studies focus on the detection of intrusions in content-based features. b) Dataset used is very outdated and does not reflect the real network traffic [14].

As the flow-based data contains less information on the network traffic compared to payload-based data, it is much harder to detect both known and unknown attacks. In this study, the goal is to detect anomalous network traffic (or intrusions) from flow-based data, which also contain statistical properties of the flows using deep learning methods. Furthermore, Autoencoder (AE) and Variational Autoencoder (VAE) methods are employed to detect unknown attacks, which mean the attacks are not used in training phase by using the flow features extracted from network traffic.

The main contributions of the study are summarized as follow:

- This study concentrates on detection of network attacks from flow-based features, based on anomaly-based approach.
- AE and VAE, which are unsupervised deep learning methods, are employed together with OCSVM as anomaly detectors and they are trained in a semi-supervised learning manner. In addition, unlike the previous studies mentioned above, this study is unique as it uses deep learning methods for detecting intrusions based on flow-based data.
- It is shown that VAE-based anomaly detection system performs much better compared to others based on the detailed discussion of ROC and AUC results.

The article is organized as follows. In the next section, the studies carried out on flow-based intrusion detection in the literature are summarized. In the third section, the theoretical information on the techniques and evaluation metrics are provided. The experimental methodology and results are presented in the fourth section. The final remarks are provided in the last section.

## II. RELATED STUDIES

The flow-based intrusion detection is on the rise and research made in this field are gathering pace. In recent years, numerous methods have been proposed, which used flow data for identifying intrusions. In this section, a review of recent trends and particular state-of-the-art algorithms that detect intrusions from flow-based data are summarized. These include Artificial Neural Networks (ANN), Support Vector Machines (SVM), K-Nearest Neighbor (KNN), Decision Trees, clustering and statistical techniques. A more comprehensive and detailed analysis of flow-based intrusion detection can be found in studies of [15] and [16].

The objective of ANNs is to model the human brain, by mimicking neurons, which are small interconnected input units. Each neuron in ANN participates in decision making, and the results are combined. Behaviors of users are modelled by ANNs to find a way to detect anomalies. Numerous ANNs used for anomaly-based IDSs were discussed by Beghdad [17]. Sui *et al.* [18] proposed an anomaly detection system that used a back-propagation neural network classifier and statistical feature vectors. They considered three scenarios of resource depletion, DoS attacks, bandwidth attack and a combination of bandwidth attack and resource depletion using network flow records with DoS attacks. Tran *et al.* [19] proposed a hybrid detection engine, which used block-based neural network (BBNN) as the detection method. In order to generate a real-time IDS, which was supplied by NetFlow data, it was added in a high-frequency FPGA board. Abuadlla *et al.* [1], proposed an IDS to detect and classify certain intrusions in flow-based data, which consisted of two phases. In the first phase, significant changes are monitored to identify potential attacks. In the second phase, if an attack is known, multi-layer and radial basis function networks are used to classify the attack. Jadidi *et al.* [20] proposed a method that was based on Multi-Layer Perceptron (MLP) in order to detect abnormal traffic in flow-based data. The interconnection weights of MLP are optimized by using Cuckoo and particle swarm optimization with a gravitational search algorithm (PSOGSA). Mirsky *et al.* [21] proposed Kitsune, which was an online anomaly detection system that identified the attacks on a local network by employing a group of ANNs named AEs, to cooperatively distinguish between normal and abnormal traffic patterns with a performance comparable to offline anomaly detectors. Marir *et al.* [22] presented a novel distributed method for identifying abnormal behavior utilizing a group of multi-layer SVMs together with a deep feature extraction in largescale networks. In the approach proposed, a non-linear dimensionality reduction was initially performed with a distributed deep belief networks on network traffic data and then the features extracted were provided as inputs to the multi-layer group of SVMs which were constructed through the iterative reduce paradigm based on Spark. Vinayakumar *et al.* [23] explored a deep neural network (DNN) to create a useful and flexible IDS, named “scale-hybrid-IDS-AlertNet”, and to identify unforeseen and unpredictable intrusions via supervised learning approach. They selected the network topologies and optimal network parameters for DNNs by applying various hyperparameter settings with KDDCup99 and the best performed DNN model was also applied on other contents- and flow-based public datasets to carry out benchmarks.

The SVM is a classification method, which transforms an n-dimensional input data into classes by generating vectors in the space. In the research area of intrusion detection, SVM is the method preferred as it provides results in lower false positive rates and higher accuracies [74]. Winter *et al.* [24] proposed an inductive network IDS, which functioned on network flows and used OCSVMs for analysis.

Instead of benign flows, the IDS proposed was trained with malicious data, as opposed to the previous approaches. Wagner *et al.* [25] presented an anomaly detection method by processing large volumes of Netflow records based on SVM. The method in which the quantitative and contextual information of Netflow data was considered was carried out by feeding the Netflow records into kernel function and forwarding the calculated results to an OCSVM. Umer *et al.* [26] proposed an intrusion detection model, which handled the flow data. The two-stage model developed used OCSVM method in order to identify malicious flows efficiently and then forwarded the malicious traffic to the second phase of the detection process. The detailed analysis of malicious flows was conducted in the second phase.

K-Nearest Neighbor (KNN) takes into account the knowledge of adjacent points to perform classification in the example given. Shubair *et al.* [27] suggested a flow-based IDS exploiting the benefits of KNN method with the combination of fuzzy logic. The study used Least Mean Square method to perform error reduction and KNN to pick out the best matching class and fuzzy logic for selecting the flow class label. Costa *et al.* [28] proposed an intrusion detection method employing Optimum-path forest clustering (OPFC), which was a KNN graph utilizing probability density function for weighing of nodes. The authors made use of enhanced nature inspired techniques (Gravitational Search, Bat Algorithm, Particle Swarm Optimization, and Harmony search) to decide the value of  $k$  in the optimization of OPFC.

Clustering methods are used to detect unique and beneficial patterns in the dataset. The aggregation of the similar examples in different clusters is performed using these patterns. In order to detect network flow anomalies, Lakhina *et al.* [29] employed the clustering methods by analyzing the feature distributions. Casas *et al.* [30] presented an anomaly-based IDS, which collected packets from the network and gathered these packets together into flows randomly by employing multiple unsupervised clustering techniques. In this study, a change detection algorithm, which utilized sub-space and density-based clustering that produced subsets of data in each sub-space, was employed to take apart the malicious flows. Hosseinpour *et al.* [31] proposed a distributed IDS based on unsupervised clustering combined with Artificial Immune System. The method proposed used DBSCAN clustering method to label traffic data as malicious or non-malicious and made available real-time and online data in order to train the immune response detectors located around hosts of the networks. The Ward Clustering approach was recommended by Satoh *et al.* [32] to identify simple and malicious attacks from the SSH dictionary. The detection process was performed based on “the existence of a connection protocol” and “the inter-arrival time of an authentication-packet and the next” by identifying “the transition point of each sub-protocol” through flow features.

Decision Trees (DTs) generate a tree model by building rules based on the attribute value of each node of the

tree. Thaseen and Kumar [33] discussed application of different DT-based classification algorithms for intrusion detection. Zhao *et al.* [34] proposed a simple and efficient flow-based approach to identify both known and novel botnets by employing a DT approach with Reduced Error Pruning method to build a model for classifying the botnets. Haddadi *et al.* [35] suggested an alternative approach for identifying the behaviors of botnets based on genetic programming and DTs. The suggested method used two different sets, which were common flow attributes and TCF flags attributes, extracted from the datasets. Stevanovic and Pedersen [36] presented an effective botnet detection approach using flow records of a 39-feature set employing a collection of supervised machine learning methods. According to results, the overall best performance was achieved by Random Forest method.

The statistical methods used in literature can be summarized as follows. Kanda *et al.* [37] suggested an anomaly detection method, called ADMIRE, based on random projection and Principal Component Analysis (PCA) to identify abnormal network flows. The assessment was carried out using the traffic traces from a transpacific connection. Haghghat and Li [38] proposed a novel entropy-based method, called as Edmund, to identify abnormal behavior and mitigate network intrusions by using NetFlow traffic data.

### III. BACKGROUND

#### A. AUTOENCODER

Autoencoder (AE) [39], [40] is a neural network method, which has an operating logic that trains the input vectors to reconstruct as output vectors with an unsupervised approach [41]. Its architecture is basically constructed by an encoder and a decoder. A single layer of AE has an encoder and a decoder as in (1) and (2), respectively. In this context,  $\sigma$  is the nonlinear transformation function and  $b$  and  $W$  are called the bias and the weight of the neural network, respectively [42].

$$h = \sigma(W_{xh}x + b_{xh}) \quad (1)$$

$$z = \sigma(W_{hx}h + b_{hx}) \quad (2)$$

$$r = \|x - z\| \quad (3)$$

By using an affine mapping resulting in a nonlinearity, the transformation of the input vector  $x$  to a hidden representation  $h$  is performed using the encoder. The transformation operation is applied to hidden representation  $h$  to reconstruct the initial input space using a decoder. The reconstruction error  $r$  is obtained by taking the difference between the reconstructed vector  $z$  and the original input vector  $x$ . In order to minimize reconstruction error  $r$ , unsupervised training procedure is accomplished in AE. The flow chart of the AE training algorithm can be illustrated as in Fig. 1.

AE-based anomaly detection is accomplished using the reconstruction error (RE) as the anomaly score. The input data with high RE are assumed to be anomalies. The training of AE is performed by feeding the network input with only

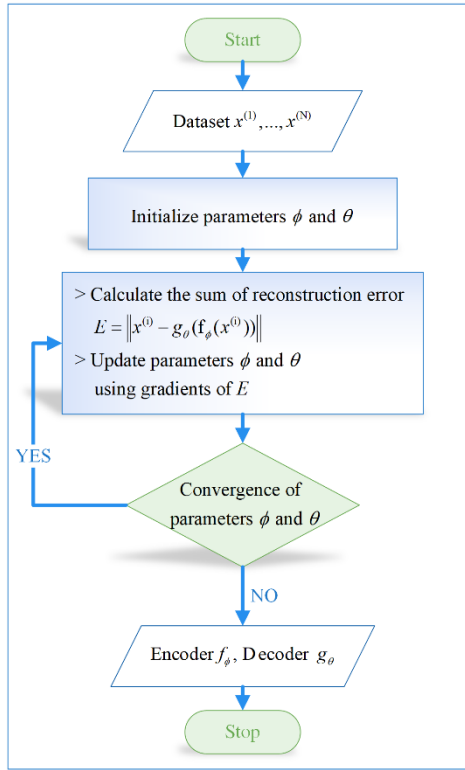


FIGURE 1. The flow chart of AE training algorithm.

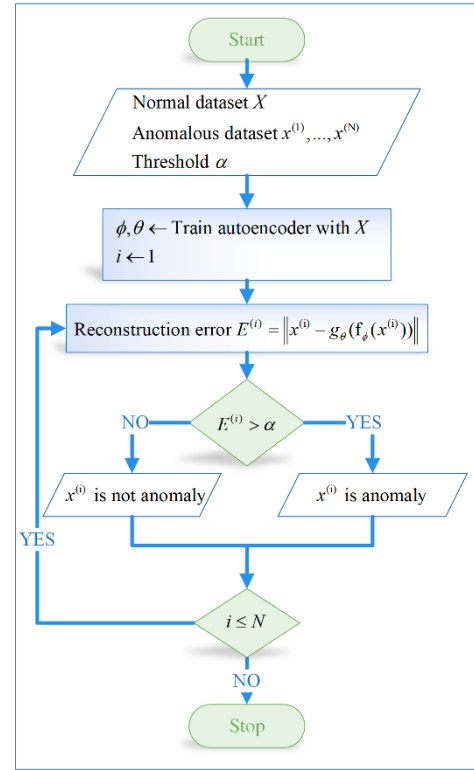


FIGURE 2. The flow chart of AE-based anomaly detection algorithm.

normal examples. The trained AE model will reconstruct normal input data with very low RE, where it is unsuccessful to do so with anomalous data it has not confronted before. The flow chart of AE-based anomaly detection algorithm can be illustrated as in Figure 2.

**B. VARIATIONAL AUTOENCODER**

Variational Autoencoder (VAE) [43] is defined as a directed probabilistic graphical model, which is obtained by approximation of an artificial neural network to its posterior [42]. In VAE, the latent variable  $z$  in which the generative process begins, is the highest layer of the graphical model. The complicated procedure of data generation, which leads to the data  $x$ , is represented by  $g(z)$  that is modeled in the formation of an artificial neural network. Because the marginal likelihood is intractable, the variational lower boundary of the marginal likelihood of input data is the objective function of VAE. The marginal likelihood is obtained by summing of the marginal likelihood of distinct data points in (4). Equation (5) is obtained if the marginal likelihood of distinct data points are reformulated [42].

$$\log p_\theta(x^{(1)}, \dots, x^{(N)}) = \sum_{i=1}^N \log p_\theta(x^{(i)}) \quad (4)$$

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(\theta, \phi; x^{(i)}) \quad (5)$$

$$= E_{q_\phi(z|x^{(i)})} [-\log q_\phi(z|x) + \log p_\theta(x|z)] \quad (6)$$

$$= -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + E_{q_\phi(z|x^{(i)})}[\log p_\theta(x|z)] \quad (7)$$

In Equation (7),  $D_{KL}$  is the Kullback–Leibler divergence between the approximate posterior and the prior of the latent variable  $z$ . The likelihood of the input data  $x$  given the latent variable  $z$  is represented as  $p_\theta(x|z)$ .

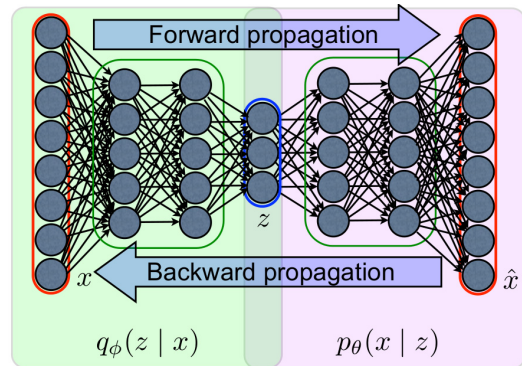


FIGURE 3. VAE architecture [42].

The parameters of the approximate posterior  $q_\phi(z|x)$  is achieved with a neural network by VAE. The directed probabilistic graphical model  $p_\theta(x|z)$  is the decoder and the approximate posterior  $q_\phi(z|x)$  is the encoder as illustrated in Fig. 3. In this context, it must be emphasized that the purpose of VAE is to model the distribution parameters instead of actual value. That is,  $f(x, \phi)$  in the encoder produces the parameter of the approximate posterior  $q_\phi(z|x)$  and to achieve the actual value of the latent variable  $z$ ,

sampling from  $q(z, f(x, \phi))$  is needed. The common choice is the isotropic normal for the distribution of latent variable  $z$ , which are  $p_\theta(z)$  and  $q_\phi(z|x)$ , since the relationship included in variables in latent variable space is expected to be a lot simpler than the original data space. The distributions of the likelihood  $p_\theta(x|z)$  vary according to the nature of the data. More specifically, multivariate Gaussian distribution is applied when the input data is in continuous form. If it is binary, Bernoulli distribution is used [42]. The flow chart of VAE training algorithm can be illustrated as in Fig. 4.

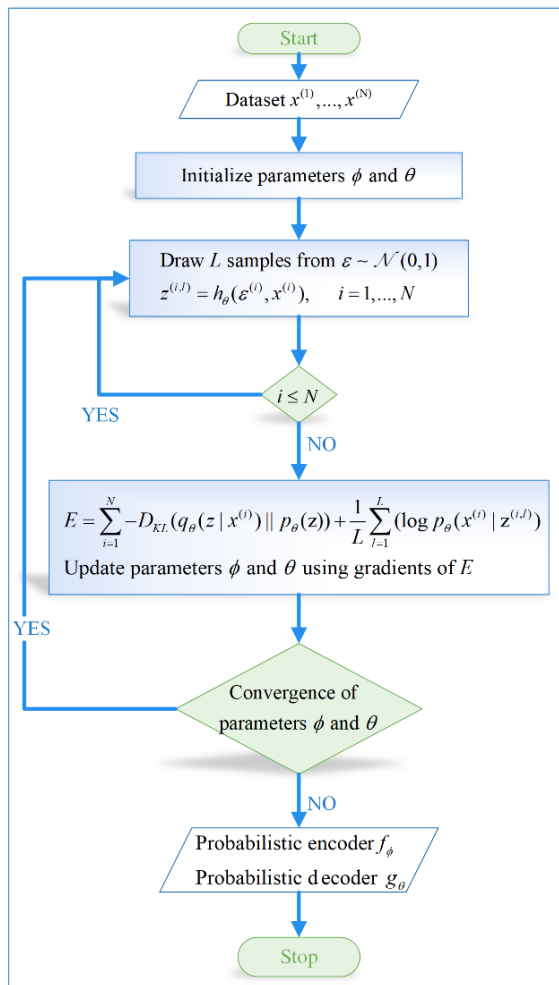


FIGURE 4. The flow chart of VAE training algorithm.

The training of VAE is executed by using the backpropagation algorithm [42]. The second term on (7) is computed using Monte Carlo gradient techniques in conjunction with a reparameterization approach, which employs a random variable from a standard normal distribution rather than a random variable from the original distribution. The random variable  $z \sim q_\phi(z|x)$  is reparametrized by a deterministic transformation  $h_\phi(\epsilon, x)$  where  $\epsilon$  is from a standard normal distribution.

$$\tilde{z} = h_\phi(\epsilon, x) \text{ with } \epsilon \sim \mathcal{N}(0, 1) \quad (8)$$

The reparameterization operation is supposed to guarantee that  $\tilde{z}$  follows the distribution of  $q_\phi(z|x)$ .

The anomaly detection task is performed in a semi-supervised way, meaning that just normal data examples are used to train VAE [42]. The probabilistic decoder  $g_\theta$  and encoder  $f_\phi$  both parameterizes an isotropic normal distribution in the original input variable space and the latent variable space, respectively. The testing process is carried out by selecting several examples from the probabilistic encoder of the trained VAE model. The probability of the original data produced from the distribution is computed using mean and variance parameters, which are generated by the probabilistic decoder or each example from the encoder. The average probability, which is also called the reconstruction probability (RP), is used as an anomaly score. The RP is the Monte Carlo estimation of

$$E_{q_\phi(z|x)} [\log p_\theta(x|z)] \quad (9)$$

The stochastic latent variables, which produce the parameters of the original input variable distribution are utilized to compute the RP. This is fundamentally equivalent to the probability of the data being produced from certain latent variables taken out of the approximate posterior distribution.

RP computed in VAE differs from RE calculated in AE in some ways [42]. First of all, while latent variables are expressed as deterministic mappings in AE, they are defined as stochastic variables in VAE. The variability of the latent space can be taken into consideration from the sampling process due to the fact that VAE employs the probabilistic encoder to model the distribution of the latent variables instead of the latent variable itself. This extends the meaningfulness of VAE in comparison with AE since the variability can vary although anomaly data and normal data may possibly have an identical mean value. Secondly, reconstructions are stochastic

variables in VAE. RP not only takes into account the difference between the original input and the reconstruction, but also considers the variability of the reconstruction by taking into account the variance parameters. The selective sensitivity to reconstruction in accordance with variable variance is empowered by using this feature, which is not available in AE due to its deterministic nature. Thirdly, probability measures correspond to reconstructions. In AE based anomaly detection, anomaly scores are generated using REs. In that sense, the calculation of anomaly scores would be challenging if the input variables were heterogeneous because of the unavailability of a general objective technique to determine the suitable weights, which vary depending on the data. In addition to this, the determination of a proper and objective threshold for RE is a problematic process. On the other hand, as the probability distribution of every single variable enables them to be independently computed by its individual variability, the computation of the RP doesn't need weighing of the RE of the heterogeneous data. Therefore, it can be concluded that the determination of the

threshold value of the RP can be performed in significantly more objective and readily comprehensible way than that of the RE.

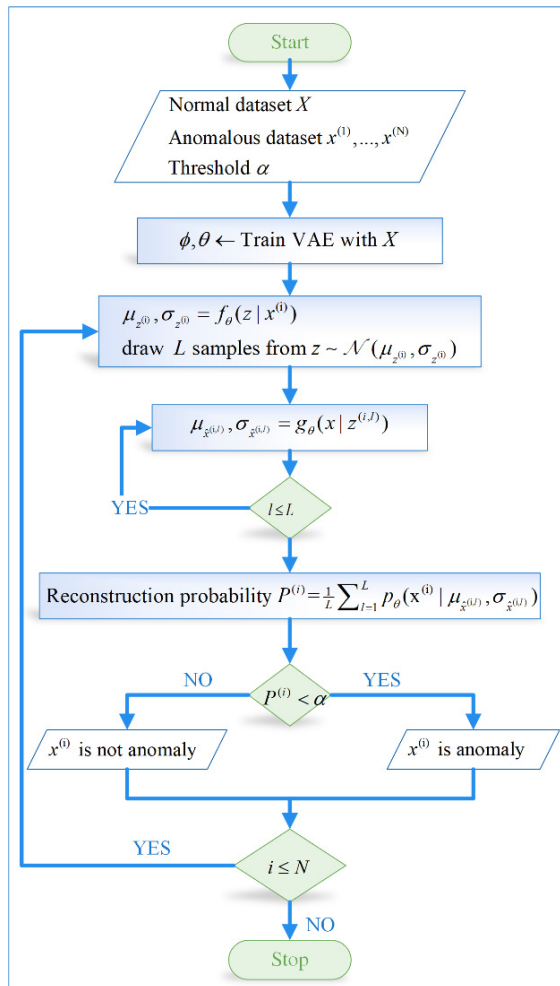


FIGURE 5. The flow chart of VAE-based anomaly detection algorithm.

### C. ONE CLASS SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) was originally suggested by Boser *et al.* [44]. The primary objective of SVM is to find out the best machine for a given dataset. SVM accomplishes this objective by maximizing the correctness of the machine for a given training dataset. In addition to this, the ability of the machine is also maximized in order to classify the forthcoming testing datasets accurately. The best machine is discovered by utilizing a mathematical optimization method [45].

The SVM method is modified into a One-Class SVM (OCSVM) as explained in [46]. The dataset, which is given as an input to the algorithm in OCSVM, consists of examples belonging to two different classes, namely positive and negative. Noises in the positive samples in the dataset, which are also defined as “anomalies” or “outliers”, are employed as negative examples. The formulation of OCSVM

can be carried out as follows [46]:

$$f(x) = \begin{cases} +1, & \text{if } x \in S \\ -1, & \text{if } x \in \bar{S} \end{cases} \quad (10)$$

The working logic of the algorithm is as follows. The input data is first transformed into a feature space  $H$  by applying a suitable kernel function. Then, the algorithm proposed attempts to find a hyperplane to separate these transformed feature vectors from the origin by maximum margin [47].

Given a dataset  $(x_1, y_1), \dots, (x_N, y_N) \in \mathbb{R}^n \times \{\pm 1\}$ , let  $\phi: \mathbb{R}^n \rightarrow H$  be a kernel map, which maps the examples into  $H$ . Afterwards, in order to separate the dataset from the origin, following quadratic programming problem needs to be resolved:

$$\min \left( \frac{1}{2} \|\omega\|^2 + \frac{1}{\nu N} \sum_{i=1}^l \xi_i - \rho \right) \quad (11)$$

subject to

$$y_i(\omega \cdot \phi(x_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, N \quad (12)$$

The parameter  $\nu \in (0, 1)$  refers to the ratio of “anomalies” or “outliers” in the training dataset. It regulates the tradeoff in between, including most of the data in the region formed by the hyperplane and maximizes the distance from the origin. Some outliers are located outside the border by means of the slack variables  $\xi_i$ . The decision function  $f(x) = \text{sign}((\omega \cdot \phi(x)) - \rho)$  will be positive for most examples  $x_i$  in the training dataset. Since the separation of some datasets cannot be performed linearly, the kernel functions, such as the radial basis kernel [48] and polynomial kernel, are generally used in SVMs to transform the inputs to high dimensional space to achieve linear separability.

### D. EVALUATION METRICS

The evaluation of the proposed method should be performed using an appropriate metric. For a binary classification, the results can be separated into four groups [49], [50]: 1) True Positive (TP): Positive examples correctly classified; 2) False Negative (FN): A positive example misclassified; 3) False Positive (FP): A negative example misclassified; 4) True Negative (TN): A negative example correctly classified. Furthermore, subsequent metrics can be calculated from the previous ones [50]:

True Positive Rate (TPR): This metric corresponds to the ratio of all “correctly identified examples” to all “examples that should be identified”.

$$TPR = TP / (TP + FN) \quad (13)$$

False Positive Rate (FPR): This metric represents the ratio of the “number of misclassified negative examples” to the “total number of negative examples”.

$$FPR = FP / (FP + TN) \quad (14)$$

Receiver Operating Characteristics (ROC): The ROC curve [51], [52] is utilized as a standard criterion in the

assessment of classifiers in the case of a class imbalance problem encountered in the dataset [53]. The ROC curves attempt to accomplish skew insensitivity by providing the summary of the efficiency of classifiers on a range of TPRs and FPRs. The ROC curves can decide which percentage of examples will be properly classified for a certain FPR by assessing the trained models at different error values. The ROC curves offer an illustrated approach for deciding the efficiency of a classifier [53].

The area under the ROC curve (AUC) metric is utilized as a de facto standard in order to measure the efficiency of classifiers facing class imbalance problem. The reason behind is that AUC is not influenced by prior probabilities and the chosen threshold. In addition, it provides a single number to make comparison between classifiers. The AUC can be taken into consideration to estimate how frequently a random positive class example ranks higher than a negative class example after it is ordered by its classification probabilities. The AUC values are always bounded between 0 and 1. If AUC value is less than 0.5, it means that the classifier is unrealistic [54]. A rough classifying system can serve as a guidance to the test accuracy as follows [55]: i) Excellent (0.90-1), ii) Good (0.80-0.90), iii) Fair (0.70-0.80), iv) Poor (0.60-0.70), v) Fail (0.50-0.60).

#### IV. EXPERIMENTAL RESULTS

Semi-Supervised Learning (SSL) paradigm is employed as learning strategy. SSL covers several different settings including [56]: a) Semi-supervised classification: Its alternative name is classification with labelled and unlabeled data (or partially labelled data). This emerged as an extension to the supervised classification problem. b) Constrained clustering: This strategy extends unsupervised clustering by using training data consisting of unlabeled examples in addition to a number of “supervised information” on clusters. SSL strategy of constrained clustering was chosen as unlabeled data were easier to get for the intrusion detection system than labelled data by virtue of the fact that it required less knowledge, time, and effort. Moreover, this strategy is more appropriate for the unsupervised training nature of the detection methods utilized.

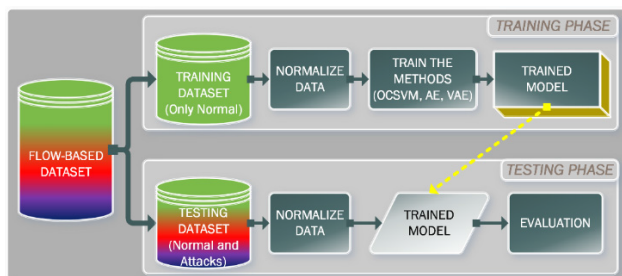


FIGURE 6. The semi-supervised learning strategy.

As seen in Fig. 6, the dataset is broken down into two parts as training and testing dataset. By applying the SSL

strategy, the labelled dataset, which contains only normal flow features is used in training phase to create the normal profile of network traffic, and in any case, the unlabeled dataset consisting of both normal and attack flow features is used in the testing phase. It is important to mention that labels in the testing dataset are taken into consideration for calculating the evaluation metrics in this study.

#### A. DATASET SETUP

In order to evaluate the methods used for intrusion detection, a proper dataset needs to be used. Intrusion detection methods that are referred to in the literature are evaluated through several datasets. Many researchers use KDDCUP99 and NSLKDD datasets to assess the models trained as mentioned in Section I. However, KDDCUP99 dataset suffers from the high redundancy problem in the training and test datasets. NSL-KDD dataset was generated from KDDCUP99 to resolve this issue. But, another issue, which is not addressed by NSL-KDD, is that this dataset is not a realistic representation of network traffic [14]. Additionally, although these datasets include some flow-based features, they mostly contain packet-based (content) features. As this study aims to detect the attacks from flow-based features, some of the candidate flow-based datasets are Kyoto 2006+ [57], CTU-13 [58], UNSW-NB15 [59], CIDD5-001 [60] and CICIDS2017 [61].

Kyoto 2006+ is a publicly accessible dataset, which encompasses real network traffic including numerous attacks performed against honeypots such as DoS, malware, backscatter, port scans, exploits and shellcode. But it contains merely a small amount of data and a small range of realistic normal user behavior with both statistical information and flow-based attributes. The CTU-13 was obtained in a campus network by characterizing 13 scenarios covering various botnet attacks and is accessible in the forms of packet, bidirectional flow, and unidirectional flow. The UNSW-NB15 was generated in a small emulated network over 31 hours by acquiring normal and malicious traffic in packet-based format. The data set, which is also offered in the form of flow dataset with extra features, comprises of nine distinct categories of attacks such as DoS, backdoors, fuzzers, worms or exploits, with predetermined separations for training and testing. The CIDD5-001 was acquired from a small emulated network by implementing the user behaviors of normal and malicious users by executing python scripts. It covers four weeks of unidirectional flow-based network traffic as well as network attacks including DoS, SSH brute force and port scan. The CICIDS2017 was produced in an emulated network environment within 5 days and covers network traffic in packet-based format as well as flow-based format including more than 80 extracted attributes with extra metadata information on IP addresses and a wide range of up-to-date attack types including FTP patator, SSH patator, DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye, Heartbleed, Brute force, XSS, SQL Injection, Infiltration, Bot, DDoS and Port Scan.

Among aforementioned datasets, CICIDS2017 was selected for evaluation purposes due to the fact that: 1) It includes various types of attacks, which had been carried out on networks recently, according to a McAfee report [62], [23], 2) It is up-to-date, 3) It is a labelled dataset consisting of flow-based features expanded by measuring some parameters statistically, 4) It possesses the characteristics of a real-time network traffic [23], 5) It is non-linearly separable [23]. Table 1 presents the number of flow examples in CICIDS2017 categorized by attack type and day. It should be noted that the only drawback of the dataset is that the distribution of the classes is highly imbalanced as specified in Table 1.

## B. MODEL SETUP

In this section, discussion is made on how the configuration of the AE, VAE and OCSVM methods is made and how the selection of their parameters is carried out to achieve the best anomaly-based intrusion detection model.

By virtue of the fact that the methods used in this study are parametrized, the performance of the models need to be determined by the optimal parameters. The cross-validation task cannot be performed in the optimization of hyperparameters as the training process is not executed using anomalous data [63]. Thus, the tuning of hyperparameters is performed by taking into consideration the recommendations proposed by Patterson and Gibson [64]. After that, the hyperparameters of AE and VAE are mostly configured using rule of thumb. The default values of the library are used in the configuration of OCSVM model.

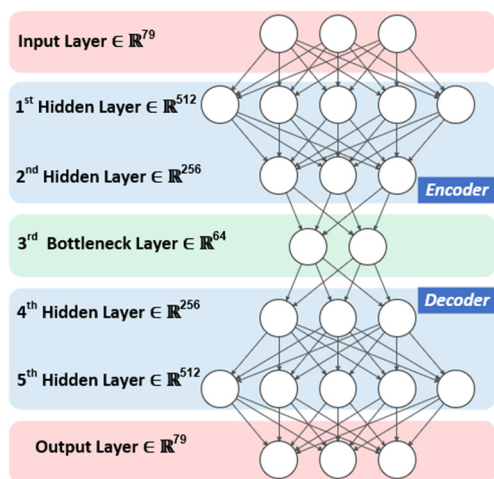


FIGURE 7. The architectural diagram of AE and VAE used in models.

A simple form of deep autoencoder architecture was chosen to perform experiments. The architectural diagram of AE and VAE is illustrated in Fig. 7. For both AE and VAE, the encoder has two hidden layers with 512 and 256 dimensions and the decoder has two hidden layers with 256 and 512 dimensions, respectively. The bottleneck layer (latent dimension) of both AE and VAE has 64 dimensions.

In the determination of the best-performed models of both AE and VAE, the dimensions in the layers were specified with trial and error by keeping the number of layers constant. The parameter settings used in the neural network configuration of both AE and VAE are as follows: The learning rates such as [0.1, 0.01, 0.001] and the momentums such as [0.3, 0.5, 0.9] are used in the training trials. The best-performed values are 0.001 for the learning rate and 0.3 for momentum. The parameter of l2 regularization [65] is set to the commonly used value of 0.001 and Xavier [66] is used as weight initialization method, which is the default in the library and is recommended if ReLU is employed in the hidden layers [64]. Both neural networks are trained with the backpropagation algorithm using conjugate gradient optimization algorithm, which is a recommended choice for large datasets and real-valued outputs [64]. Nesterovs updater [67] is selected because it uses the momentum that supports the learning procedure to escape local minima and discover better solutions to the optimization process [64]. The additional parameters used in VAE configuration are as follow. Leaky Rectified Linear Unit [68] is used in hidden layers as activation function. It is preferred over sigmoid or tanh since it resolves the “dying ReLU” problem of the vanilla ReLU and the vanishing gradient problem of sigmoid/tanh [64]. Gaussian reconstruction distribution is used with hyperbolic tangent (tanh) pzx activation function because the type of data being modeled is real-valued [42]. In the additional parameters used in AE configuration, which are commonly used and constitute the basis of the studies such as [41], [69], the sigmoid activation function is used, except for the output layer employing soft-max activation function [65], with mean square error loss function. Both AE and VAE neural networks were implemented with deeplearning4j library [70] and trained in 1000 epochs with a batch size of 8192.

In OCSVM, radial basis function was used as a kernel with the default parameters, where  $\gamma$  (gamma) was set to 0.5, the cost parameter C was set to 1 and the parameter  $\nu$  (nu) was set to 0.5. The model was created using LibSVM library [71].

As an anomaly score, the RP is used in VAE, the RE is used in AE, and the prediction score is used in OCSVM without applying any threshold (if the score is less than 0, the example is an anomaly, else it is normal).

## C. PERFORMANCE EVALUATION

Initially, the normalization was carried out in the features of the dataset using the feature scaling method [72], which was also called unity-based normalization, to bring all values into the range [0,1]. Separate training and test datasets were used to assess the performance of the methods. The models were created by training the neural network with only normal flow data on “Monday”. The reason is that the models are trained in unsupervised approach, the last column in dataset corresponds to attack class that is not used in training. Both the normal and attack traffic in other days were used for the testing process. In the testing process, all the classes



**TABLE 1.** The number of examples in CICDS2017 dataset categorized by attack type and day.

		Number of Flows (Daily)				
	Traffic Type	Monday	Tuesday	Wednesady	Thursday	Friday
<i>NORMAL</i>	Benign	529918	432074	440031	456752	500514
<i>ANOMALY</i>	FTP-Patator		7938			
	SSH-Patator		5897			
	DoS slowloris			5796		
	DoS Slowhttptest			5499		
	DoS Hulk			231073		
	DoS GoldenEye			10293		
	Heartbleed			11		
	Web Attack–Brute force				1507	
	Web Attack–XSS				652	
	Web Attack–SQL Injection				21	
	Infiltration				36	
	Bot					1966
	DDoS					41835
Port scan					158930	
<b>Daily Total</b>		529918	445909	692703	458968	703245
<b>TOTAL</b>		2830743				

except “normal” or “benign” are considered as “anomaly”. In the performance evaluation of an attack class, the metrics are calculated only using the normal flow records and flow records of that attack class excluding all the other attack classes. For example, in the evaluation of “FTP-Patator” attack, “1829371” normal flow records and “7938” attack flow records are used. Another evaluation strategy is also performed by combining all attacks into one class “anomaly” other than “normal” in order to comprehend if the methods can discriminate the unknown attacks from normal traffic.

The performance evaluation of methods is accomplished by using both ROC and AUC metrics. An evaluation that is based on AUC measure, is carried out as it is the de facto standard in unsupervised anomaly detection along with its feasible interpretability [73]. More specifically, the whole ROC curve is summarized while the performance is aggregated over the entire range of threshold values [73]. ROC based evaluation was performed as it was insensitive to class distribution and also to demonstrate that one method could perform better than another in different thresholds (corresponds to different FP values in ROC graph).

The AUC results of the methods are given in Table 2. In addition, the ROC curves of the algorithm applied to different kinds of unknown (meaning not trained) attacks are demonstrated in Fig. 8. The AUC results and the interpretation of ROC curves are explained as follows. An academic point system was also used as a guide in the interpretation of AUC results.

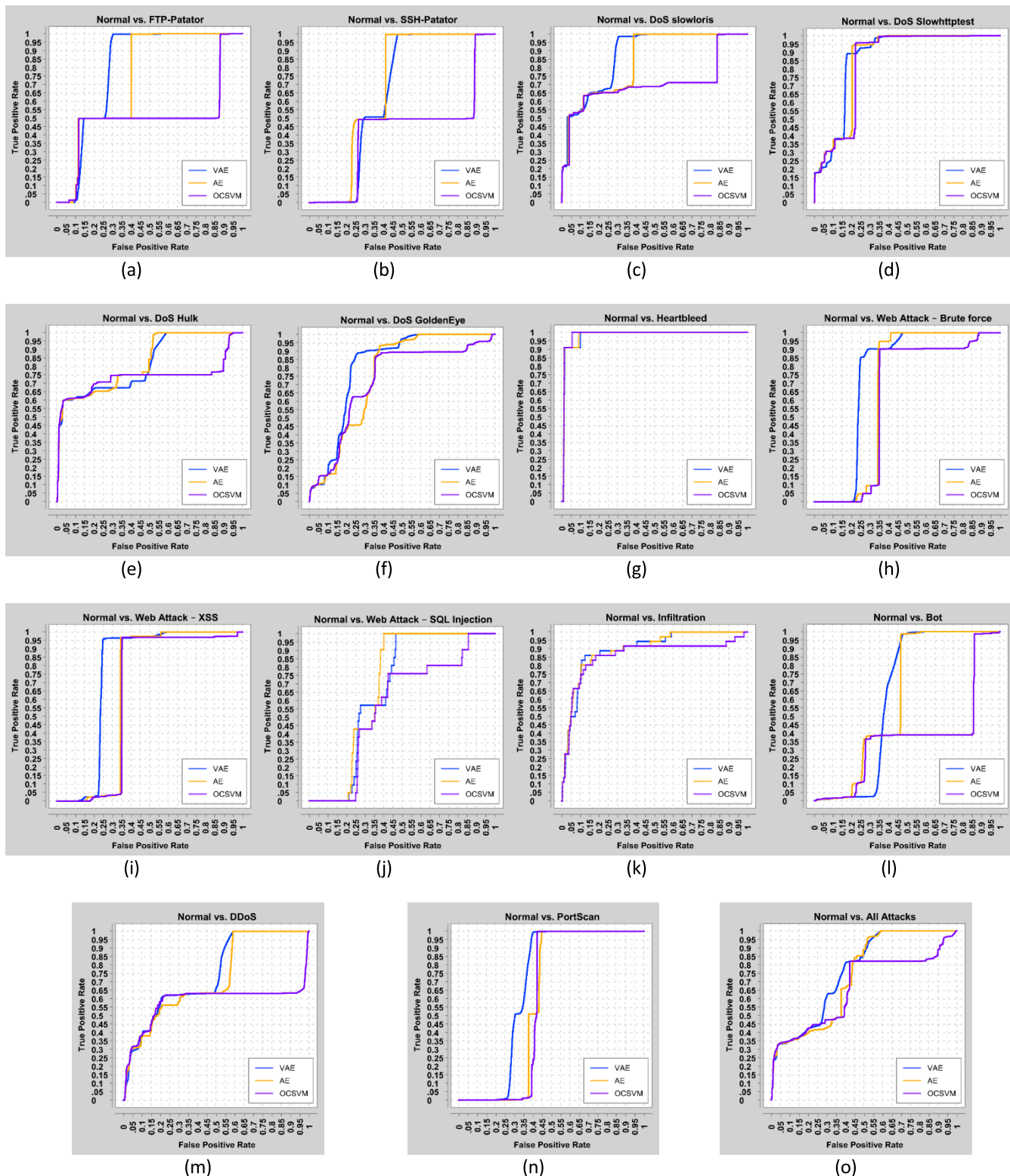
If the AUC values in Table 2 are analyzed in general, it is realized that VAE performs better than the other two methods, and OCSVM performs poorly, except for some attacks. Moreover, some attacks have excellent or good AUC values for all three methods, while others show poor or unrealistic

**TABLE 2.** The AUC results of the methods for specified attacks.

Anomaly Type	AUC (Normal vs. Anomaly)		
	VAE	AE	OCSVM
<i>FTP Patator</i>	<b>0.7955</b>	0.7429	0.5028
<i>SSH Patator</i>	0.6472	<b>0.6787</b>	0.4221
<i>DoS Slowloris</i>	<b>0.8752</b>	0.8428	0.7059
<i>Dos Slowhttptest</i>	<b>0.8664</b>	0.8529	0.8421
<i>Dos Hulk</i>	0.8155	<b>0.8275</b>	0.7361
<i>Dos Goldeneye</i>	<b>0.8088</b>	0.7536	0.7228
<i>Heartbleed</i>	0.9805	0.9816	<b>0.9849</b>
<i>Web Attack–Brute force</i>	<b>0.7436</b>	0.6622	0.6090
<i>Web Attack–XSS</i>	<b>0.7567</b>	0.6571	0.6372
<i>Web Attack–SQL Injection</i>	0.6664	<b>0.6878</b>	0.5695
<i>Infiltration</i>	<b>0.8965</b>	0.8964	0.8588
<i>Bot</i>	0.6197	<b>0.621</b>	0.3749
<i>DDoS</i>	<b>0.7452</b>	0.7221	0.5840
<i>Portscan</i>	<b>0.6753</b>	0.5953	0.5888
<i>All Attacks</i>	<b>0.7596</b>	0.7338	0.6636

\*The bold values indicate best-performed method.

performance which indicate they fail to distinguish attacks or abnormal cases from normal. It is observed that VAE detects abnormal cases better than AE does in the occurrence of high-rate of attacks such as various kinds of DoS attacks and DDoS, and both of them exhibit good performance. In the attacks of “SSH Patator”, “Web Attack – SQL Injection”, “Bot” and “Portscan”, it is noticed that all three methods perform poorly. Although each one contains a different number of attack samples and is in different attack categories such as brute force, web application attack, bot, and port



**FIGURE 8.** The ROC curve of (a) normal vs FTP-Patator, (b) normal vs SSH-Patator, (c) normal vs DoS Slowloris, (d) normal vs DoS Slowhttptest, (e) normal vs DoS Hulk, (f) normal vs DoS GoldenEye, (g) normal vs Heartbleed, (h) normal vs Web Attack-Brute force (i) normal vs Web Attack-XSS, (j) normal vs Web attack-SQL injection, (k) normal vs infiltration, (l) normal vs Bot, (m) normal vs DDoS, (n) normal vs PortScan, (o) normal vs all attacks.

scan respectively, it can be seen that none of the training models of all three methods are able to distinguish these attacks sufficiently. Consequently, additional research needs

to be carried out to improve the performance of the models and to differentiate between these attacks by appending further different flow-based features or data or employing

more highly developed machine learning methodology. It is interesting to see that all three methods appear to exhibit good and excellent performances in the cases of “Infiltration” and “Heartbleed” attacks, respectfully. The reason for this may be that the certain attack types are in small numbers in the dataset, so further research need to be conducted to confirm this assumption by increasing the number of samples of these attacks.

By combining all attacks types as an anomaly to calculate a single AUC result and ROC curve, how the methods behave when “All attacks” take place in the network can be assessed. According to AUC results, VAE wins over all others with fair performance. The AE also displays fair performance whereas the performance of OCSVM is poor. This denotes that the ability of VAE and AE to model normal network traffic is quite successful in detecting fourteen different types of unknown intrusions in wide variety of attack categories. Moreover, it should be noted that although the number of data samples used in training process (about 19% of the total dataset) is much less than the number of data samples used in the testing process (about 81% of the total data set), the good performances of both VAE and AE clearly show that these methods are highly capable of detecting the intrusions or anomalies from flow based features. The use of much more normal data samples in training (ratios of commonly used splitting is 70% for training and 30% for testing in supervised learning approach) when building the model of normal network traffic from flow features can increase the performance ratio even further.

If all ROC curves in Fig. 8 are examined, it is seen that the attacks having the same behavioral nature exhibit similar ROC curves in VAE even if it is not applicable to all attacks. For example, the ROC curves of “Web Attack-Brute force”, “Web Attack-XSS” and “Web Attack-SQL Injection” proves this assumption right. Another similar observation can be made for “FTP Patator” and “SSH Patator” in the category of brute force. Even though AUC values of VAE and AE methods yield good performance, the indicators that make it difficult to use these methods in real life scenario without any supplementary methods are clearly noticed in ROC curves. Moreover, all ROC curves show that no high TP values could be obtained in the low FP values in almost any of the attack types (except Heartbleed). This observation can be interpreted as it is not easy to determine a proper threshold value that provides high detection accuracy or low false alarm rate in a practical intrusion detection system.

## V. CONCLUSION

In this study, the detection capabilities of AE and VAE deep learning methods together with OCSVM were analyzed by applying a semi-supervised learning strategy. The creation of the models was carried out using normal flow-based data only. Moreover, the testing of the models was realized by using both normal and anomaly data. The experimental results were computed in terms of ROC curves and AUC metrics. Based on the results, the detection rate of VAE is

better, for the most part, than AE and OCSVM. However, it is also important to mention that the methods need to be supported by supervised learning methods due to their high false alarms. Furthermore, in order to increase the detection rate of methods, the flow-based features collected at specified time intervals can be considered due to the fact that the characteristics of some attacks could be better modelled.

We thank Google by providing free credit in order to be able to use its infrastructure for training and testing algorithms.

## REFERENCES

- [1] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanovic, “Flow-based anomaly intrusion detection system using two neural network stages,” *Comput. Sci. Inf. Syst.*, vol. 11, no. 2, pp. 601–622, 2014, doi: [10.2298/CSIS130415035A](https://doi.org/10.2298/CSIS130415035A).
- [2] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, “Autoencoder-based feature learning for cyber security applications,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 3854–3861, doi: [10.1109/IJCNN.2017.7966342](https://doi.org/10.1109/IJCNN.2017.7966342).
- [3] B. Zhang, Y. Yu, and J. Li, “Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method,” in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2018, pp. 1–6, doi: [10.1109/ICCW.2018.8403759](https://doi.org/10.1109/ICCW.2018.8403759).
- [4] B. Abolhasanzadeh, “Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features,” in *Proc. 7th Conf. Inf. Knowl. Technol. (IKT)*, May 2015, pp. 1–5, doi: [10.1109/IKT.2015.7288799](https://doi.org/10.1109/IKT.2015.7288799).
- [5] S. N. Mighan and M. Kahani, “Deep learning based latent feature extraction for intrusion detection,” in *Proc. Iranian Conf. Electr. Eng. (ICEE)*, May 2018, pp. 1511–1516, doi: [10.1109/ICEE.2018.8472418](https://doi.org/10.1109/ICEE.2018.8472418).
- [6] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: [10.1109/TETCI.2017.2772792](https://doi.org/10.1109/TETCI.2017.2772792).
- [7] U. Cekmez, Z. Erdem, A. G. Yavuz, O. K. Sahingoz, and A. Buldu, “Network anomaly detection with deep learning,” in *Proc. 26th Signal Process. Commun. Appl. Conf. (SIU)*, May 2018, pp. 1–4, doi: [10.1109/SIU.2018.8404817](https://doi.org/10.1109/SIU.2018.8404817).
- [8] R. C. Aygun and A. G. Yavuz, “A stochastic data discrimination based autoencoder approach for network anomaly detection,” in *Proc. 25th Signal Process. Commun. Appl. Conf. (SIU)*, May 2017, pp. 1–4, doi: [10.1109/SIU.2017.7960410](https://doi.org/10.1109/SIU.2017.7960410).
- [9] 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. 48231–48246, 2018, doi: [10.1109/ACCESS.2018.2863036](https://doi.org/10.1109/ACCESS.2018.2863036).
- [10] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, “Autoencoder-based network anomaly detection,” in *Proc. Wireless Telecommun. Symp. (WTS)*, Apr. 2018, pp. 1–5, doi: [10.1109/WTS.2018.8363930](https://doi.org/10.1109/WTS.2018.8363930).
- [11] *KDD Cup 1999 Data*. Accessed: Mar. 30, 2019. [Online]. Available: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
- [12] F. Farahnakian and J. Heikkonen, “A deep auto-encoder based approach for intrusion detection system,” in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 178–183, doi: [10.23919/ICACT.2018.8323688](https://doi.org/10.23919/ICACT.2018.8323688).
- [13] *NSL-KDD Dataset*. Accessed: Jan. 30, 2019. [Online]. Available: <https://www.unb.ca/cic/datasets/nsl.html>
- [14] J. McHugh, “Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory,” *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000.
- [15] M. F. Umer, M. Sher, and Y. Bi, “Flow-based intrusion detection: Techniques and challenges,” *Comput. Secur.*, vol. 70, no. 1, pp. 238–254, 2017, doi: [10.1016/j.cose.2017.05.009](https://doi.org/10.1016/j.cose.2017.05.009).
- [16] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, “An overview of IP flow-based intrusion detection,” *IEEE Commun. Surveys Tuts.*, vol. 12, no. 3, pp. 343–356, 3rd Quart., 2010, doi: [10.1109/SURV.2010.032210.00054](https://doi.org/10.1109/SURV.2010.032210.00054).

- [17] R. Beghdad, "Critical study of neural networks in detecting intrusions," *Comput. Secur.*, vol. 27, nos. 5–6, pp. 168–175, Oct. 2008, doi: [10.1016/j.cose.2008.06.001](https://doi.org/10.1016/j.cose.2008.06.001).
- [18] S. Song, L. Ling, and C. N. Manikopoulos, "Flow-based statistical aggregation schemes for network anomaly detection," in *Proc. IEEE Int. Conf. Netw., Sens. Control*, Dec. 2006, pp. 786–791, doi: [10.1109/icnsc.2006.1673246](https://doi.org/10.1109/icnsc.2006.1673246).
- [19] Q. A. Tran, F. Jiang, and J. Hu, "A real-time netFlow-based intrusion detection system with improved BBNN and high-frequency field programmable gate arrays," in *Proc. IEEE 11th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jun. 2012, pp. 201–208, doi: [10.1109/TrustCom.2012.51](https://doi.org/10.1109/TrustCom.2012.51).
- [20] Z. Jadidi, V. Muthukkumarasamy, and E. Sithirasanen, "Metaheuristic algorithms based flow anomaly detector," in *Proc. 19th Asia-Pacific Conf. Commun. (APCC)*, Aug. 2013, pp. 717–722, doi: [10.1109/APCC.2013.6766043](https://doi.org/10.1109/APCC.2013.6766043).
- [21] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–4, doi: [10.14722/ndss.2018.23204](https://doi.org/10.14722/ndss.2018.23204).
- [22] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark," *IEEE Access*, vol. 6, pp. 59657–59671, 2018, doi: [10.1109/ACCESS.2018.2875045](https://doi.org/10.1109/ACCESS.2018.2875045).
- [23] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: [10.1109/ACCESS.2019.2895334](https://doi.org/10.1109/ACCESS.2019.2895334).
- [24] P. Winter, E. Hermann, and M. Zeilinger, "Inductive intrusion detection in flow-based network data using one-class support vector machines," in *Proc. 4th IFIP Int. Conf. New Technol., Mobility Secur.*, Feb. 2011, pp. 1–5, doi: [10.1109/NTMS.2011.5720582](https://doi.org/10.1109/NTMS.2011.5720582).
- [25] C. Wagner, J. François, R. State, and T. Engel, "Machine learning approach for IP-flow record anomaly detection," *Lecture Notes Comput. Sci.*, vol. 6640, no. 1, pp. 28–39, 2011, doi: [10.1007/978-3-642-20757-0\\_3](https://doi.org/10.1007/978-3-642-20757-0_3).
- [26] M. F. Umer, M. Sher, and Y. Bi, "A two-stage flow-based intrusion detection model for next-generation networks," *PLoS ONE*, vol. 13, no. 1, Jan. 2018, Art. no. e0180945, doi: [10.1371/journal.pone.0180945](https://doi.org/10.1371/journal.pone.0180945).
- [27] A. Shubair, S. Ramadass, and A. A. Altyeb, "KENFIS: KNN-based evolving neuro-fuzzy inference system for computer worms detection," *J. Intell. Fuzzy Syst.*, vol. 26, no. 4, pp. 1893–1908, 2014, doi: [10.3233/IFS-130868](https://doi.org/10.3233/IFS-130868).
- [28] K. A. P. Costa, L. A. M. Pereira, R. Y. M. Nakamura, C. R. Pereira, J. P. Papa, and A. X. Falcão, "A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks," *Inf. Sci.*, vol. 294, pp. 95–108, Feb. 2015, doi: [10.1016/j.ins.2014.09.025](https://doi.org/10.1016/j.ins.2014.09.025).
- [29] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2005, p. 217, doi: [10.1145/1080091.1080118](https://doi.org/10.1145/1080091.1080118).
- [30] P. Casas, J. Mazel, and P. Owezarski, "UNADA: Unsupervised network anomaly detection using sub-space outliers ranking," in *Lecture Notes Comput. Sci.*, vol. 6640, no. 1, pp. 40–51, 2011, doi: [10.1007/978-3-642-20757-0\\_4](https://doi.org/10.1007/978-3-642-20757-0_4).
- [31] F. Hosseinpour, P. V. Amoli, F. Farahnakian, J. Plosila, T. Hämäläinen, and C. Author, "Artificial immune system based intrusion detection: Innate immunity using an unsupervised learning approach," *Int. J. Digit. Content Technol. Appl.*, vol. 8, no. 5, 2014.
- [32] A. Satoh, Y. Nakamura, and T. Ikenaga, "A flow-based detection method for stealthy dictionary attacks against secure shell," *J. Inf. Secur. Appl.*, vol. 21, pp. 31–41, Apr. 2015, doi: [10.1016/j.jisa.2014.08.003](https://doi.org/10.1016/j.jisa.2014.08.003).
- [33] S. Thaseen and C. A. Kumar, "An analysis of supervised tree based classifiers for intrusion detection system," in *Proc. Int. Conf. Pattern Recognit., Informat. Mobile Eng.*, Feb. 2013, pp. 294–299, doi: [10.1109/ICPRIME.2013.6496489](https://doi.org/10.1109/ICPRIME.2013.6496489).
- [34] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013, doi: [10.1016/j.cose.2013.04.007](https://doi.org/10.1016/j.cose.2013.04.007).
- [35] F. Haddadi, D. Runkel, A. Nur Zincir-Heywood, and M. I. Heywood, "On botnet behaviour analysis using GP and C4.5," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, 2014, pp. 1253–1260, doi: [10.1145/2598394.2605435](https://doi.org/10.1145/2598394.2605435).
- [36] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2014, pp. 797–801, doi: [10.1109/ICNC.2014.6785439](https://doi.org/10.1109/ICNC.2014.6785439).
- [37] Y. Kanda, R. Fontugne, K. Fukuda, and T. Sugawara, "ADMIRE: Anomaly detection method using entropy-based PCA with three-step sketches," *Comput. Commun.*, vol. 36, no. 5, pp. 575–588, Mar. 2013, doi: [10.1016/J.COMCOM.2012.12.002](https://doi.org/10.1016/J.COMCOM.2012.12.002).
- [38] M. H. Haghghat and J. Li, "Edmund: Entropy based attack detection and mitigation engine using netflow data," in *Proc. 8th Int. Conf. Commun. Netw. Secur.*, 2018, pp. 1–6, doi: [10.1145/3290480.3290484](https://doi.org/10.1145/3290480.3290484).
- [39] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 6, 1994, pp. 3–10, doi: [10.1021/jp906511z](https://doi.org/10.1021/jp906511z).
- [40] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biol. Cybern.*, vol. 59, nos. 4–5, pp. 291–294, Sep. 1988, doi: [10.1007/BF00332918](https://doi.org/10.1007/BF00332918).
- [41] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. MLSDA 2nd Workshop Mach. Learn. Sensory Data Anal.*, 2014, p. 4, doi: [10.1145/2689746.2689747](https://doi.org/10.1145/2689746.2689747).
- [42] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," SNU Data Mining Center, Seoul, South Korea, Tech. Rep., 2015.
- [43] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: <http://arxiv.org/abs/1312.6114>
- [44] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152, doi: [10.1145/130385.130401](https://doi.org/10.1145/130385.130401).
- [45] Z.-Q. Zeng, H.-B. Yu, H.-R. Xu, Y.-Q. Xie, and J. Gao, "Fast training support vector machines using parallel sequential minimal optimization," in *Proc. 3rd Int. Conf. Intell. Syst. Knowl. Eng.*, Nov. 2008, pp. 185–208.
- [46] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.
- [47] Y. Wang, J. Wong, and A. Miner, "Anomaly intrusion detection using one class SVM," in *Proc. 5th Annu. IEEE SMC Inf. Assurance Workshop*, 2004, pp. 358–364, doi: [10.1109/IAW.2004.1437839](https://doi.org/10.1109/IAW.2004.1437839).
- [48] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011.
- [49] O. Gu, P. Fogla, D. Dagon, W. Lee, and B. Škorić, "Measuring intrusion detection capability: An information-theoretic approach," in *Proc. ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, vol. 2006, 2006, pp. 90–101, doi: [10.1145/1128817.1128834](https://doi.org/10.1145/1128817.1128834).
- [50] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018, doi: [10.1109/ACCESS.2018.2836950](https://doi.org/10.1109/ACCESS.2018.2836950).
- [51] K. A. Spackman, "Signal detection theory: Valuable tools for evaluating inductive learning," in *Proc. 6th Int. Workshop Mach. Learn.*, 1989, pp. 160–163.
- [52] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," HP Labs, Palo Alto, CA, USA, Tech Rep. HPL-2003-4, 2004, pp. 1–38, doi: [10.1.1.10.9777](https://doi.org/10.1.1.10.9777).
- [53] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*. Hoboken, NJ, USA: Wiley, 2013.
- [54] A. Tharwat, "Classification assessment methods," *Appl. Comput. Inform.*, pp. 1–13, Aug. 2018, doi: [10.1016/j.aci.2018.08.003](https://doi.org/10.1016/j.aci.2018.08.003).
- [55] G. T. Tape, *The Area Under an ROC Curve*. Accessed: Feb. 18, 2019. [Online]. Available: <http://gim.unmc.edu/dxtests/roc3.htm>
- [56] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synth. Lect. Artif. Intell. Mach. Learn.*, vol. 3, no. 1, pp. 1–130, 2009, doi: [10.2200/S00196ED1V01Y200906AIM006](https://doi.org/10.2200/S00196ED1V01Y200906AIM006).
- [57] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical Analysis of HoneyPot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation," in *Proc. 1st Workshop Building Anal. Datasets Gathering Exper. Returns Secur.*, vol. 2011, pp. 29–36, doi: [10.1145/1978672.1978676](https://doi.org/10.1145/1978672.1978676).
- [58] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014, doi: [10.1016/J.COSE.2014.05.011](https://doi.org/10.1016/J.COSE.2014.05.011).
- [59] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf.*, 2015, pp. 1–8, doi: [10.1109/MilCIS.2015.7348942](https://doi.org/10.1109/MilCIS.2015.7348942).
- [60] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proc. 16th Eur. Conf. Cyber Warfare Secur. (ACPI)*, 2017, pp. 361–369.

- [61] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116, doi: [10.5220/0006639801080116](https://doi.org/10.5220/0006639801080116).
- [62] C. Beek, D. Dinkar, Y. Gund, G. Lancioni, N. Minihane, F. Moreno, E. Peterson, T. Roccia, C. Schmugar, and R. Simon, "McAfee labs threats report: September 2017," McAfee, Santa Clara, CA, USA, Tech. Rep., 2017. Accessed: Feb. 20, 2019. [Online]. Available: <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-sept-2017.pdf>
- [63] V. Loi Cao, M. Nicolau, and J. McDermott, "Learning neural representations for network anomaly detection," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3074–3087, Aug. 2019, doi: [10.1109/TCYB.2018.2838668](https://doi.org/10.1109/TCYB.2018.2838668).
- [64] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach*. Newton, MA, USA: O'Reilly Media, 2017.
- [65] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
- [66] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," Tech. Rep.
- [67] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," Tech. Rep., 2013.
- [68] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML Workshop Deep Learn. Audio, Speech Lang. Process.*, 2013, pp. 1–8. Accessed: Jan. 30, 2019. [Online]. Available: [https://ai.stanford.edu/~amaas/papers/relu\\_hybrid\\_icml2013\\_final.pdf](https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf)
- [69] C. Zhou and R. C. Paaenroth, "Anomaly detection with robust deep autoencoders," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 665–674, doi: [10.1145/3097983.3098052](https://doi.org/10.1145/3097983.3098052).
- [70] (2019). *Deeplearning4j*. Accessed: Feb. 23, 2019, [Online]. Available: <https://deeplearning4j.org/>
- [71] *LIBSVM—A Library for Support Vector Machines*. Accessed: Feb. 23, 2019. [Online]. Available: <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- [72] S. Aksoy and R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognit. Lett.*, vol. 22, no. 5, pp. 563–582, 2001.
- [73] W. J. Krzanowski and D. J. Hand, *ROC Curves for Continuous Data*. Boca Raton, FL, USA: CRC Press, 2009.
- [74] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013, doi: [10.1016/j.jnca.2012.09.004](https://doi.org/10.1016/j.jnca.2012.09.004).



**SULTAN ZAVRAK** received the B.Sc. and M.Sc. degrees in computer engineering from Karadeniz Technical University, Trabzon, Turkey, in 2010 and 2013, respectively. He is currently pursuing the Ph.D. degree with Sakarya University, Sakarya, Turkey, under the supervision of Murat İskefiyeli. He is also working as a Research Assistant with the Department of Computer Engineering, Engineering Faculty, Düzce University. His research interests include computer networks, network security, and machine learning.



**MURAT İSKEFIYELI** (Associate Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in electrical-electronics engineering from Sakarya University, Sakarya, Turkey, in 1998, 2002, and 2010, respectively. He is currently working as an Assistant Professor with the Department of Computer Engineering, Faculty of Computer and Information Sciences, Sakarya University. His research interests include computer networks, network security, and machine learning.

• • •