The Journal of Supercomputing (2021) 77:3571–3593
https://doi.org/10.1007/s11227-020-03410-y

Network intrusion detection using multi-architectural modular deep neural network

Ramin Atefinia[1] · Mahmood Ahmadi[1]

Published online: 25 August 2020

Abstract
The exponential growth of computer networks and the adoption of new network-based technologies have made computer security an important challenge. With the emergence of new internet-connected devices, the attack surface is increasing for cyber intruders. Many intrusion detection systems attempt to detect known attacks using signatures in network traffic. In recent years, researchers used several machine learning techniques to detect network attacks without relying on these signatures. These techniques generally suffer from a high false-positive rate which is not acceptable for an industry-ready intrusion detection product. In this paper, we propose a multi-architectural modular deep neural network model to decrease the false-positive rate of anomaly-based intrusion detection systems. Our model consists of a feed-forward module, a stack of restricted Boltzmann machine module, and two recurrent modules, the output weights of these modules are fed to an aggregator module to produce the answer of the model. The experiments are performed using CSE-CIC-IDS2018 dataset, and final models can be used in an IDS for generating alerts or preventing new attacks. The experimental results show improvement in the detection of some types of network attacks with accuracy as high as 100% for network-level attacks compared to related works.

Keywords Intrusion detection · Artificial neural networks · Cyber security

✉ Mahmood Ahmadi
m.ahmadi@razi.ac.ir

Ramin Atefinia
ramina@post.com

[1] Computer Engineering and Information Technology Department, Razi University, Kermanshah, Iran

1 Introduction

Computer and Information Security is a growing problem. In the past 20 years, intrusion techniques as well as security protections have advanced rapidly. Although cyberattacks have evolved using new techniques, most organizations are still using the old generation of cybersecurity measures. These new attacks can bypass the static defense methods being used by today's organizations. The government today holds valuable information on web servers which include sensitive data related to every citizen. This makes web servers a popular target for intruders.

An Intrusion Detection System (IDS) is a security software/hardware system that alerts administrators when suspicious activity is discovered in networks or computers. Some systems can also attempt to stop intrusions and block the potential threats; these systems are called Intrusion Prevention Systems (IPS), but organizations should consider that an IPS can block legitimate activity due to false positives. In terms of scope, an IDS can be classified into network-based, host-based, or hybrid systems. Host-based IDS (HIDS) analyzes and monitors inside of computers such as operating system audit records, application logs, and key system files for suspicious activity. Network-based IDS (NIDS) monitors the network for suspicious traffic and is deployed in a strategic point or multiple points in the network [25, 27]. An IDS that can detect attacks from host and network sources is using the hybrid approach.

In terms of detection method, an IDS can be classified into signature-based or misuse detection and anomaly-based methods. The signature-based method originates from anti-virus software that can detect patterns known as signatures. This type of detection has high accuracy and low false-positive rate for already recognized attacks, but when no pattern is available, it is difficult to detect the new attacks. The anomaly-based method tries to detect unknown attacks by using techniques such as machine learning to create a model of normal and anomaly or a class of anomalous activities and compare the new traffic pattern against the model. The problem of this approach is a higher false-positive rate and more time needed to detect anomalies. Based on recent published academic works, anomaly-based intrusion detection shows more promising results in detecting network attacks compared to host-based attacks. In the past two decades, researchers have used several machine learning methods to enhance the performance and accuracy of anomaly-based network IDSes. These techniques include pattern tracking, classification, clustering, association, outlier detection, regression, and prediction. Examples of these include Self-Organising Map (SOM) [14, 29], K-Nearest Neighbor (KNN) [4], Decision Tree [19, 28], support vector machine (SVM) [1, 30], etc. The main weakness of nearest-neighbor-based techniques is that if normal traffic data do not have close enough neighbors or attack traffic do not have close enough neighbors, the KNN technique cannot properly label the traffic. Also computing the distance of each item might require complex computation. The clustering techniques might not perform well in small datasets, and dynamic updating of attack profiles might require a high amount of time. The statistical techniques also depend on some assumptions about the generation of data in a particular distribution.

Over last few years, the use of neural networks and deep learning methods for intrusion detection proved to be very more effective [6]. These networks have unique properties that result in a higher detection rate and a low false-positive rate. The deep learning strength is the ability of the neural network to perform automatic feature selection, but dimensionality reduction techniques like Principal Component Analysis (PCA) can still be used by common sense. Examples of deep learning methods deployed for intrusion detection include autoencoders [31] Convolutional Neural Network (CNN) [15, 33, 36], Multilayer Perceptron (MLP) [7, 13], Feed-Forward Artificial Neural Network (FNN) [16], Deep Belief Network (DBN) [17], etc. The underfitting vs overfitting problem is usually a challenge in these methods.

In the traditional machine learning approach to intrusion detection, we need to build entities and make representations of data; therefore, too much feature engineering is needed in these approaches. Useful features of network traffic data always need to be represented in these techniques [24]. Deep Learning algorithms with the ability of automated extraction of very complex features can overcome these challenges. In traditional Machine learning techniques, a domain expert should identify the applied features to reduce the complexity and make patterns visible to the algorithm. Deep Learning algorithms can eliminate the need for domain expertise and save more time. In determining the success of an IDS method, reducing the number of false-positives is the main factor. If we develop a model with higher accuracy, a smaller number of unnecessary generated alerts need to be suppressed in the deployed system and less manual confirmation is needed to verify their legitimacy. This issue even exists for the current signature-based IDS technology but the number of false-positives is currently higher in anomaly-based methods. By disabling some rules in a signature-based IDS, we can reduce false alarm rates and have a more accurate system. In this work, we have reduced the false-positive rate for network-level attacks by using a modular architecture. A better architecture of the model can result in a lower false-positive rate and narrow the gap between anomaly-based and signature-based systems.

In this paper, we focus on analyzing features extracted from network attacks on vulnerable systems using different network and web attack tools; we construct a new bio-inspired modular deep neural network architecture using a feed-forward module, restricted Boltzmann machines (SRBM) module, RNN with long short-term memory (LSTM) units [12] and RNN with gated recurrent units (GRU) [5]. The main advantages of this method are fault tolerance, lower false positive, functional specialization, scalability, and extendibility. The results of our work show that the accuracy for some type of network-level attacks can be as high as 100%.

The main contributions of this paper are in the following.

- Proposal of a novel modular deep neural network model, with hyperparameter optimization for the detection of network attacks.
- Use of more efficient techniques such as batch normalization in the feed-forward module instead of dropout to maximum benefit from the batch norm.
- The use of an efficient aggregator module to reduce the variance of a single neural network model.

The rest of the paper is organized as follows. In Sect. 2 of this paper, we review several related approaches for the identification of web attacks and detection of network intrusion. In Sect. 3, we describe modular neural networks. In Sect. 4, we describe our proposed approach and describe our dataset, and in Sect. 5, experimental results are presented. Finally Sect. 6 explains the conclusion of the paper.

2 Related work

The network traffic generally consists of legitimate activity with only a few attack attempts. This makes network anomaly detection an imbalanced classification task and requires up to date datasets with realistic scenarios. In this section, recent works on two realistic datasets, namely CIC-IDS-2017 and CSE-CIC-IDS2018, are briefly discussed.

In [35], they suggested a two-level anomalous activity detection model in order to detect intrusions in the Internet of Things networks. The level-1 model categorizes the network flow as normal or anomaly and operates at the network layer. The network flow is then forwarded to the level-2 model to determine which category the detected anomaly belongs to. They used the decision tree classifier for the level-1 model. They showed that a binary classification can achieve 100% Precision for CICIDS2017 dataset while the performance measures are lower than determining the class of attack, especially web attacks.

In [32], they used four variable selection algorithms, namely Classifier Subset Evaluator using Naive Bayes, CfsSubset Attribute Evaluator, Classifier Subset Evaluator using J48, and Classifier Subset Evaluator combined with Decision Tree using weka data mining tool and applied OneR and REPTree to CICIDS2017 dataset. The results show that different combinations of variable selection algorithms along with the REPTree algorithm can provide the best performance for each different type of attack. Thus, a single feature selection technique cannot provide the best result for all types of attacks.

In [3], they used state-of-the-art deep learning frameworks such as TensorFlow, Theano, fast.ai, and PyTorch to detect network intrusions in CSE-CIC-IDS2018 dataset. The results of their work show that the fast.ai library that sits on top of PyTorch provides the best results with accuracy as high as about 99% with low false rates in the detection and also classification of different intrusion types.

In research at [37], they used six supervised machine learning classification models to detect Zero-Day intrusions. They implemented algorithms using sklearn and numpy library. By using data laundry techniques such as deleting noisy features, formatting data into standard datatype, replacing NaN and Infinity numbers on CSE-CIC-IDS2018 dataset they achieved high accuracy results; however, the result of their work cannot be compared with others since they reduced the labels to two classes and the method they used to change the dataset is unknown.

In [18], they used six algorithms, namely Linear Discriminant Analysis, K Nearest Neighbor, Gradient Boosting, Adaboost, Decision Tree, and Random Forest to carry out a practical IDS using CSE-CIC-IDS2018 cybersecurity dataset. They used a synthetic data generation model known as SMOTE to reduce the imbalance ratio.

This approach is shown to enhance the detection rate for intrusions with few samples in the CSE-CIC-IDS2018 dataset.

In [22], they used LSTM to make a model with good performance that can process time-correlated sequences. They used the CSE-CIC-IDS2018 dataset to reflect the real attributes of today's network traffic. They also used an over-sampling algorithm known as SMOTE to solve the class-imbalance issue. In their experimental results, they achieved 96.2% overall accuracy by combining two techniques.

In [26], they used the broad learning system (BLS) to detect denial of service (DoS) attack in communication networks. The effectiveness of their approach is tested using subsets of CICIDS2017 and CSE-CIC-IDS2018 dataset that both contain DoS attacks. The developed BLS and its extensions are evaluated as alternative supervised learning algorithms for the detection of network anomalies.

In [21], they used CICIDS2017, CIC-DoS and CSE-CIC-IDS2018 and a customized dataset to detect DoS and DDoS Attack Using Machine Learning. Their approach has the advantage of early detection of a variety of volumetric attacks including UDP flood, TCP flood and HTTP flood, etc. Their smart detection system benefits from the Random Forest Tree algorithm to classify and label network traffic based on samples. The samples are taken directly from network devices by the sFlow protocol. Based on the Smart Detection approach results, the Detection Rate (DR) is higher than 93% with False Alarm Rate (FAR) as low as 1%.

In a patent at [20], they proposed a hierarchical neural network to monitor the network functions. In this method, we need to build a set of primary neural networks that can receive inputs associated with respective ones of the network functions. Each primary neural network has an output, and tier 1 neural networks are operatively connected to consolidate selected outputs of the primary neural networks. Detection of an anomaly is accomplished by monitoring chosen areas of network behavior, such as protocols, that are anticipated in advance. Joining outputs of neural networks within the hierarchical network yields adequate anomaly detection.

The main advantage of our work compared to similar works is lower optimizations needed for parameters. That is because our approach follows a bio-inspired architecture that reduces false positives without much hyperparameter optimization or tuning of a single neural network. So the number of trials and errors in the training phase is reduced.

3 Modular neural networks

Modular Neural networks (MNNs) use the modularity principle. They deploy different techniques to achieve modularity. By using these networks, it is possible to make isolated subproblems that can be solved individually [2]. Loose coupling in MMNs allows interconnecting the components of a big neural network with the least practicable dependability which results in fault tolerance. This type of design allows scaling the model without redesigning the entire network. The recent research on the artificial neural network tries to inspire from the biological basis of these networks and emulate the modularization and segmentation in the brain. The human brain, for example, can divide the complex visual perception task into many subtasks [8].

The modularity can be used to lower the number of parameters that need to be optimized and also have a good generalization capability. Figure 1 shows a Highly-Clustered Non-Regular module (HCNR). In this kind of topology, we have dense within-module connections and the connectivity between different modules is sparse. The connections are non-regular which means the topology cannot be explained by a template containing repeating structures.

Figure 2 shows a multi-architectural topology. This kind of topology is composed of some full network architectures which are integrated using a simple algorithm at the end. Each module has its separate output and different architectures could be homogeneous or heterogeneous. The difference between modules could be limited to an implementation scheme or only the type of activation functions.

Figure 3 shows the repeated block topologies. In this type of topologies, modular neural networks are structured from repeated blocks connected in a certain way. The blocks can have slight differences but the general blueprint is the same. Figure 3a shows a multi-path topology which is neural networks with semi-independent subnetworks that connect the inputs to outputs. These networks are inspired by the microcircuitry of the retina. Figure 3b shows a modular node topology which is a traditional feedforward neural network but each node is replaced by a module with multiple neurons. This topology has the advantage of replacing a single activation function that depends on only one weight vector, with a collection of functions depending on multiple vectors. Figure 3c depicts a sequential topology. In sequential topology, similar modules are connected in series. The idea of this topology is the same as increasing hidden layers in neural networks and building deep neural networks. Figure 3d demonstrates a recursive network with nested levels of modules. Each module is defined by the earlier module in the recursion. These networks can be specifically targeted for recursive problems.

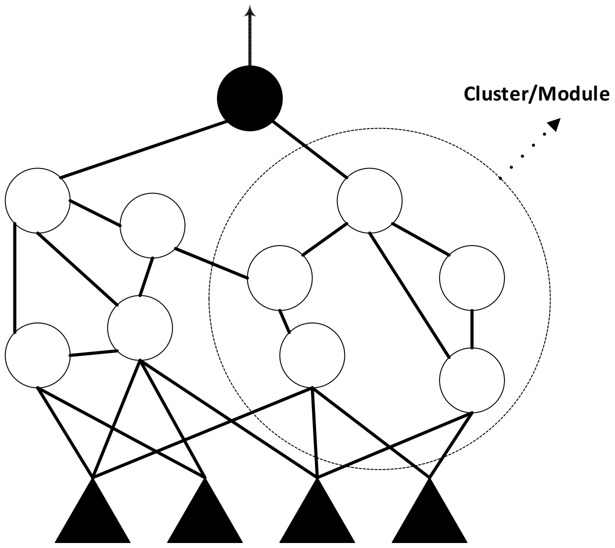


Fig. 1 Highly clustered non-regular (HCNR)

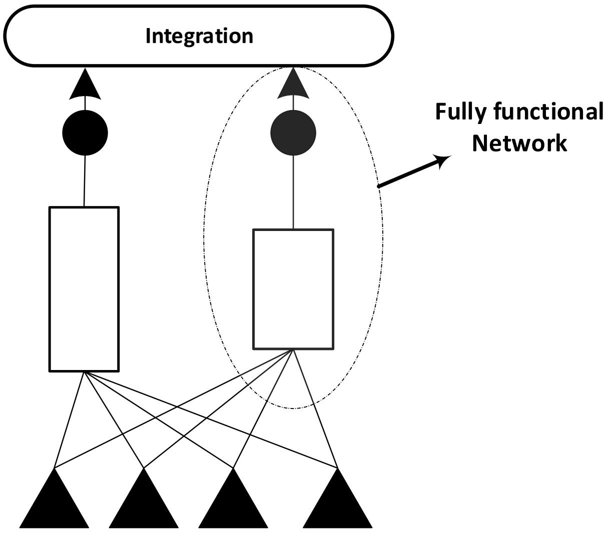


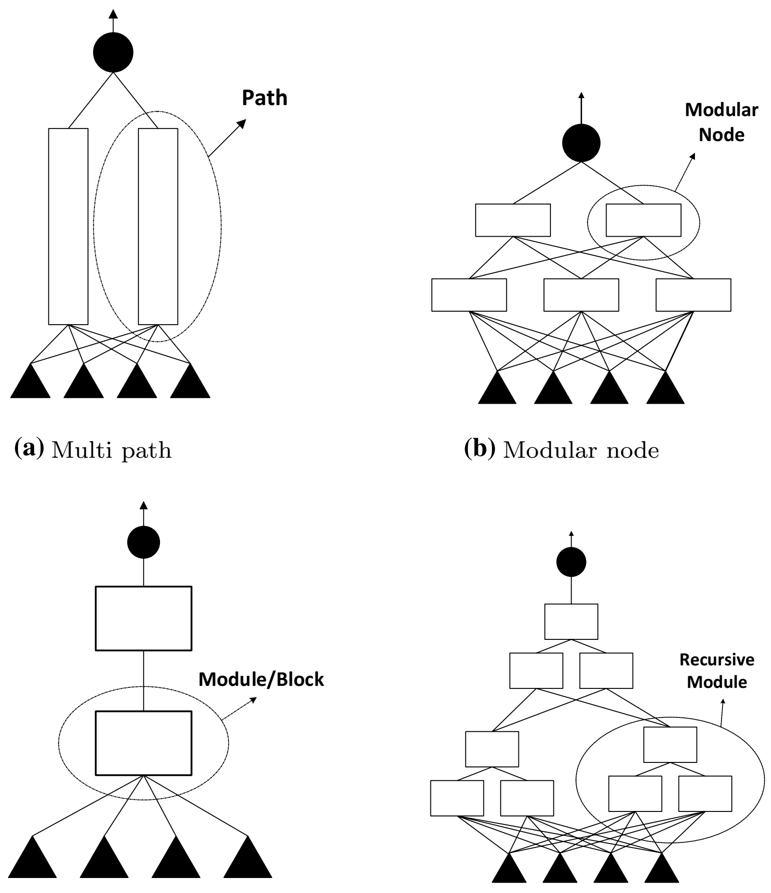
Fig. 2 Multi-architectural topology

The topology of a network suggests that how different nodes and modules in the whole network connect with each other. There is sparse connectivity between modules. If we want to assign a subtask to each different module, we need functional modularity. The human brain also tries to be evolvable and minimize the connection cost [34]. For a single neural network, it is very hard and sometimes impossible to decipher how the network makes predictions about the problem; that's because the nodes inside a single network are tightly coupled components and the functions cannot be separated from each other so the modularity can be exploited to create fault tolerance.

4 Applying multi-architectural modular deep neural network to network intrusion detection

4.1 Proposed modular deep neural network model

Our proposed modular deep neural network model has a multi-architectural topology and is a combination of 4 full architectures integrated with an aggregator module. As shown in the abstract of our model in Fig. 4, our model has 4 component networks each producing separate outputs. We used a Deep Feed-Forward Module (DFFM), A Stacked Restricted Boltzmann Machine Module (SRBMM), and two recurrent modules one with gated recurrent units (GRUM) and one with long short-term memory (LSTMM). As discussed in Sect. 2, these models perform well in the problem of network attack detection. Thus, the prediction of a deep modular network created using



(a) Multi path **(b)** Modular node

(c) Sequential **(d)** Recursive

Fig. 3 Repeated blocks

these models will be highly correlated and can find patterns that are harder to predict. So the combination of these modules can lead to more accurate results.

As depicted in Fig. 4, each of our modules is using different networks to achieve better collective performance. Although our models are different, we can still exploit modular networks with the same type of modules; that's because stochastic learning and random initialization make each module perform differently. The training time for the entire modular architecture is equal to the training time of the slowest module, which in our case is the SRBMM module. The structure of each module will be presented in the following subsections.

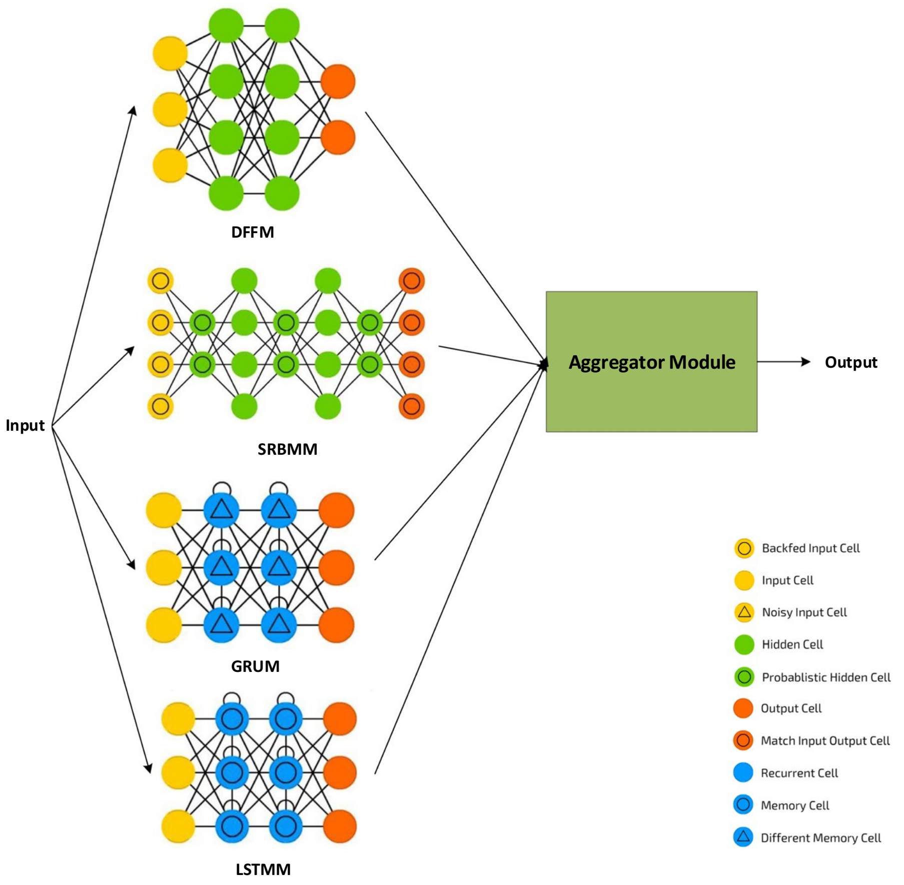


Fig. 4 Proposed modular deep neural network architecture

4.2 Stacked restricted Boltzmann machine module (SRBMM)

RBMs are also called Stochastic Neural Networks because they take a probabilistic approach. They are composed of three parts:

1. One input layer or visible unit
2. One hidden layer or hidden unit
3. Bias unit

RBM can be utilized as a generative model for unlabeled or labeled data. A visualization of an RBM is depicted in Fig. 5, as depicted in this figure, there is no intra-layer communication in RBM which is the restriction. At the first node of the hidden layer, an input value is multiplied by a weight and added to bias and the result is sent to the activation function and then the output is produced [10, 11].

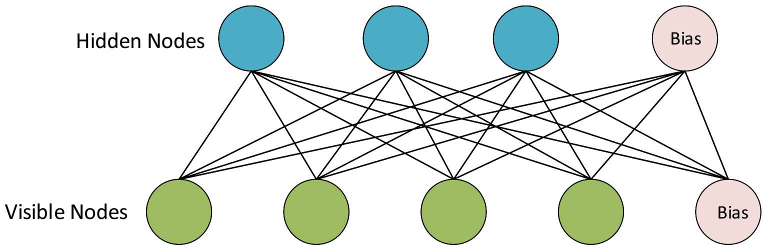


Fig. 5 Visualization of an RBM

The energy of a joint configuration of hidden and visible units is calculated as:

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{1}$$

v_i and h_j are the binary states of visible unit i and hidden unit j, a_i; b_j are their biases and w_{ij} is the weight between them. In order to calculate the contribution of biases and the hidden or visible units, the dot product of them is calculated. The probability of every possible pair of hidden and visible vector is then calculated using energy function in:

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{2}$$

In Eq. 2, Z is the partition function and formulated as follows:

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

The probability that the network assigns to a visible vector is calculated as:

$$p(v) = \frac{1}{Z} \sum_h e^{-E(v,h)} \tag{4}$$

The derivative of the log probability of a training vector with respect to weight is calculated as shown in Eq. 5.

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \left\langle v_i h_j \right\rangle_{data} - \left\langle v_i h_j \right\rangle_{model} \tag{5}$$

Which leads to a very simple rule with ϵ as learning rate:

$$\Delta w_{ij} = \epsilon(\left\langle v_i h_j \right\rangle_{data} - \left\langle v_i h_j \right\rangle_{model}) \tag{6}$$

Geoffrey Hinton proposed a much faster learning procedure in 2002 in which a reconstruction is produced by setting each v_i to 1:

$$\Delta w_{ij} = \epsilon(\left\langle v_i h_j \right\rangle_{data} - \left\langle v_i h_j \right\rangle_{reconstruction}) \tag{7}$$

This module has the highest training time among all the modules; thus, the overall training time of our model is equal to the training time of the SRBMM module.

The value of network traffic features in our used dataset corresponds to ”visible” units of the RBM since they can be observed.

The pre-training step involves a stack of RBMs with contrastive divergence algorithm. The extracted features from first complete representation RBM are used for training in the next RBM, and finally, back-propagation of errors is performed.

In our case is not possible to use Eq. 2 in our approach. We have to replace binary visible units by linear units (with independent Gaussian noise). Gaussian RBM assumes there are real-valued visible units between 0 and 1 and so the function then becomes:

$$E(v,h) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij} \tag{8}$$

The purpose of training RBM is to update our weights and biases to maximize the likelihood of learning rule by gradient descent. The weights are initialized to small random values. These values are chosen from a zero-mean Gaussian with a standard deviation of about 0:01. For determining a good number of hidden units, we follow Hinton’s approach. Considering the number of labels in our case, 6 layers of RBMs are used to get the most satisfying results. A learning rate decay value of 0.9 is used which will be multiplied by initial learning rate after each repeat of the training and the fine-tuning step is done for 5 epochs.

4.3 Deep feed-forward module (DFFM)

In a fully connected module, the neurons receive some inputs, perform dot products, and use a nonlinear function such as Sigmoid, Tanh, ReLU, Leaky ReLU, Parametric ReLU, or other nonlinear functions. These functions allow models to make complex mappings between inputs and outputs of the network. This is important for datasets with high dimensionality. Figure 6 depicts a sample FNN. In this figure, every neuron use a nonlinear activation function and the last layer will use softmax.

In a forward pass of FNN, a set of operations that transform input to output are performed. Activation functions are used for introducing nonlinearity into the network and learning more complex operations. The backward pass is used if an incorrect output is reached. In backpropagation, error gradients concerning neuron weights and biases are calculated. Cross-entropy loss or log loss can measure the performance of the classification model with outputs between 0 and 1. The loss function can be a differentiable mathematical expression.

If x ∈ R represents the input to the fully connected layer, $y_i \in$ R is the i-th output from our fully connected layer. $y_i \in$ R is thus computed as:

$$y_i = \sigma(w_1 x_1 + w_2 x_2 + \dots + w_m x_m) \tag{9}$$

The nonlinear σ function in our method is a rectifier to stop vanishing gradient problem:

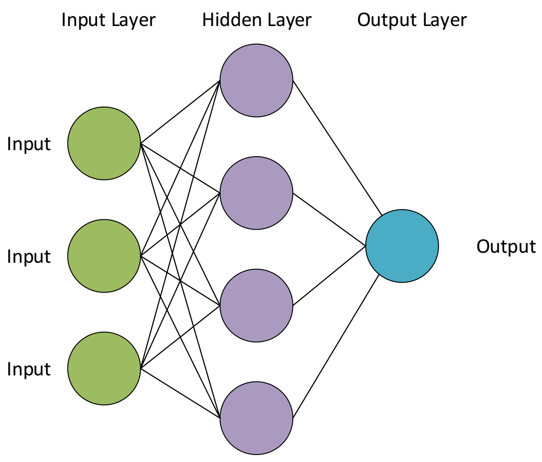


Fig. 6 Network architecture of FNN

$$\sigma = max(0, x) \tag{10}$$

DFFM can be trained much faster than SRBMM. So a deep network with 12 hidden layers is used. We use batch normalization with a batch size of 10000 instead of a dropout technique to get more accurate results, and the gradient of the full dataset will have a stable estimate. The batch size only changes the speed of learning and does not affect the quality of learning. To prevent our model from memorizing the attacks, early stopping is also used. We use the value of 0.01 as the learning rate; choosing a smaller value can lead to longer training time and higher values make the training unstable. Adam optimization algorithm which is an extension to stochastic gradient descent is used for updating the network weights, and this algorithm is more effective than algorithms like AdaGrad or RMSProp in our scenario. Categorical cross-entropy is used in our multi-class scenario.

4.4 Recurrent modules

Recurrent layers have some loop units which allow them to persist information. In the human brain, each concept is understood from previous concepts and we don't have to start thinking from the starting point. The DFFM and SRBMM modules do not have this capability. In a recurrent layer, the output from the last stage is fed as input to the current stage. Although the recurrent network has these advantages, they still suffer from gradient vanishing and exploding problems. The training time of RNN depends on the way of implementation, but due to the slow training time of these networks, we used 6 hidden layers in these modules.

In RNN, the same weights and biases are provided to all the layers to convert the independent activations into dependent activations. The current state can be calculated using the below formula.

$$h_t = f(h_{(t-1)}, x_t) \tag{11}$$

where h_t, h_{t-1} and x_t are current, previous and input state, respectively. The formula for applying tanh activation function is:

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{12}$$

where w_{hh} and w_{xh} are weight at recurrent and input neuron, respectively, and the formula for calculating the output is:

$$y_t = W_{hy}h_t \tag{13}$$

where Y_t is output and W_{hy} is the weight at the output layer.

When training an RNN first, a single time step of the input is present to the network. In the next step, the current state is calculated using the current input and the previous state. The value of h_t will become h_{t-1} for the next time step. After the completion of all needed time steps, the final state is used to calculate the output. After the output is generated, it is compared to true output and an error is generated. The error can be back-propagated to the network to update the weights.

We use two variants of RNN in our work to make the two GRU and LSTM modules. Long short-term memory (LSTM) is augmented by forget recurrent gates. This system can avoid the vanishing and exploding gradient problem. Figure 7 shows a long short-term memory unit.

Gated recurrent units (GRUs) are introduced in 2014 and have fewer parameters than LSTM because they do not have an output gate. They use the update gate and reset gate to solve the vanishing gradient problem in vanilla RNN. Figure 8 shows a Gated recurrent unit.

The reset gate and update gate vectors decide what information must be passed to the output. They can keep information that is irrelevant to the prediction.

4.5 Aggregator module

Our aggregator module uses a weighted averaging technique to produce the output of the modular network based on 4 inputs. The weighted averaging enables us to

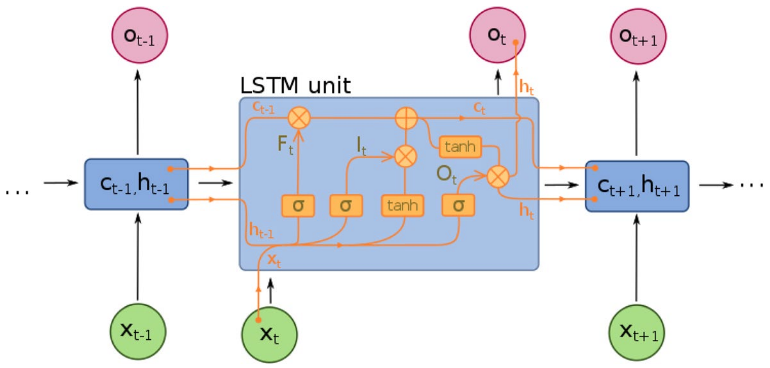


Fig. 7 Long short-term memory unit

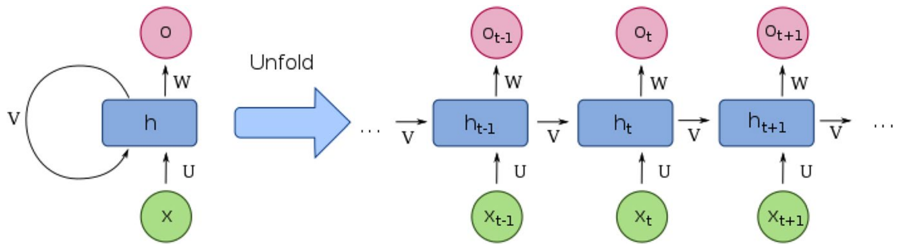


Fig. 8 Gated recurrent unit

weight predictions proportional to the trust we have for our implemented model. This method averages out the various errors of the individual modules and relies on the following properties of artificial neural networks:

- In any neural network, the bias can be decreased at the expense of increased variance.
- In a collection of neural networks, the variance can be decreased at no expense to bias.

The method for the implementation of the aggregator module is presented in Algorithm 1. In this algorithm, (a) is a set of weights and we can find their optimized values using a neural network and each expert is defined as y_i and the overall result is defined as $\bar{y}$.

Algorithm 1 Model Averaging, (a) is a set of weights and each expert is defined as y_i and the overall result is defined as $\bar{y}$

```
$expert = y_i$
$result = \bar{y}$
$\bar{y}(x; a) = \sum_{i=1}^{p} a_i y_i(x)$
```

The provided model averaging algorithm is an extension of the simple model averaging technique. In this algorithm, we can give a higher weight to the results of that particular module proportional to the robustness of that module. The weight (a) of a module has a positive value, and the sum of all weights are equal to one. So, the weights of a skillful module can contribute more to the results. If we want to use a raw average, we can set all a_i values equal to some constant value.

Algorithm 1 is implemented using python and the sci-kit-learn library.

4.6 Dataset description

The CSE-CIC-IDS2018 dataset is created by the same organization that created the CICIDS2017 dataset. This dataset contains more up to date and number of attacks and normal records compared to the CICIDS2017 dataset. The attacks in this dataset include Brute-force, Heartbleed, Botnet, Denial-of-Service, Distributed Denial-of-Service, Infiltration of the network from inside, and Web Attacks. The dataset is organized per day. In each day, the PCAP files and operating system event logs are recorded per machine, and CICFlowMeter-V3 is used to extract more than 80 network traffic features as a CSV file. CSV files can be loaded easily to a pandas data frame and PCAP and Log files can be used to extract new features from traffic. Table 1 depicts a list of executed attacks and the duration of each attack.

Table 2 shows some features of this dataset.

4.7 Dataset preprocessing

The first step to work with the CSE-CIC-IDS2018 dataset is to remove socket information. To make our predictions unbiased toward certain socket information, it is necessary to remove the information such as IP address and port numbers of the source hosts but the destination port number can be useful in the detection of some type of attacks. Also, the labels in the dataset have string values and it is necessary to encode them into numerical values corresponding to each label. Before we start feeding our dataset to our neural network, we want the features to be properly scaled so that we do not have some features dominating others. The range of 0 and 1 is suitable as we have a stack of RBMs module in our model. The rows with missing values and the columns with too much missing values are also dropped from the dataset.

5 Experimental results

In this section, we discuss the experimental results of applying our model to the intrusion detection dataset.

The performance of an IDS is generally calculated using True Positive (TP) or Hit, True Negative (TN) or Rejection, False Positive (FP), False Alarm or Type I error, and False Negative (FN), miss or Type II error [9]. True Positive (TP) is the number of correctly classified attacks. True Negative (TN) is the number of correctly classified benign records. False Positive (FP) is the number of incorrectly classified benign records. False Negative (FN) is the number of incorrectly classified malicious records. Performance is then evaluated in terms of the following equations:

Recall:

$$(TP/(TP+FN) \tag{14}$$

Precision:

$$(TP/(TP+FP) \tag{15}$$

Accuracy:

$$(TN+TP)/(TN+TP+FN+FP) \tag{16}$$

F1 score or F-measure (It considers both the precision and recall to compute the score):

Table 1 List of executed attacks and duration

Attack	Tools	Duration	Attacker	Victim
Bruteforce attack	FTP-Patator SSH-Patator	One day	Kali linux	Ubuntu 16.4 (web server)
DoS attack	Hulk, GoldenEye, Slowloris, Slowhttptest	One day	Kali linux	Ubuntu 16.4 (Apache)
DoS attack	He art leech	One day	Kali linux	Ubuntu 12.04 (Open SSL)
Web attack	Damn vulnerable web app (DVWA) In-house selenium framework (XSS and Brute-force)	Two days	Kali linux	Ubuntu 16.4 (Web Server)
Infiltration attack	First level: Dropbox download in a windows machine Second Level: Nmap and portscan	Two days	Kali linux	Windows Vista and Macintosh
Botnet attack	Ares {developed by Python): remote shell, file upload/download, capturing screenshots and key logging	One day	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)
DDoS+PortScan	Low Orbit Ion Canon (LOIC) for UDP, TCP, or HTTP requests	Two days	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)

Table 2 Example features of CSE-CIC-IDS2018 dataset and description

Feature name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	Total size of packet in forward direction
fw_pkt_l_max	Maximum size of packet in forward direction
fw_pkt_l_min	Minimum size of packet in forward direction
fw_pkt_l_avg	Average size of packet in forward direction
fw_pkt_l_std	Standard deviation size of packet in forward direction
Bw_pkt_l_max	Maximum size of packet in backward direction
Bw_pkt_l_min	Minimum size of packet in backward direction
Bw_pkt_l_avg	Mean size of packet in backward direction

$$(2TP)/(2TP+FP+FN) \tag{17}$$

Some researchers also use the term Detection Rate which is simply 1 minus the False Negative Rate. The experiments are performed using intel Xeon CPU with 4 cores, 2.30GHz clock speed, 46MB cache, and 24GB of RAM. The train and test dataset are generated using 20:80 Stratified sampling of each subset. The results are compared in terms of accuracy, precision, recall, and F1 score with related comprehensive work at [3] (only the same implementations are compared) and in terms of train accuracy with best results of similar work at [23]. Table 3 shows the different accuracy metrics of our network for CSE-CIC-IDS2018 datasets using the MNN approach for different types of network attacks.

The evaluation results include 20:80 stratified sampling of the datasets. The results of modular network classification are presented by the 4 most used performance measures. As presented in Table 3, we can see improvements in detection rate in all attack types except for web attacks compared to related work. One reason for this can be improper web attack features in the CSE-CIC-IDS2018 dataset. Most extracted features in the CSE-CIC-IDS2018 dataset are related to network properties, and we need more content features to increase the detection rate in the web attacks subset. Thus, the modular architecture cannot improve web attack results. The infiltration attacks also can be detected better using signatures in raw traffic. The aggregator module can remove some false negatives from Botnet and DoS attack, so the performance measures are improved compared to related work.

Figures 9, 10, 11, 12, 13 and 14 depict the confusion matrix for BruteForce, Web, Bot, DoS, DDoS and Infiltration, respectively. These figures describe the performance of our classification model. The confusion matrix is a performance measurement for our classification problem. It shows different combinations of predicted and actual values. For Brute-force attacks, classes 1, 2, and 3 represent Benign, FTP-BruteForce and SSH-Bruteforce, respectively, and for Web attacks, classes 1, 2, 3, and 4 represent Benign, Brute Force-web, Brute Force -XSS and SQL Injection, respectively. Most instances belong to Benign class so we have an imbalanced

Table 3 Accuracy comparison of proposed MNN model using CSE-CIC-IDS2018 datasets

Attack type	Accuracy (%)	Precision	Recall	F1 score
Botnet (MDNN)	99.98, 99.98	1.0, 1.0	1.0 , 1.0	1.0, 1.0
Botnet (DNN [3])	99.88, 99.88	1.0, 1.0	1.0, 1.0	1.0, 1.0
Botnet (RF [23])	99.0	N/A	N/A	N/A
Denial-of-Service (MDNN)	100, 100, 100	1.0, 1.0, 1.0	1.0, 1.0, 1.0	1.0,1.0,1.0
Denial-of-Service (DNN [3])	99.94, 99.94, 100	1.0, 1.0, 1.0	1.0, 1.0, 1.0	1.0 ,1.0, 1.0
Denial-of-Service (RF [23])	99.0	N/A	N/A	N/A
Distributed Denial-of-Service (MDNN)	100, 100, 100,	1.0, 1.0, 1.0	1.0, 1.0, 1.0	1.0, 1.0, 1.0
Distributed Denial-of-Service (DNN [3])	N/A	N/A	N/A	N/A
Distributed Denial-of-Service (RF [23])	99.0	N/A	N/A	N/A
Web Attacks (MDNN)	100, 100, 100, 100	1.0, 0.46, 0.0, 0.0	1.0, 0.58, 0.0, 0.0	1.0, 0.51, 0.0, 0.0
Web Attacks (DNN [3])	N/A	N/A	N/A	N/A
Web Attacks (RF [23])	99.0	N/A	N/A	N/A
Infiltration (MDNN)	82.83, 82.83	1.0, 0.012	0.83, 0.52	0.91, 0.024
Infiltration (DNN [3])	N/A	N/A	N/A	N/A
Infiltration (RF [23])	100	N/A	N/A	N/A
Brute-Force (MDNN)	100, 100, 100	1.0, 1.0, 1.0	1.0, 1.0, 1.0	1.0, 1.0, 1.0
Brute-Force (DNN [3])	99.99, 100, 99.99	1.0, 1.0, 1.0	1.0, 1.0, 1.0	1.0, 1.0, 1.0
Brute-Force (RF [23])	99.0	N/A	N/A	N/A

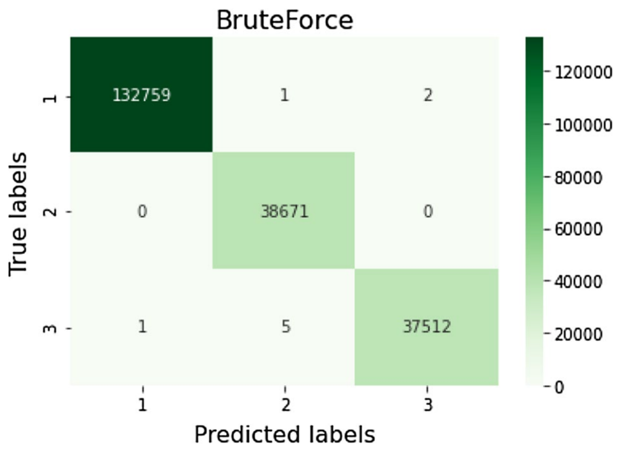


Fig. 9 Confusion matrix for BruteForce attacks

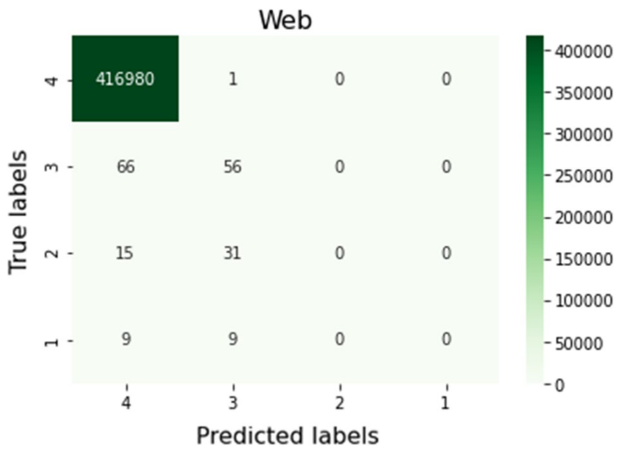


Fig. 10 Confusion matrix for web attacks

dataset. More instances are displayed with a darker green color. We can see the combinations of predicted and actual values in this matrix and decide whether to change our model or not. By looking at the confusion matrix, the number of type 1 and type 2 errors can be understood. The results show a very low false-positive rate for Bot, DoS, DDoS, and BruteForce attacks as can be seen by high elements in diagonal and near-zero elements in off-diagonal of confusion matrixes. If fewer errors are present, then the model is less confused and is making true predictions. Confusion matrix thus can give more valuable information than accuracy measures about the performance of our model. The diagonal elements show the number of correct classifications and off-diagonal elements show incorrect classifications. As depicted in these figures, almost all network-level attacks are detected with our proposed method.

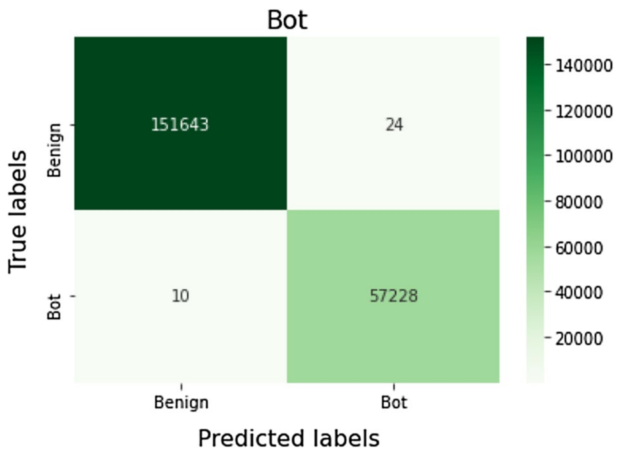


Fig. 11 Confusion matrix for Bot attacks

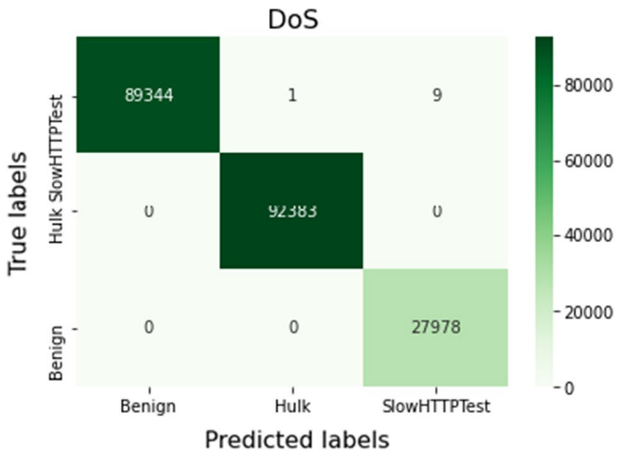


Fig. 12 Confusion matrix for DoS attacks

6 Conclusion

In this paper, we have designed a modular deep neural network model to detect intrusions in network traffic. To tackle the problem of a high false-positive rate which is not suitable for a real-world implementation of an IDS, we used the concept of modularity in neural networks inspired by the human brain. The experimental results show that our model can decrease false alarms in some type of intrusions and achieve accuracies as high as 100% compared to monolithic neural networks. Extracting features that can help to identify Nmap or Metasploit traffic can be a new research topic for anomaly-based intrusion detection researches. For future work, we plan to make a custom feature extractor and build a custom

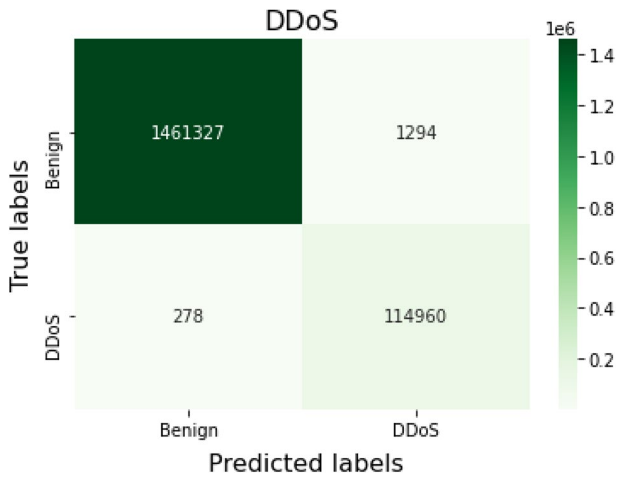


Fig. 13 Confusion matrix for DDoS attacks

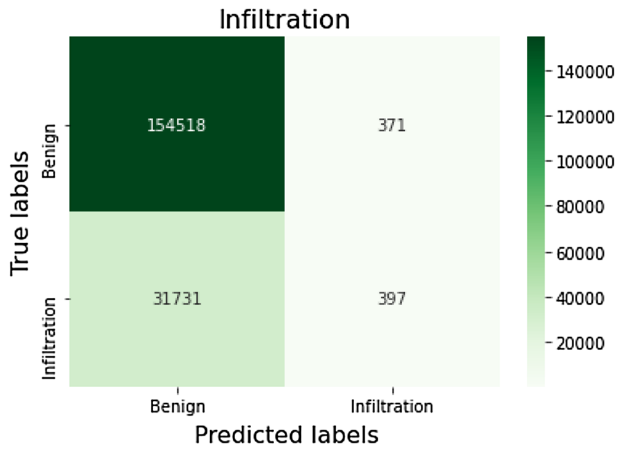


Fig. 14 Confusion matrix for infiltration attacks

dataset for our purpose and enhance training time using parallel and big data frameworks such as Apache Spark.

Acknowledgements This research has been supported by the Computer Emergency Response Team at Razi University (Razi CERT) under grant number 97P101.

References

1. Al-Yaseen WL, Othman ZA, Nazri MZA (2017) Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system. Expert Syst Appl 67:296–303

2. Amer M, Maul T (2019) A review of modularization techniques in artificial neural networks. Artif Intell Rev 52(1):527–561
3. Basnet RB, Shash R, Johnson C, Walgren L, Doleck T (2019) Towards detecting and classifying network intrusion traffic using deep learning frameworks. J Internet Serv Inf Secur 9(4):1–17
4. Chen CM, Chen YL, Lin HC (2010) An efficient network intrusion detection. Comput Commun 33(4):477–484
5. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:14123555
6. De la Hoz E, Emiro DLH, Andres O, Julio O, Beatriz P (2015) PCA filtering and probabilistic SOM for network intrusion detection. Neurocomputing 164:71–81
7. de Lima Filho FS, Silveira FA, de Medeiros Brito Junior A, Vargas-Solar G, Silveira LF (2019) Smart detection: an online approach for DoS/DDoS attack detection using machine learning. Security and Communication Networks 2019
8. Dong B, Wang X (2016) Comparison deep learning method to traditional methods using for network intrusion detection. In: 8th IEEE International Conference on Communication Software and Networks (ICCSN), pp 581–585
9. Govindarajan M, Chandrasekaran R (2011) Intrusion detection using neural based hybrid classification methods. Comput Netw 55(8):1662–1671
10. Happel BL, Murre JM (1994) Design and evolution of modular neural network architectures. Neural Netw 7(6–7):985–1004
11. Heberlein LT (2007) Statistical problems with statistical based intrusion detection. Tech. rep., Version1, Net Squared, Inc
12. Hinton GE (2012) A practical guide to training restricted boltzmann machines. In: Neural networks: tricks of the trade, pp 599–619
13. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief networks. Neural Comput 18(7):1527–1554
14. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
15. Hodo E, Bellekens X, Hamilton A, Dubouilh PL, Iorkyase E, Tachtatzis C, Atkinson R (2016) Threat analysis of iot networks using artificial neural network intrusion detection system. In: International Symposium on Networks, Computers and Communications (ISNCC), pp 1–6
16. Hsu CM, Hsieh HY, Prakosa SW, Azhari MZ, Leu JS (2018) Using long-short-term memory based convolutional neural networks for network intrusion detection. In: IEEE International Wireless Internet Conference, pp 86–94
17. Iqbal A, Aftab S (2019) A feed-forward and pattern recognition ann model for network intrusion detection. Int J Comput Netw Inf Secur 11(4):19–25
18. Javaid A, Niyaz Q, Sun W, Alam M (2016) A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), pp 21–26
19. Karatas G, Demir O, Sahingoz OK (2020) Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset. IEEE Access
20. Kevric J, Jukic S, Subasi A (2017) An effective combining classifier approach using tree algorithms for network intrusion detection. Neural Comput Appl 28(1):1051–1058
21. Lee S (2004) Hierarchical neural network intrusion detector. US Patent App. 10/433,713
22. Lin P, Ye K, Xu CZ (2019) Dynamic network anomaly detection system by using deep learning techniques. In: International Conference on Cloud Computing, pp 161–176
23. Lypa B, Iver O, Kifer V (2019) Application of machine learning methods for network intrusion detection system
24. Marir N, Wang H, Feng G, Li B, Jia M (2018) Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark. IEEE Access 6:59657–59671
25. Paxson V (1999) Bro: a system for detecting network intruders in real-time. Comput Netw 31(23–24):2435–2463
26. Rios ALG, Li Z, Bekshentayeva K, Trajkovic L (2020) Detection of denial of service attacks in communication networks
27. Roesch M (1999) Snort: lightweight intrusion detection for networks. In: LISA '99: Proceedings of the 13th USENIX Conference on System Administration, vol 99, pp 229–238
28. Sahu S, Mehtre BM (2015) Network intrusion detection system using j48 decision tree. In: 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp 2023–2026

29. Saraswati A, Hagenbuchner M, Zhou ZQ (2016) High resolution som approach to improving anomaly detection in intrusion detection systems. In: AI 2016: Advances in Artificial Intelligence, pp 191–199
30. Shams EA, Rizaner A (2018) A novel support vector machine based intrusion detection system for mobile ad hoc networks. Wireless Netw 24(5):1821–1829
31. Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell 2(1):41–50
32. Singh Panwar S, Raiwani Y, Panwar LS (2019) Evaluation of network intrusion detection with features selection and machine learning algorithms on CICIDS-2017 dataset. Available at SSRN 3394103
33. Song H, Woo J, Li FF (2019) In-vehicle network intrusion detection using deep convolutional neural network
34. Sporns O, Betzel RF (2016) Modular brain networks. Annu Rev Psychol 67:613–640
35. Ullah I, Mahmoud QH (2019) A two-level hybrid model for anomalous activity detection in IoT networks. In: 2019 16th IEEE Annual Consumer Communications and Networking Conference (CCNC), pp 1–6
36. Xiao Y, Xing C, Zhang T, Zhao Z (2019) An intrusion detection model based on feature reduction and convolutional neural networks. IEEE Access 7:42210–42219
37. Zhou Q, Pezaros D (2019) Evaluation of machine learning classifiers for zero-day intrusion detection: an analysis on CIC-AWS-2018 dataset. arXiv preprint arXiv:190503685