A Hybrid Model based on Radial basis Function Neural Network for Intrusion Detection

Marwan Albahar[1],Ayman Alharbi[2], Manal Alsuwat[3], Hind Aljuaid[4]
Umm Al Qura University[1,2], Taif University[3,4]

***Abstract*—An Intrusion Detection System (IDS) is a system that monitors the network for identifying malicious activities. Upon identifying the unusual activities, IDS sends a notification to the network administrators to warn about the hackers' hostile activities. To detect intrusion, signature-based systems are considered to be one of the most effective methods. However, they cannot detect new attacks. Additionally, it is costly and challenging to keep the attack signatures database up to date with known signatures, which constructed a significant drawback. Neural networks are capable of learning through input patterns and have the potential to generalize data. In this paper, we propose a hybrid model based on Directed Batch Growing Self-Organizing Map (DBGSOM) combined with a Radial Basis Function Neural Network (RBFNN) detecting abnormalities in the network. Based on our experiment, the proposed model performed well and has resulted in satisfactory performance measures compared to Self-Organizing Maps and Radial Basis Function Neural Network (SOM&RBFNN) model.**

Keywords*—Intrusion detection; neural network; radial basis function; directed batch growing self-organizing map

I. Introduction

The immense flow of network traffic has created opportunities for attackers to breach privacy and harm the integrity of the network and its users. In such circumstances, intrusion detection has become a crucial part of computer security to ensure that attacks and intrusion activities can be detected [1]. Intrusions are threats to network systems that may come in different forms, such as damaging the systems and making it unavailable, information examination, and information manipulation [2]. There are two kinds of intrusions passive and active, where passive intrusions are surreptitiously and without detection, whereas active intrusions lead to change and effect to network resources. Intrusion Detection System (IDS), an influential approach, is designated to detect normal and malicious activities in computers. It is a system that defends the network by identifying suspicious activities while analyzing the network traffic to detect and counter intrusions [2] timely. An IDS has two main methods of detecting intrusions: signature-based IDS and anomaly-based IDS. Signature-based detection is utilized to searching for a "signature" pattern or known attacks. This type of IDS, it requires regular updates to currently common signatures or identities to ensure that the intruders' database is current. Nevertheless, attackers can change small things in signatures, so the databases cannot recognize it. Consequently, a new attack type may not be picked up because the signature doesn't exist in the database. Moreover, the larger the databases, the more processing to analyze each connection and verify it. In contrast to signature-based IDS, anomaly-based detection is employed to detect known and unknown attacks depending on learning their behavior by specifying observations that deviate from a basic model in a computer network and inform the network's administrator to take necessary actions. The main advantage of anomaly-based detection is the ability to detect unknown attacks [1,3]. As a result of their importance, different systems have been proposed for intrusion detection by many researchers. Among these systems, machine learning models and specifically neural networks can effectively detect malicious activities on a network by getting trained using enough intrusion detection recorded data [4-7]. Non-neural network machine learning models such as SVM have specific limitations such as low repetition attack detection rates, detection instability, and training process complexity [7]. Neural network models have been used for anomaly detection, such as autoencoders and variational autoencoders (VAEs). Autoencoders are composed of encoder and decoder networks that are sequentially connected. An encoder can compress the input data and a decoder to reconstruct the input data. Autoencoders attempt to reduce the error in reconstruction (i.e., the difference between the decoder output and the original input). This error is used as an anomaly score to detect anomalies [8-9]. Small reconstruction errors correspond to normal data, while larger reconstruction errors correspond to anomalous data [8]. A VAE, which is a directed probabilistic graphical model (DPGM), combines Bayesian inference with the autoencoder framework. It provides a probability measure rather than an error of reconstruction as an anomaly score. As probabilities are more principled and objective than reconstruction errors and do not require model specific thresholds for anomalies to judge, VAEs outperform autoencoders in intrusion detection. However, for natural inference and learning, VAE assumes complicated data distributions can be modeled with a smaller group of latent variables whose probability density distributions are Gaussian. At the same time, studies show the data distribution is usually multi-modal, and a single Gaussian distribution cannot be considered before the latent space [8]. As a result, some other methods, such as the GGM-VAE method are proposed to compensate for this shortcoming. In GGM-VAE, gated recurrent unit (GRU) cells are used to find the correlations between data. Then, the GGM-VAE method uses Gaussian Mixture prior in the latent space to characterize the multimodal data using a series of Gaussian distributions and applies VAE. While GGM-VAE yields a better detection rate over VAE, similar to previously discussed deep networks, it needs extensive data for training. On the other hand, Radial basis function neural networks (RBFNNs), which are capable of classifying patterns in nonlinear space and linear encoding, can estimate non-parametric multi-dimensional functions through a limited set of educational information [7]. Also, RBFNNs are rapid, comprehensive, and yields highly accurate in training [7]. New research by Mohammadi and Amiri also verifies their effectiveness in detecting intrusion in the networks [7]. This research first uses self-organizing maps (SOMs) to cluster the data. It then uses the cluster centers to determine the number of radial basis functions and set the parameters of these functions in RBFNN. SOMs can profoundly reduce the dimensionality of the data and therefore reduce the computational runtime of the process. However, one issue with SOMs is that they need a predefined structure, including determining the number of clusters/neurons. To handle this limitation growing SOMs (GSOMs) and researchers present its variations. One of the recent variations of GSOMs is directed batch GSOMs (DBG-SOMs) which not only resolve the limitation of SOMs but also introduces a suitable mechanism to discover an appropriate growth position and assigning new neurons initial weight vectors [7]. A batch learning strategy for growing self-organizing maps was employed in DBGSOM to resolve SOM and GSOM models' issues. Consequentially, DBGSOM has a better ability to conservation the topology and reduce exposure for twisting and tangling in the network [10]. Furthermore, RBFNNs are rapid and comprehensive in training and produce high precision in detecting intrusion [7]. In this study, we measure the effectiveness of combining DBGSOM and RBFNN to build an effective anomaly-based intrusion detection system. Then, we compare its efficacy to SOM with RBFNN. We first explain SOMs and its limitations and then discuss DBGSOM before representing the RBFNN process. Then, we apply our proposed method on three publicly available datasets. Finally, based on our experimental results, we show that our hybrid model outperforms the SOM&RBFNN model.

The paper proceeds as follows: In Section II, we provide related works. In Section III, we describe DBGSOMs and RBFNN used in this work. In Section IV, we introduce our proposed model. Next , we introduce the dataset utilized in Section V. In Section VI, we shows our experimental results, and Section VII concludes and gives our future works.

II. Related Work

There is a large number of published researches that prove the effectiveness and ability of machine learning especially neural networks in intrusion detection techniques, some of them will be summarized in the following: A recent study by Vinayakumar et al. [11], shows convolutional neural networks (CNNs) and its various architectures styles generally perform good efficiency compared to classical machine learning classifiers [12]. They modeled network traffic with various learning methods including multi-layer perceptron (MLP), CNN, CNN-recurrent neural network (CNN-RNN), CNN-long short-term memory (CNNLSTM) and CNN-gated recurrent unit (GRU) and reported their results for the NSL-KDD dataset. Their reported detection rate seems promising. However, since these methods need millions of known good and bad network connections for training, and obtaining a right deep model usually needs trying complex architecture, which they consequently require more computational cost, they cannot be yet the best option to be employed for the job. In another study, Li et at. used a representation learning method of graphic conversion to transform intrusion data into image shape and then apply CNN on the transformed features to detect anomalies [12]. In this study, they, also, only applied their method on the NSL-KDD dataset and achieved Impressive results. However, clearly the converting the data to the image form proposed is time-consuming and the method also needs huge data for training to perform well.

Researchers in [1] applied a long short-term memory (LSTM) model to discover intrusion and utilized the CIDDS001 dataset for assessing the LSTM model's performance and they discovered that it outperforms on SVM, MLP, and Naïve Bayes techniques concerning to multiclassification problem. Regarding the self-organization map (SOM), Sadeq and Ahmad studied the effectiveness of combining the SOM with a backpropagation neural network (BPNN) to reveal intrusion systems [2]. The proposed approach is divided into two-stage. In the first classification stage, SOM was utilized for categorizing normal traffic from attacks. Then, in the second classification stage, BPNN is utilized for categorizing the attacks into DoS, R2L, Probe, and U2R. They applied this proposed approach for dataset NSL KDD and the outcomes showed that the performance and precision of the intrusion detection system had improved.

In 2019, JIN et al. [13], proposed a new model based on the simple recurrent unit (SRU) and deep convolutional generative adversarial networks (DCGANs) to detection intrusion in the network. DCGAN was utilized to extract features from the raw-data directly and then create sufficient and balanced data samples from current attack samples. In addition to retaining the information that appeared in raw sample data. Because of the dependency that exists in LSTM, it has been replaced by SRU to allow the parallelization. Which led to improved speed of training 10 times than LSTM. Extensive experiments have been done on the dataset NSL-KDD to verify the efficiency of this model and it achieved satisfactory results in classification performance and accuracy in intrusion detection.

In addition, a previous study by Alia et al. [14], a radial basis function (RBF)-based intelligent intrusion detection system within the framework of approximation theory was applied to the dataset NSL-KDD through the k-means algorithm to detect the attack and classify its type under one category: DoS, Probe, R2L, and U2R. The results of attack classification were high for rare attacks (R2L and U2R) whereas low for frequent attacks (DoS and Probe).

In [7], a hybrid self-learner intrusion detection model was proposed using self-organized neural network algorithms and perceptron networks. The authors highlighted the advantages of the hybrid model, such as the self-learner ability of intrusion detection and memory storage. The authors also pointed out that RBF neural networks can learn quicker and more comfortable than MLP networks. Moreover, the authors showed the error rate of the proposed algorithm with the back error propagation method has a lower error rate, which reveals its higher accuracy than back error propagation method.

III. Background

A. Directed Batch Growing Self-organizing Maps (DBG-SOMs)

SOMs are the foundation of some machine learning models in anomaly-based intrusion detection systems [6, 7]. This is because SOMs are unsupervised and do not need any user feedback, while their output maps can be visualized and help understand how data are clustered [6]. SOM is also an excellent tool for dimension reduction as the SOMs' weights can be considered as cluster centers and be good representatives of data. There are two necessary steps to implement the SOM algorithm: initially, finding the winner by competition between neurons. After that, the weight vector is adapted to the winning neurons and their topological neighbors. One of the main limitations for SOM is needing a pre-defined structure and learning parameter before initializing the training process. To resolve this issue, growing SOMs (GSOMs) were introduced, which require fewer epochs, offer a flexible structure, and allow the capability to learn the nonlinear manifolds at high dimensional feature space [10, 15]. Three main aspects in which the GSOM models are different are determining when and where to add new neurons and assigning proper weight vectors. Fig. 1 shows the initialization of the GSOM network that begins with four neurons at a rectangular network, guaranteeing that all neurons have a similar lattice state in the initialization stage. GSOM models fill all available positions around the candidate

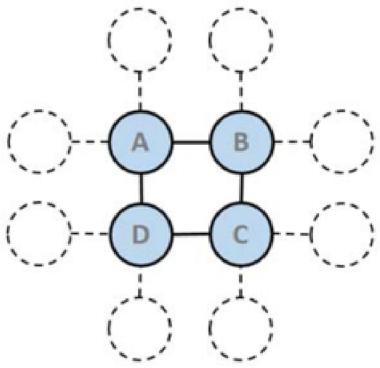


Fig. 1. Initial Structure of GSOM with Four Neurons A, B, C and D on a Square Grid [15].

neuron. Consequently, because of misconfiguration and map twisting that can result from unexpected network growth and incorrect neuron addition and the initialization of weight, they reduce the topology preservation quality of the map despite their dynamic nature. A new variation of the GSOM model called directed batch GSOMs (DBGSOMs) [15] is recently proposed to resolve the issues of GSOM models by introducing a batch learning strategy for GSOMs. DBGSOMs have a better ability to conservation the topology and reduce exposure for twisting and tangling. This model directs the network growth appropriately within the feature space by looking at the cumulative error around candidate boundary neurons. As a result, DBGSOM offers appropriate mechanisms for finding a suitable growth position and assign primary weight vectors to new neurons; moreover, it has been verified that it has a better clustering capability than GSOM and SOM. It requires less time for learning data points manifold compared to GSOM because it generates fewer neurons [15].

B. Radial Basis Function Neural Network (RBFNN)

RBFNN is a forward network type based on function approximation theory. RBFNN has consisted of input, hidden, and output layers (Fig. 2). The hidden layer RBF of RBFNN mostly utilizes one of the following nonlinear functions [16]: Gaussian function, poly-quadratic function, inverse poly-quadratic function, and thin plate spline function.

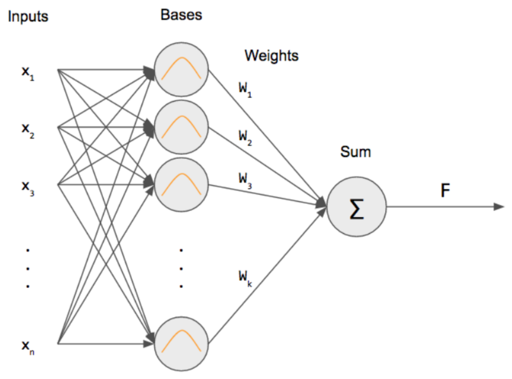


Fig. 2. The Framework of a Radial basis Function Neural Network (RBFNN)

Among these functions, Gaussian functions are the more popular ones. The RBF form used in the RBFNN is not essential to network performance while choosing c_i and σ_i (i.e., mean and standard deviation in the Gaussian function) is key to the whole network 's performance [16]. Clustering algorithms like K-means clustering and self-organizing maps can be used to cluster the data, and then use the clustering/SOM outputs for the Gaussian function in RBFNN. When using K-means clustering, the cluster centers c_i determine the mean values in the Gaussian function. The standard deviation of data in each cluster can be considered as σ_i in the Gaussian radial basis function. When self-organizing maps are used, the number of SOM neurons is the number of the clusters, and the final weights of the SOM networks can be used as the c_i for the Gaussian radial basis functions. Then, for standard deviations, for the cluster center i, we can compute the distance of the cluster i to the other cluster centers and then use $\frac{1}{2}$x minimum distance as the corresponding standard deviation. This lets RBFNN knows where to place the Gaussian functions.

IV. Proposed Method

We discussed the advantages of DBGSOM over SOM and explained the RBFNN method. As mentioned earlier, RBFNNs are fast and comprehensive in training and yield high precision. As Mohammadi and Amiri showed in [7], a combination of SOM with RBFNN is useful in detecting intrusion in a network. Therefore, in this paper, we use DBGSOM in conjunction with RBFNN for anomaly-based intrusion detection. The diagram for our hybrid model process is shown in Fig. 3.

V. Dataset

In the following, we explain the datasets which were used in our experiments. The three datasets are NSL-KDD, UNSW-NB15, and CICIDS2017 datasets, publicly available, and can be downloaded from their designated websites.

A. NSL-KDD

NSL-KDD is an intrusion detection benchmark data set suggested for resolving several inherent issues in the KDD'99

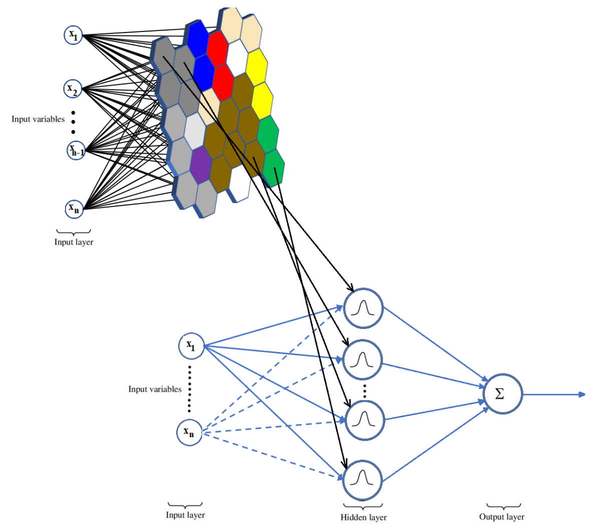


Fig. 3. The Proposed Model Process. Data are First Mapped to K Neurons/Clusters using SOM, and then Obtained Weights for each Neuron is used as the Mean Value to Calculate the Gaussian Function in RBFNN. The Standard Deviation for each Gaussian Function is also Considered as $\frac{1}{2}$x Minimum Distance of the Distances from the Corresponding Neurons in SOM to other Neurons. The Number of Neurons/Clusters in SOM/DBGSOM is equal to the Number of Radial basis Functions in RBFNN.

data set. The number of records in the NSL-KDD train and test sets is sensible. This benefit makes it possible to run experiments on the full set without randomly selecting a small portion. Consequently, the assessment findings will be consistent and comparable to various research work. Compared with KDD, the NSl-KDD dataset does not contain duplicated records in the train set, so classifiers are not biased to more frequent records. Duplicate records were also removed in NSL-KDD, and therefore, the learners' performance is not biased towards the methods with better detection rates on the repeated records. In our experiment, we use the "KDDTrain+_20Percent" version of this dataset for training, and the "KDDTest+" data are utilized for testing. The set of training includes 25192x41 data, and the testing set contains 22544x41 data.

B. UNSW-NB15

The UNSW-NB 15 dataset's raw network packets were generated by the IXIA PerfectStorm tool at the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) to create a hybrid of real, modern normal activities and contemporary synthetic attack behaviors [18]. There are nine types of attacks in this dataset: name, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The Argus, Bro-IDS tools are utilized, and twelve algorithms are developed to generate a total of 49 features with the class label. A section of this data set was created as a training group (i.e., UNSW_NB15_training-set.csv) and testing set (i.e., UNSW_NB15_testing-set.csv). The training set contains 82333x43 data, and the testing set contains 175342x43 data. This partition is also used in our experiments.

C. CICIDS2017

The dataset CICIDS2017 includes benign and most up-to-date common attacks, which resemble real-world results (PCAPs). The top priority in building this dataset has been generating realistic background traffic. Data is captured in five days from Monday to Friday. The dataset for each day based on the day of the week, type of activity, and size of data is summarized as follows:

- Monday, Normal Activity, 11.0G
- Tuesday, attacks + Normal Activity, 11G
- Wednesday, attacks + Normal Activity, 13G
- Thursday, attacks + Normal Activity, 7.8G
- Friday, attacks + Normal Activity, 8.3G

Because of the hardware limitation, we only used the "Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.xlsx" from the MachineLearningCSV version. This data only contains DDoS attack and normal attack. It has 225745 activities, and each activity has 78 features (i.e. 225745x 78). Among these data, 97718 data is normal traffic, and the rest (i.e. 128027) is attack data. We randomly chose .2 data from normal data (i.e. 19544x78) and the same number of data from attack data for training, and we use the rest of the data for testing. Therefore, we have 39088x78 data for training and 186657x78 data for testing.

VI. Experiment Settings and Result

A. Data Pre-processing

Intrusion datasets contain different types of values. The information regarding each scenario (normal or attack) can be both numerical values and categorical values. For categorical values, datasets use text values. For example, each record in NSL-KDD dataset looks like the following data:
"0 tcp ftp_data SF 491 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 0 1 0 0 150 25 0.17 0.03 0.17 0 0 0 0.05 0 normal".

In this dataset, four columns which are related to tcp, ftp_data, SF, and $normal$ contain categorical values. In contrast, the other columns contain numerical values (we should note the last column lists the label for each record by indicating whether it is a $normal$ condition or an $attack$ one). To use the categorical features of these columns in our machine learning, we set numerical value for each category. For example, the second column $Protocol\_type$ contains three categories including tcp, $icmp$, and udp which are respectively set to 1, 2, and 3. We normalize the numerical values in each column by subtracting the mean value of that column and dividing it by the respective standard deviation. However, we notice in all datasets that $most$ of the values in each column are similar to each other, and some values might vary much compared to the rest. Based on this observation, we decide to use each column's median value, which relates to our dataset more closely instead of the mean value.

B. Experimental Results

In this section, we evaluate our anomaly-based intrusion detection method using the three publicly available datasets:

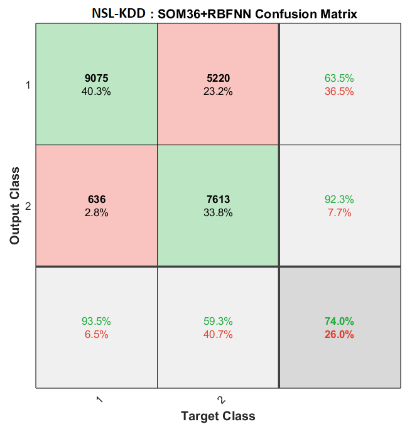
NSL-KDD, UNSW_NB15, and a subset of CICIDS2017 datasets. We have conducted our experiments using a Windows laptop with Core i7 CPU 2.70GHz and 16 GB RAM. We applied the proposed model, and (SOM36&RBFNN) **SOM was implemented with 36 pre-defined clusters followed by RBFNN**. The methods are implemented in MATLAB. We used the built-in MATLAB function for SOM (i.e., the $selforgmap$ function to create the map and the train function to train the SOM map), DBGSOM MATLAB codes are provided by the authors [10], and we implemented the RBFNN method. Table I, Table II, and Table III summarize results for the three datasets. Our hybrid model outperforms (SOM36&RBFNN) in all three cases. The row related to $DBGSOM\_Number\_Of\_Clusters$ in each table indicates the number of clusters (i.e. the final number of neurons) in each run of the proposed model. Since some neurons do not contain any data or they might have very few data, we remove any cluster which contains less than 0.001 of data. While this only removes unnecessary neurons, it speeds up RBFNN as it significantly reduces the number of radial basis functions. In each table, the $DBGSOM\_Final\_Number\_Of\_Clusters$ row indicates the final number of DBGSOM clusters after removing these neurons. Clearly, for all the three datasets, our hybrid model surpasses its (SOM&RBFNN) peer. Also, for each dataset, we included a confusion matrix of one run of the method in Fig. 4, 5 and 6. Confusion matrices for the three datasets also verify that the proposed model outperforms the (SOM&RBFNN) model.

C. Network Traffic Data Visualization

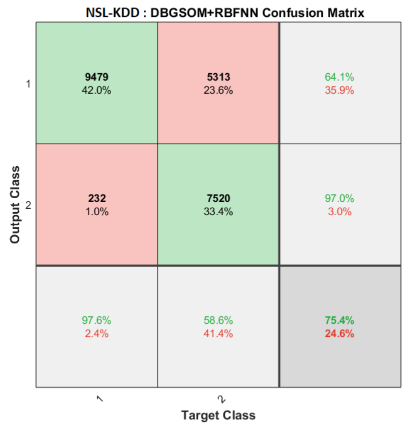
As it was mentioned before, one advantage of SOMs and consequently DBGSOMs is that they can visualize high dimensional data in two dimensions. In Fig. 7, 8 and 9, we illustrate the 2D visualization obtained by SOM and DBGSOM for the three examined datasets. These figures, which are the hit maps (where each data hit on the SOM map), are useful in showing how intrusion or traffic data are different from while connected to each others.

D. Computational Runtime

To compare the CPU time consumed by the DBGSOM+RBFNN method with the SOM+RBFNN, we need to compare the CPU time by SOM with DBGSOM the RBFNN method would have the same calculations when SOM employed with the same number of neurons that DBGSOM ends with. For the sake of this experiment, we fixed the number of epochs for SOM, DBGSOM, and RBFNN to 100. Table IV summarizes the average runtime for applying DBGSOM and SOM on the three datasets. The first column indicates the name of the dataset, the second column shows the number of neurons found by DBGSOM, which is also used by SOM, and the third and fourth columns list the CPU time by DBGSOM and SOM in seconds. From the table, we can see that SOM is faster than DBGSOM because DBGSOM has spent more CPU time than conventional SOM due to the step of inserting neurons, which involves the correct growth position and the weight initialization but has the benefit of conserving data topology on the map [10]. We also included the average runtime of implementing RBFNN as the next step after employing either SOM or DBGSOM in the fifth column. Since applying both



(a) SOM36+RBFNN



(b) DBGSOM+RBFNN

Fig. 4. Confusion Matrix for the NSL-KDD Dataset obtained by running the (SOM&RBFNN) and the Proposed Hybrid Models

our hybrid and the (SOM&RBFNN) models to train a model is offline, we may be interested in knowing the CPU time for testing the model on test data. Therefore, we calculate the CPU time that our system needed to evaluate all the test data, classify them as either normal or attack, and report the result in the sixth column. The next column indicates the average runtime our system needed to evaluate each data. The CPU is taken by all test images divided by the number of test images. The CPU time for testing one record of data for all three datasets is very small. This indicates the proposed method can be employed in a real-time application and would even be speeded up by using high-config systems. The last two columns of the table show the dimension of our training data and testing data. The number of neurons found by DBGSOM indicates why the CPU time for testing one record of data in each dataset is different.

It is significant to highlight the fact that SOM's batch learning principles can save training time dramatically. Besides, the procedure of growing and the ultimate grid structure are independent from the arrangement of the input vector presentation.

TABLE I. Classification rate for the compared methods on the NSL_KDD dataset in 5 runs. Data are normalized; SF value for DBGSOM is .01. After DBGSOM computes centers, classes with less than .001 of data are removed.

Methods	Run#1	Run#2	Run#3	Run#4	Run#5	Average
SOM36+RBFNN	74.31	74.94	74.96	77.72	75.74	75.53
Proposed hybrid model	75.27	76.60	75.71	75.60	76.66	75.97
DBGSOM_Number_Of_Clusters	263	294	299	269	259	276.8
DBGSOM_Final_Number_Of_Clusters	166	191	181	175	165	175.6

TABLE II. Classification rate for the compared methods on UNSW_NB15 data set in 5 runs. Data are normalized, SF value for DBGSOM is .01. After DBGSOM computes centers, classes with less than .001 of data are removed.

Methods	Run#1	Run#2	Run#3	Run#4	Run#5	Average
SOM36+RBFNN	75.69	74.71	82.38	81.47	76.63	78.18
Proposed hybrid model	91.57	91.47	85.72	90.50	85.85	89.02
DBGSOM_Number_Of_Clusters	652	698	658	626	638	654.40
DBGSOM_Final_Number_Of_Clusters	339	373	354	353	366	357.00

TABLE III. Classification rate for the compared methods on the subset of the CICIDS 2017 dataset (i.e., Friday-WorkingHours-Afternoon-DDos.pcap_ISCX) in 5 runs. Data are normalized; SF value for DBGSOM is .01. After DBGSOM computes centers, classes with less than .001 of data are removed.

Methods	Run#1	Run#2	Run#3	Run#4	Run#5	Average
SOM36+RBFNN	98.86	98.82	97.73	98.45	88.44	96.46
Proposed hybrid model	99.27	99.17	99.70	99.22	97.39	98.95
DBGSOM_Number_Of_Clusters	277	245	277	302	293	279
DBGSOM_Final_Number_Of_Clusters	153	150	152	170	165	159

TABLE IV. The average runtime for applying DBGSOM and SOM on the three datasets.

Dataset	Neurons	DBGSOM	SOM	RBFNN	Testing on all test image set	Testing per test image	Training Dimension	Testing Dimension
NSL-KDD	304	252.5041	173.3397	432.4043	3.992026	0.0191805	25192x41	22544x41
UNSW-NB15	833	2260.498	1635.984	3942.145	86.45262	0.0224827	82332x42	175341x42
CICIDS2017	324	761.0038	466.5995	1338.836	67.031675	0.0071727	39088x78	186657x78

With this technique, in each training cycle, the training vectors are offered to the network once, and neurons cumulative fault calculated immediately after the step of weight adaptation. Thus, the neurons network has the opportunity for growth from more than one boundary node which causes difficulty in managing the growth process that leads to dead neurons, which are that don't represent any input vector at the end of the training, and unexpected growth of the network and therefore raises the computational cost. Our hybrid model presents many rules for network growth and that fills just one position around each boundary neuron. Because of the narrow neighborhood function in GSOM, allocation weight vectors to recent neurons have a dramatic influence on map tangling and twisting and must be regarded for serving the quality and smoothness of the network. In DBGSOM, the cumulative error for detecting eligible boundary neurons is not only considered but also for assigning appropriate weight vectors and network location to new neurons.

VII. Conclusion

We propose to apply DBGSOM together with RBFNN for detecting anomaly-based intrusion in the network. DBGSOM employs a batch learning strategy for GSOMs to resolve the issues of SOM and GSOM models. It has a better ability to conservation topology. It reduces exposure for twisting and tangling while offers suitable mechanisms to discover a proper growing position and designating initial weight vectors for the new neurons. RBFNNs, on the other hand, are fast and comprehensive in training, and yields high precision in detecting intrusion. We implemented the DGBSOM+RBFNN method in MATLAB and applied it on three publicly available datasets: NSL-KDD, UNSW-NB15, and CICIDS2017. Our extensive experiment indicates that the proposed method outperforms the SOM+RBFNN method for anomaly-based intrusion detection.

Future works intend to integrate the proposed model with a novel regularization technique that utilizes the standard deviation for classifying and detecting intrusions efficiency. We will incorporate the regularization technique to discourage learning from complex model. As a result, we expect more generalization from the machine learning model to accurately perform on unseen data.

Data Availability: Data used to support the findings of this study are included within the article (see [17] and [18]).

Fund Statement:This research received no external funding.

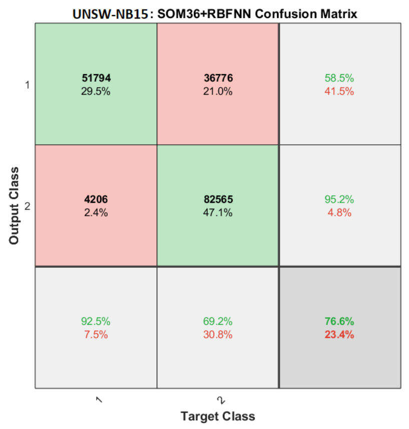
Conflicts of interest:The authors declare that there is no conflict of interest.

References

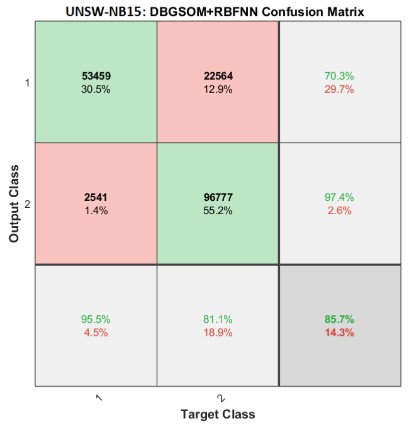
[1] S. A. Althubiti, E. M. Jones, and K. Roy, "LSTM for Anomaly-Based Network Intrusion Detection," in 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), 2018.

[2] S. AlHamouz and A. Abu-Shareha, "Hybrid Classification Approach Using Self-Organizing Map and Back Propagation Artificial Neural Networks for Intrusion Detection," in 2017 10th International Conference on Developments in eSystems Engineering (DeSE), 2017.

[3] https://www.dnsstuff.com/intrusion-detection-system

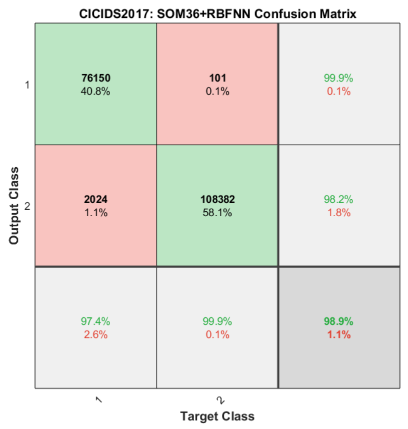


(a) SOM36+RBFNN

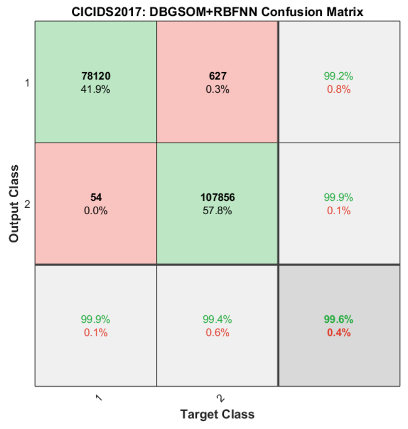


(b) DBGSOM+RBFNN

Fig. 5. Confusion Matrix for the UNSW-NB15 Dataset obtained by running the (SOM36&RBFNN) and the Proposed Hybrid Models.



(a) SOM36+RBFNN



(b) DBGSOM+RBFNN

Fig. 6. Confusion Matrix for the CICIDS2017 Dataset obtained by running the SOM36+RBFNN and Proposed Hybrid Models.

[4] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks,” in 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), 2018.

[5] R. C. Staudemeyer, “Applying long short-term memory recurrent neural networks to intrusion detection,” South Afr. Comput. J., vol. 56, no. 1, pp. 136–154, 2015

[6] S. Albayrak, C. Scheel, D. Milosevic, and A. Muller, “Combining Self-Organizing Map Algorithms for Robust and Scalable Intrusion Detection,” in International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06).

[7] S. Mohammadi and F. Amiri, “An Efficient Hybrid Self-Learning Intrusion Detection System Based on Neural Networks,” International Journal of Computational Intelligence and Applications, vol. 18, no. 1, p. 1950001, Mar. 2019.

[8] Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji, and P. Li, “Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach,” in Proceedings of the 10th Asian Conference on Machine Learning (ACML18), Beijing, China, Nov. 14-16, 2018.

[9] J. An and S. Cho. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. Technical Report, SNU Data Mining Center, 2015. http: //dm.snu.ac.kr/static/docs/TR/SNUDM-TR-2015-03.pdf

[10] M. Vasighi and H. Amini, “A directed batch growing approach to enhance the topology preservation of self-organizing map,” Applied Soft Computing, vol. 55, pp. 424–435, Jun. 2017

[11] R. Vinayakumar, K. P. Soman and P. Poornachandran, ”Applying convolutional neural network for network intrusion detection,” 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, 2017, pp. 1222-1228.

[12] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, “Intrusion Detection Using Convolutional Neural Networks for Representation Learning,” in Neural Information Processing, Springer International Publishing, 2017, pp. 858–866.

[13] J. Yang, T. Li, G. Liang, W. He, and Y. Zhao, “A Simple Recurrent Unit Model Based Intrusion Detection System With DCGAN,” IEEE Access, vol. 7, pp. 83286–83296, 2019.

[14] A. AbuGhazleh, M. Almiani, B. Magableh, and A. Razaque, “Intelligent Intrusion Detection Using Radial Basis Function Neural Network,” in 2019 Sixth International Conference on Software Defined Systems (SDS), 2019

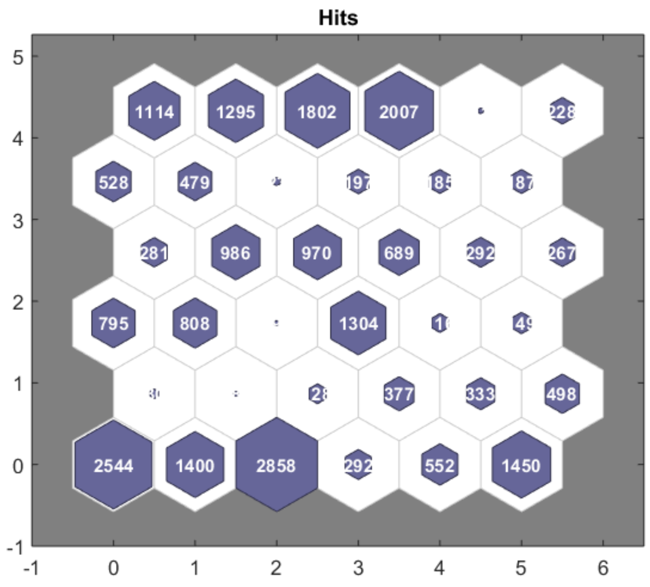
[15] D. Alahakoon ; S.K. Halgamuge ; B. Srinivasan, “Dynamic self-organizing maps with controlled growth for knowledge discovery”, IEEE Transactions on Neural Networks, Volume: 11, Issue: 3, pp.

601-614, 2000

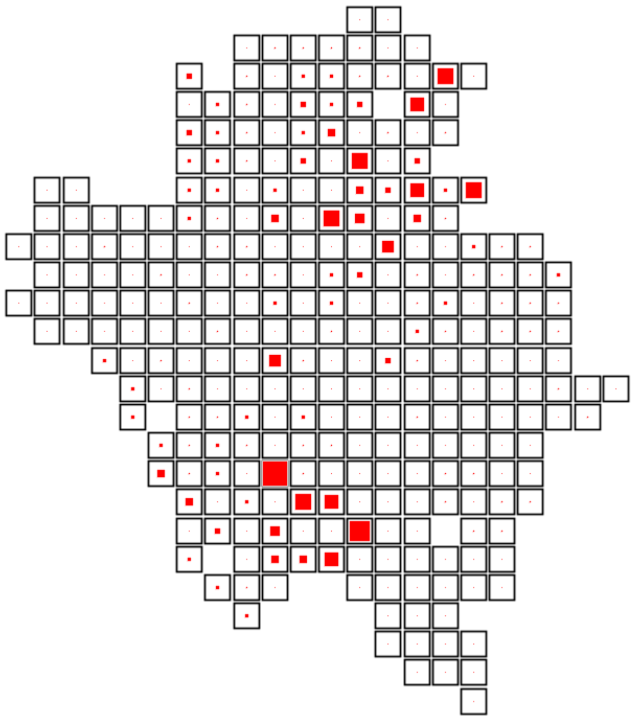
[16] Q. Zhang and F. Wilson, RBNN Application and Simulation in Big Data Set Classification, Journal of Intelligent& Fuzzy Systems (JIFS), pp. 1-9, 2019.

[17] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[18] N. Moustafa, "Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic." Ph.D. dissertation, University of New South Wales, Canberra, Australia, 2017.

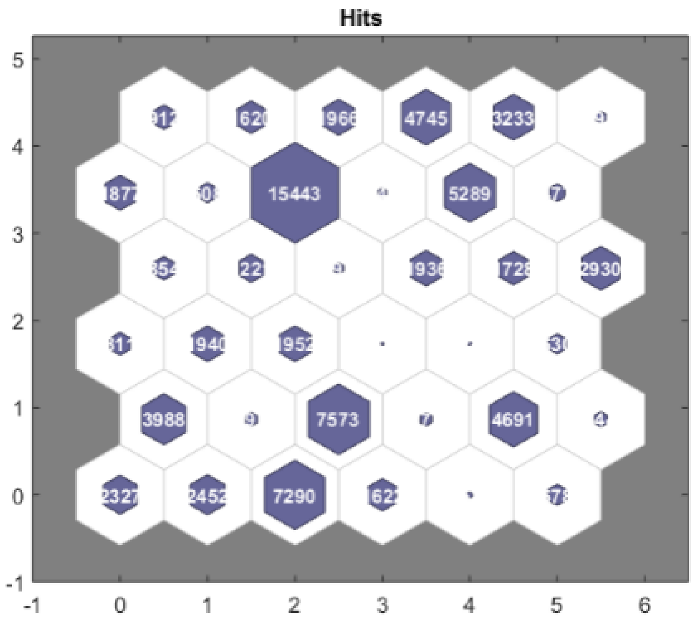


(a) SOM36

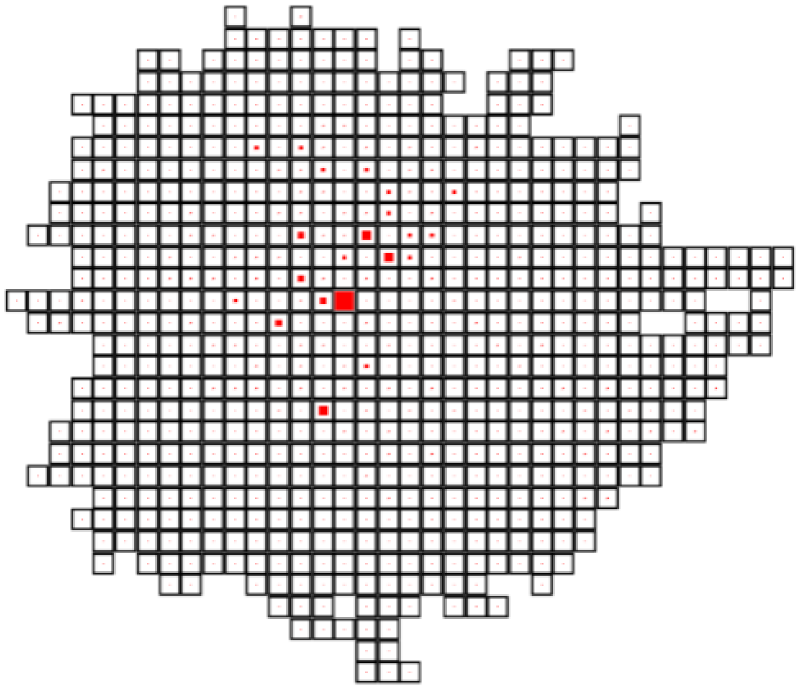


(b) DBGSOM

Fig. 7. Hit Maps from the NSL-KDD Data Set obtained by (a) SOM with 36 Neurons, (b) DBGSOM with 310 Neurons. For SOM, each Cell Number shows the Number of Data assigned to those Cell/Neurons and for DBGSOM, the Number of Data assigned to each Cell/Neuron is Proportional to how much that Cell is Colored.

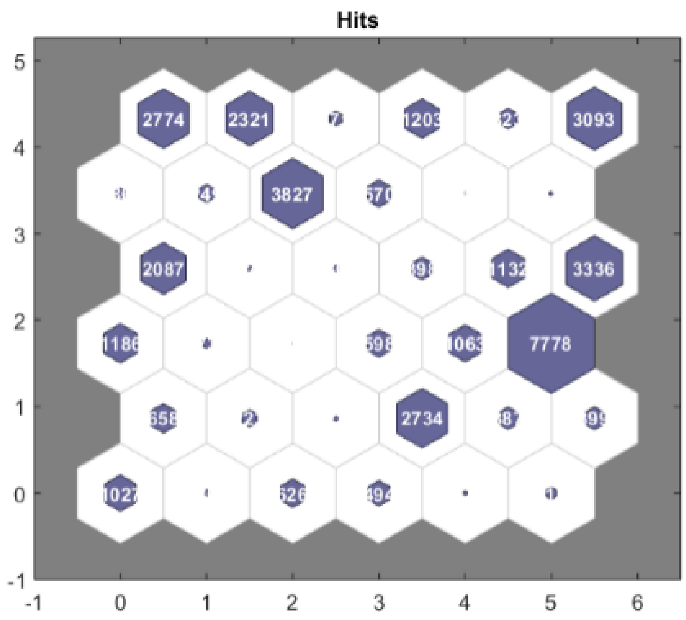


(a) SOM36

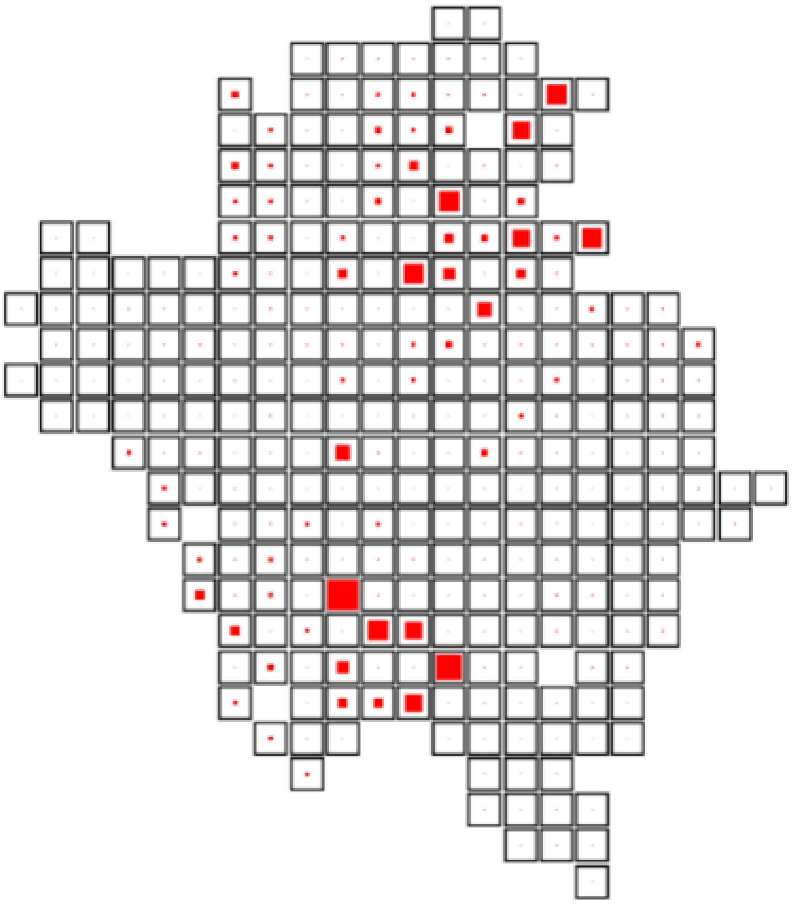


(b) DBGSOM

Fig. 8. Hit Maps from the UNSW-NB15 Data Set obtained by (a) SOM with 36 Neurons, (b) DBGSOM with 310 Neurons. For SOM, each Cell Number shows the Number of Data assigned to those Cell/Neurons and for DBGSOM, the Number of Data assigned to each Cell/Neuron is Proportional to how much that Cell is Colored.



(a) SOM36



(b) DBGSOM

Fig. 9. Hit Maps from the CICIDS2017 Data Set obtained by (a) SOM with 36 Neurons, (b) DBGSOM with 310 Neurons. For SOM, each Cell Number shows the Number of Data assigned to those Cell/Neurons and for DBGSOM, the Number of Data assigned to each Cell/Neuron is Proportional to how much that Cell is Colored.