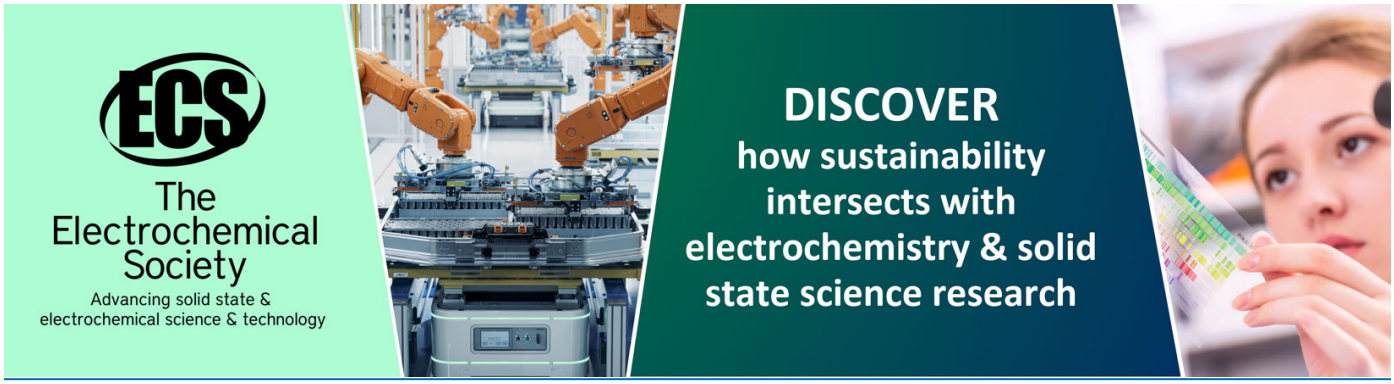
New Improved Training for Deep Neural Networks Based on Intrusion Detection System



IOP Conf. Series: Materials Science and Engineering **435** (2018) 012034 doi:10.1088/1757-899X/435/1/012034

New Improved Training for Deep Neural Networks Based on Intrusion Detection System

Ilyas Benmessahel[1, a,*], Kun Xie[1,b] and Mouna Chellal[2, c]

[1] College of Computer Science and Electronics Engineering, Hunan University, Changsha, China.
[2] School of Information Science and Engineering, Central South University, Changsha, China.

[a] * ilyasbenms@hnu.edu.cn; [b] kunxie@hun.edu.cn; [c] mounachellal@gmail.com

Abstract. All Network intrusion detection is designed for detecting, preventing, and repelling network security breaches and it has become an urgent issue. Maintaining a safe and secure network requires an efficient and flexible solution called an intrusion detection system. This paper reports an advanced intrusion detection method created with a deep learning approach. Evolutionary operators can reduce the probability of stagnation in local solutions due to high local optima avoidance and have thus superseded conventional training algorithms, such as back propagation (BP). Combining a deep neural network (DNN) and an evolutionary algorithm (EA) may solve problems or outperform DNN in solving existing problems. We develop a hybrid training method that combines simulated annealing (SA) and BP to improve the performance of DNN (SABP-DNN). The NSL-KDD dataset is used to verify the accuracy and efficiency of the proposed method. The proposed method is also compared with the original DNN based on PB (PB-DNN) and DNN based on SA (SA-DNN). We confirm that the proposed method presents a strong potential to become an alternative solution to IDS through experiments and comparisons with existing methods.

1. Introduction

The increase in the number of local networks has led to continuous evolution of Internet data and the availability of massive amounts of network data has promoted the development of information technology, which requires careful attention. Consequently, this evolution, in turn, has increased the vulnerability of systems to various threats [1]. Any intrusion can have disastrous consequences. For example, personal data can be destroyed, corrupted, or illegally accessed as a result of confidentiality breaches. Furthermore, integrity breaches can lead to the alteration of personal data. Computer network security has become a promising tool to provide secure channels. One of the promising tools that can detect attacks is the intrusion detection system (IDS). Cybersecurity infrastructures employ IDS as an essential component and use it to protect systems and infrastructures from various threats.

IDS is a permanent monitoring scheme applied in computer or network systems to monitor targeted systems, collect audit data, analyze the gathered information to determine unusual activities and establish response plans [2]. IDS should either model any type of attacks or anomalous events that can affect the network under consideration (signature based) or build a general model that describes normal traffic (anomaly based).

Network IDS comprises a set of single-purpose sensors or host computers that are positioned at different points in a network or interconnected set of networks to monitor transmitted traffic. These units examine network traffic packets in real time, perform local analysis for a captured packet, and finally take action, such administrator notification [3]. NIDS involves two major detection techniques, namely, misuse-based detection and anomaly-based detection [4]. Misuse detection systems, such as works in [5, 6], define the behavior of an intruder and scan predetermined attack data to match signatures. Anomaly detection systems, such as works in [5, 7], set only normal (typical) or expected behavior and flag any deviation from this behavior. However, these two detection systems possess weaknesses. Misuse detection systems cannot detect new forms of attack because their signatures are not yet available for pattern matching. Anomaly detection systems produce a high false alarm rate when adopted in most existing approaches because building practical normal behavior to protect targeted systems is difficult [8].

Many techniques have been utilized to enhance the performance of IDS. However, security experts still strive to achieve IDS with improved performance, the highest detection rate and the lowest false alarm rate. Machine learning has been impeccably accomplished in many areas of computer science, such as speech, visual and face detection, but not sufficient in intrusion detection [3].

Recently, developed techniques have elicited considerable attention in the search for a solution to accommodate concerns regarding intrusion detection. These techniques include artificial neural networks [5], evolutionary algorithms (e.g., genetic algorithm (GA) [9], particle swarm optimization (PSO) [10], and ant colony optimization (ACO) [11]) and deep learning approaches (e.g., deep neural network (DNN), restricted Boltzman machine, deep Boltzman machine, and convolutional neural network (CNN)) [12]. The present study is focused on two of these methods: deep neural network (DNN) and simulated annealing (SA).

DNN and SA have not been sufficiently explored in literature. Furthermore, we selected these methods due to the fact that DNN employs consecutive layers of information processing stages in a hierarchical manner for pattern classification and feature or representation learning [10]. Moreover, weight learning tasks are considered essential in the application of DNN. In other hand, SA has shown good performance in solving continuous optimization problems due to the search mechanisms used in its mutation operators [13]. Thus, a natural question arises: can we benefit from the advantages of SA techniques in overcoming the training phase of DNN by injecting search strategies of SA into the training operator of DNN algorithms?

Although DNN can solve various learning tasks, it is difficult to train [14]. Deep-learning algorithms, which inspired by the neurons that make up the human brain, have gained numerous attention in scientific community due to their successful application in area. Recently proposed optimization techniques for training DL use layer-wise pre-training. DNN training requires powerful optimization techniques. Therefore, several conventional algorithms, such as BP, are used to solve this issue. However, the BP algorithm is sometimes susceptible to convergence into local optima because it is used as a local search method in which the final solution depends strongly on the initial weights [8]. The SA algorithm has been used to solve the global optimization problem. The primary advantage of SA over other methods is its capability to avoid local minima [15]. The algorithm functions by selecting the neighbourhood solution within a search space, accepting the worse solution, and moving to inferior solutions to escape entrapment (i.e., local optimal solution). Thus, the efficiency and robustness of the SA algorithm have been verified through the solution of numerical optimization samples.

We developed a new hybrid algorithm called SABP-DNN that combines SA and BP to build an advanced training algorithm for DNN. The proposed algorithm was applied to detect intrusion in the NSL-KDD dataset. This work exploited the local search capability of BP and the global search capability of SA. We injected SA into DNN to facilitate training and improve the detection rate. Consequently, combining SA with BP can reinforce the strengths of the two methods because of their computational advantages and can thus reduce the computational time of the training phase. The well-

known intrusion detection dataset (NSL-KDD) was used to test the performance of the proposed method. The main contributions of this study can be summarized as follows:

- An intrusion detection model based on DNN-trained SA combined with BP was presented and proven feasible its usefulness, as evidenced by the increased detection rate for NSL-KDD data.
- The principles of SA were examined to optimize the DNN weight and determine its effectiveness in the IDS field.
- The efficiency of DNN was evaluated by using the NSL-KDD dataset. The SABP-DNN model outperformed SA-DNN and BP-DNN in the detection task.
- The proposed SABP-DNN was compared with other techniques by using the NSL-KDD dataset to confirm the applicability of the proposed approach.

The rest of this paper is structured as follows: Sections 2 provide a brief introduction to the concepts of ANN and outline the mathematical overview of the DNN and SA, respectively. Section 3 and Section 4 discusses the use of SA combined with BP in DNN training. Section 5 discusses an overview of the architecture of the proposed IDS. Section 6 presents the experimental setup and results. Section 7 summarizes the conclusions and directions for future work.

2. Background

2.1. Deep neural network

The term DNN can have several meanings, deep is usually related to the hidden layers contained in a neural network and formally defined as a learning model with many layers of nonlinear transformations. Deep learning is one of the newest techniques of machine learning, it allows computers to acquire human brain abilities in natural functionality such as learning using examples. A DNN is comprised of a collection of advanced deep connected neurons listed in a distinctly layered topology, namely, an input layer, one or multiple hidden layers, and a single output layer [16]. In DNN, which is one of the mainstream deep learning methods, data moves forward these artificial deep neurons from the input units, through the hidden units which processes an aspect of the data, and eventually reaching the output units. The sample DNN includes of three hidden layers, one input layer, and one output layer for a global of five layers. DNNs are typically trained by updating and adjusting neurons weights and biases through the use of the supervised learning BP algorithm [14]. Each node in layer performs a specific calculation, which is defined as:

$$v^{\{l\}} = f\left(z^{\{l\}}\right) = f\left(\sum_{\{j=1\}}^{K} \left\{w_{\{ij\}}^{\{l\}} * I_i^{\{l\}} + \beta_i^{\{l\}}\right\}\right) \quad (1)$$

Where $z^l \in R^{N_l}$ is the stimulation vector, v^l is is the activation vector, $w_{ij} \in R^{N_l * N_l}$ the connection weight matrix, and $\beta_i \in R^N$ is the threshold (or bias) vector. $N \in R^{N_l * N_l}$ is the number neurons at layer l. $f() : R^{N_l} \rightarrow R^{N_l}$ is the activation function applied to the stimulation vector element wise. In most applications, the sigmoid function and the hyperbolic tangent function is the most used as activation function for ANN. Rectified Linear Unit function is popular activation functions for DNN.

2.2. Simulated Annealing Algorithm

SA is a generic probabilistic meta-heuristic for complex system optimization. SA was first proposed by Kirkpatrick et al. [17] in 1983. The SA algorithm simulates the annealing process in metallurgy. Annealing is a technique that involves heating a solid to the highest level and controlling cooling until the energy state of internal atoms is reduced to a minimum and crystal growth has initiated.

Here, we present a general outline of the algorithm. First, a random solution is generated and then the cost is calculated using one of the defined functions. A random neighbor solution is generated and the cost of the neighbor solution is calculated. The following comparisons are established: If the cost of the neighbor solution is ‘smaller’ than the cost of the current solution, then the new solution is accepted. If the cost of the neighbor solution is ‘larger’ than the cost of the current solution, then

'maybe' the new solution is accepted. Nevertheless, it may be accepted after checking the Boltzmann probability factor:

$$P(x) = e^{-\delta f(x)/k*T} \quad (2)$$

where f(x) is the energy function, k is Boltzmann's constant, and T is temperature [13]. The steps are repeated until the algorithm reaches an acceptable solution or the maximum allowable iteration.

3. Design of proposed training method

The training process aims to generate a convenient network approach for the corresponding incoming input, and this procedure is achieved through an iterative learning process. The learning process can be considered as a modification of the randomly generated neural network weights based on the neural network response to a set of training input patterns. Therefore, DNN training can be interpreted as constructing a predictive model (function) and optimizing it in an n-dimensional space. This optimization problem can be solved by SA, which is a stochastic global optimization method suitable for nonlinear function optimization.

The design of the proposed method uses DNN: -input layer one hidden layer with 10 neurons -one hidden layer with 20 neurons -one hidden layer with 10 neurons -output layer. The most important step in DNN training is the use of SA, which is a problem caused by the optimal combination of connection weight values between DNN and SA. The problem of training DNN should be approached suitably for the SA algorithm.

The main goals of using SA to train DNN are to optimize weight complexity, minimize approximation error, and achieve the required accuracy. These purposes can be accomplished through a cost-based process by computing the cost value of each vector solution. The optimal solutions with minimal cost value are selected. The solution vector is represented using values ranging from -1 to 1, and the number of features in each solution vector is computed as follows:

$$Indv_{nbr} = (n_i * n_h) + (2 * n_h) + 1 \quad (3)$$

where nh, ni are the number of hidden neurons and input features, respectively.

In this study, the cost function refers to the representation and measures of the represented solution that used in the optimization algorithm. Cost function evaluates the efficiency of a single solution in a population or the chance of finding a solution and reaching high coverage. The optimization process terminates when the termination criterion is satisfied, which is the allowable number of iterations or a satisfactory cost level, has been achieved. One of the cost functions is adopted in this study is as follows:

$$MSE = \frac{1}{R} * \sum_{i=1}^{R} o - y \quad (4)$$

where o is the desired output, y is the actual output, and R is the number of training samples. Optimization can be terminated on the basis of two situations: First, the maximum allowable iteration is reached. Second, the cost function threshold is attained. Consequently, the optimal state of network has been achieved.

4. Improved training DNN with SA and BP

The improved training algorithm combines SA with BP; therefore the fundamental idea behind this hybrid algorithm is to exploit the global search capability of SA and the local search capability of BP. SA has a strong exploration capability for finding a global optimal solution, which means that SA has stable capabilities in global convergence but arrives at the global optimum. Thus, its search process slows down. By contrast, the BP algorithm can determine whether the global optimal solution is weak or strong in finding a locally optimal solution. Thus, the BP algorithm is capable of attaining rapid convergence around the global optimum for finding the local optimum.

The main principle behind combining SA with BP is as follows: SA is used to explore the global optimum during the initial stage of optimum searching, and BP is used to search around the global optimum to find the best local optimum.

A collection of random solutions is generated to initiate searching process and training. SA compares each network fitness value during each pass through a dataset. The solution with the optimal cost value is considered as the global best optimum. The BP algorithm then exploits the global best optimum to search for the local best optimum. The general training pseudo-code of SABP-DNN is presented in Algorithm 1.

Algorithm 1 Combining SA with BP training DNN pseudo code

1: Initialize training parameters: lb, ub, K, n.
2: Creating a set of random solutions X = x1,x2,,,xn in range of [ub,lb].
3: for each solution do do
4: Calculate the MSE f(X) for solutions via equation 4.
5: if f(x) < fbest then
6: fbest = f(x)
7: end if
8: end for
9: Until (minimum T or K=0)
10: for each solution do
11: Use the BP algorithm to calculate a new solution x0.
12: Calculate the MSE of f(x),f(x0).
13: if f(x0) < f(x) then
14: fbest = f(x0),x = x0
15: else
16: Calculate the difference $\Delta f = f(x0) - f(x)$
17: Random value r(0 , 1)
18: if $r > e^{-\Delta f/kT}$ then
19: fbest = f(x0),x = x0
20: else
21: fbest = f(x)
22: end if
23: f = fbest
24: Reduce the temperature gradually $T = \alpha * T$
25: end if
26: end for
27: Save the best solution with fbest
28: Go to step 9 if the end criterion is not satisfied.

5. OVERVIEW OF PROPOSED MODEL

As shown in Figure 1, the architecture of our proposed IDS mainly consists of three components: The data input module and two basic modules, which comprise the DNN network and SA module.

Figure 1 shows that the data are not directly fed to the DNN module. The data must be pre-processed by normalizing, filtering, and extracting features from the NSL-KDD dataset [18]. The training and testing dataset will be provided as the input to the next phase for various processes by the DNN module.

In this study, the framework of the proposed system is based on the two stages of network training and detection. The DNN training stage uses the SA algorithm such that network weights are treated as solutions that have evolved in the SA algorithm. Once the evolutionary process of finding a suitable solution for DNN is completed, a selected solution representing a set of weights, which are regarded as the optimal weights, is achieved. The evolution process stops when termination is satisfied.

Finally, in the detection stage, the testing data are loaded into predicted models to predict whether the output exhibits normal or abnormal behavior. The detection stage uses the model established during the training stage to map the predicted output with the matched classes.

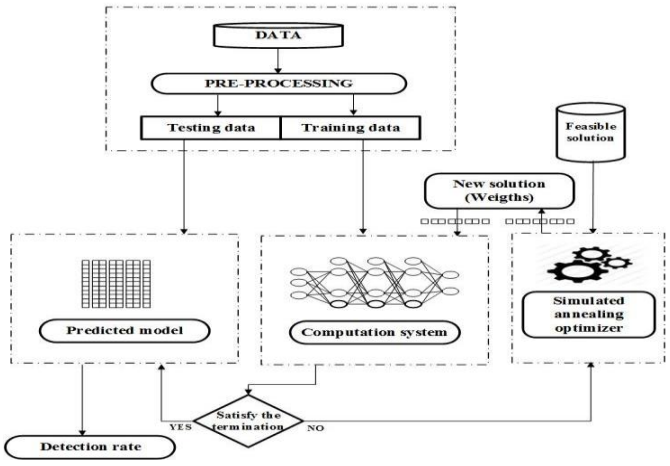


Figure 1. Framework of the proposed IDS.

6. EXPERIMENTAL SETUP AND RESULTS

6.1. NSL-KDD DATASET

NSL-KDD was developed in a suitable form by decreasing redundant instances that influence the evaluation of IDS results [19]. NSL-KDD, which is constructed from records selected from the whole KDDCup99 dataset, addresses many issues related to the KDDCup99 dataset by eliminating all duplicate and redundant instances in the whole dataset.

The NSL-KDD dataset includes the same features as the KDDCup99. Each NSL-KDD instance has 41 features that are labelled as either normal or an attack, with one specific attack type [1]. The whole set is grouped into training and test sets whose attack instances are classified into four major classes, i.e., DoS, U2R, R2L, and Probe, in accordance with purpose. Table 1 shows the distribution of the records in the NSL-KDD dataset for the training and testing sets.

Table 1. Distribution of records in the NSL-KDD dataset.

	Normal	Attack	Total
Training set	13449	11743	25192
Testing set	9714	12830	22544

6.2. Evaluation Metrics

This proposed approach was implemented and evaluated in Visual Basic 2010 on a personal PC with Core I5 2.4 GHz CPU and 4 GB RAM. This study does not focus on setting parameter values. Nevertheless, the selection of these parameters remains an open question. The performance of the proposed models is highly dependent on the proper setting of DNN and SA parameters. All feature records in the experiment were represented at the same predefined boundary [0, 1]. Min-max normalization is the most popular method to fit data in the same boundaries. The performance of the proposed model is assessed via four main parameters: recall (R), precision (P), accuracy (ACC), and F-measure (F). The result of the proposed model requires a high R, high ACC, and low false-alarm rate [2].

6.3. Experimental Results

The performance of the proposed DNN-based IDS was evaluated through a series of experiments. Two experiments were designed to study the performance of the DNN-IDS model for binary classification (normal, anomalous) and compare it with other existing approaches using deep learning.

Minimum and maximum MSE and accuracy were evaluated from all the simulations. Initially, the performance of the model depended on the value of the hyper-parameter initiated.

The first phase of our experiment was performed with the NSL-KDD dataset. Here, 20% of the whole NSL-KDD training dataset was used because these datasets have a large number of records that may require a long computation time with ordinary CPUs and machines or may exceed memory requirements. The training stage was completed with the selected training dataset. On the other hand, the size of the randomly selected testing set is only 80% of the whole NSL-KDD testing dataset.

Firstly, the Learners (BP) were tuned by selecting the hyper-parameters necessary to improve performance. Moreover, we attempted to tune BP-DNN by finding the optimal learning rate within the range of 0.6 to 0.02. We minimized MSE and maximized accuracy when training a model. Table 2 shows MSE of BP-DNN methods during training with NSL-KDD training dataset and the result of the testing with NSL-KDD testing dataset where MSE and training time are taken as performance evaluation factor. Table 3 shows performance of BP-DNN in terms of ACC, RECALL, PRECISION, FPR and F-measure.

Table 2. Results of the experiment based on MSE for BP-DNN.

Learning rate	Training set			Testing set	
	MSE		Time	MSE	Accuracy
	MIN	MAX			
0.6	0,80628	0.96472	25960	0.51880	52.30
0.5	0.16171	0.96472	27278	0.23939	87.51
0.4	0.82659	0.96472	29686	0.50701	47.69
0.3	0.20567	0.96472	27478	0.21357	88.16
0.2	0.11717	0.96472	27329	0.22201	88.94
0.1	0.04025	0.96472	27683	0.12703	92.68
0.08	0.05152	0.96472	26586	0.12372	92.78
0.06	0.03015	0.96472	29153	0.15449	92.00
0.04	0.02786	0.96472	27877	0.16073	91.51
0.02	0.02224	0.96472	28570	0.15430	90.80

Table 3. Experimental results of BP-DNN for NSL-KDD.

Metric	ACC	DR=RECALL	PRECISION	FPR	F-measure
results	92.78	85.78	79.41	0.020	91.32

Secondly, the Learners (SA) were tuned by selecting the hyper-parameters necessary to improve performance. Moreover, we optimized SA-DNN by finding the proper value of Alpha (α) within the range of 0.9 to 0.1. We minimized MSE and maximized accuracy when training a model. Table 4 shows the MSE of SA-DNN methods during training with the NSL-KDD training dataset and the result of the testing with the NSL-KDD testing dataset, where MSE and training time are regarded as performance evaluation factors. Table 5 shows performance of BP-DNN in terms of ACC, RECALL, PRECISION, FPR and F-measure.

Table 2 shows that in the training phase, a low learning rate was followed by a decrease in MSE and an increase in ACC. However, in the testing phase, when we reduced the learning rate to 0.009, the results of each change were matched by a change in MSE and accuracy. Therefore, all of the resulting changes were not as good as the learning rate of 0.08. Table 5 shows that the changing of (α) value leads to the changing of MSE and ACC values. Alpha decreases from 0.7 to 0.3 where MSE reaches its minimal value. Consequently, ACC achieves good result when MSE is minimal.

Table 4. Results of the experiment based on MSE for SA-DNN.

Alpha	Training set			Testing set	
	MSE		Time	MSE	Accuracy
	MIN	MAX			
0,9	0.27312	0.85425	24667	0.27958	84.57
0.8	0.22926	0.85425	27278	0.22541	86.88
0.7	0.23744	0.85425	25229	0.32475	82.05
0.6	0.22063	0.85425	24798	0.28204	70.46
0.5	0.21523	0.85425	25109	0.32514	82.08
0.4	0.19672	0.85425	23413	0.19908	87.96
0.3	0.16918	0.85425	23388	0.18335	93.10
0.2	0.17396	0.85425	24736	0.22996	89.76
0.1	0.27549	0.85425	24429	0.27973	85.05

Table 5. Experimental results of SA-DNN for NSL-KDD.

Metric	ACC	DR=RECALL	PRECISION	FPR	F-measure
results	93.10	88.00	97.25	0.022	92.17

Comparison of the results of the training and testing phases related to the proposed approach showed that SA-DNN achieved acceptable performance in terms of ACC (93.10), RECALL (88.00) and time of training compared with BP-DNN. BP-DNN performed better than SA-DNN in terms of PRECISION (97.41) and FPR (0.020). However, determining which method performs better by using ACC, RECALL, PRECISION, and FPR is impossible because all the results are close. The metric of F-measure provides a better measure of the accuracy of a model compared with all of the metrics because both of precision (P) and recall (R) are considered for calculating F-measure. The goal is to achieve a high F value to provide accurate results, which means all instances are classified correctly. SA-DNN obtained (92.17) better results than BP-DNN (91.32). Therefore, SA provides accurate results. For the overall evaluation, the learning rate of 0.08 and alpha of 0.3 of BP-DNN and SA-DNN, respectively, reveal that these two methods showed the best performance. Thus, they were selected for further evaluation.

Although the results of BP-DNN and SA-DNN were not good enough, we evaluated the result of the SABP-DNN model based on the two to show the performance of the proposed model. The performance of the SABP-DNN algorithm was evaluated using the best parameter found in the two previous experiments. Table 6 shows the MSE of the model with 0.08 learning rate and 0.3 alpha. Table 7 shows performance of SABP-DNN based on ACC, RECALL, PRECISION, FPR and F-measure.

Table 6 shows that the result of SABP-DNN in terms of MSE was 0.03356 in 21842 ms. this result shows an improvement compared with SA-DNN and BPDNN, which obtained 0.05152 in 26586 ms and 0.16918 in 23388 ms, respectively.

Table 6. Results of the experiment based on MSE for SABP-DNN.

Learning rate / alpha	Training set			Testing set	
	MSE		Time	MSE	Accuracy
	MIN	MAX			
0.08 / 0.3	0,03356	0.08539	21840	0.08097	95.56

Table 7. Experimental results of SABP-DNN for NSL-KDD.

Metric	ACC	DR=RECALL	PRECISION	FPR	F-measure
results	95.56	91.88	98.73	0.010	94.17

Table 7 also shows an improvement in the results in terms of ACC, RECALL, PRECISION, FPR, and F-measure, which gained values of 95.56, 91.88, 98.73, 0.010, and 94.17, respectively.

Table 8 shows that our SABP-DNN approach performed better than the existing approaches.

Table 8. Performance of our proposed method SABP-DNN compared to other methods for KDD99 dataset.

Author/reference	Technique	Accuracy (%)	DR (%)	FAR (%)
[7] (2016)	ANN	95.04	-	1.48
	LR	92.75	-	18.84
	NB	95.00	-	5
[11] (2014)	SVM	-	66.70	5.53
	CSOACN	-	80.10	2.84
	CSOAC	-	78.18	2.77
[18] (2012)	SSOWLS	93.30	-	-
	SSO	89.60	-	-
	PSO	88.50	-	-
[4] (2015)	DNN	75.75	76.00	0.86
[17] (2016)	ADBCC	92.71	91.79	3.50
[8] (2017)	GSPSO-ANN	95.26	-	-
	PSO-ANN	92.06	-	-
	GS-ANN	92.81	-	-
In this work(2018)	SABP-DNN	95.65	91.88	0.01

Overall, the SA algorithm is highly recommended for use in intelligent hybrid optimization schemes, such as hybridization with BP, for finding optimal DNN weights. This recommendation is based on the high exploratory behavior of the algorithm, which prevents finding the local optima through training DNNs. The SA algorithm employs a random neighbor search, which allows changes in acceptance probability to minimize loss function and inferior solutions to escape obtaining the local optima. The highly exploitative behavior of the algorithm is another reason for the rapid convergence of an SA-based trainer toward the global optimum for different datasets. Meanwhile, the BP algorithm can determine if the global optimal solution is weak and finds a locally optimal solution. Moreover, the BP algorithm can attain rapid convergence around the global optimum for finding the local optimum. Combining SA with BP can reinforce the strengths of each method because of their computational advantages and can reduce the computational time of DNN training.

7. Conclusion

This study demonstrated that the deep learning method of DNN can be successfully applied in intrusion detection. This deep learning model learns high-dimensional representations and efficiently performs attacks detection. SA can be combined with BP to train and adjust a DNN and to learn similarity representation over nonlinear and high-dimensional input data. Doing so can perfectly facilitate attack detection. In the IDS environment, the deep learning approach can be used to successfully develop an advanced detection model for detecting potential attacks. The main advantage of combining SA with BP is the use of distinct patterns to explicitly avoid being trapped in the local minima. Hence, many of the issues of premature convergence can be prevented, and good data generalization can be achieved.

Experiments on the NSL-KDD dataset showed that our proposed approach, SABP-DNN, effectively trains DNN compared with BP-DNN and SA-DNN training methods. The statistical results showed that SABPDNN is robust because the variance values are small. The experimental results for the NSL-KDD dataset showed that DNN can learn a good generative model and performs well in intrusion detection. Deep learning approaches provide new design ideas and methods for future IDS research. The results of this study prove the potential applicability of DNN as an alternative solution for developing practical IDSs.

References

[1] I Ahmad, A. B Abdullah, and A S Alghamdi. Application of artificial neural network in detection of probing attacks. *IEEE Symposium on Industrial Electronics Applications*, volume **2**, pages 557–562, Oct 2009.

[2] T Ma, Y Yu, F Wang, Q Zhang, and X Chen. A Hybrid Methodologies for Intrusion Detection Based Deep Neural Network with Support Vector Machine and Clustering Technique, pages 123–134, 2018.

[3] E Hodo, X Bellekens, A Hamilton, C Tachtatzis, and R Atkinson. Shallow and deep networks intrusion detection system: A taxonomy and survey. *arXiv preprint arXiv:1701.02145*, 2017.

[4] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho. Deep learning approach for network intrusion detection in software defined networking. In *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pages 258–263, Oct 2016.

[5] H Berlin, L Djionang, and G Tindo. A new networks intrusion detection architecture based on neural networks. *Global Journal of Computer Science: Network, Web and Security*, **17**(1):19–27, 2017.

[6] Q Xu, L Yang, Q Zhao, and Z He. A novel intrusion detection mode based on understandable neural network trees. Journal of Electronics (China), 23(4):574–579, Jul 2006.

[7] N Moustafa and J Slay. The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal: A Global Perspective*, **25**(1-3):18–31, 2016.

[8] T Dash. A study on intrusion detection using neural networks trained with evolutionary algorithms. *Soft Computing*, **21**(10):2687–2700, 2017.

[9] G Ke and Y H Hong. The research of network intrusion detection technology based on genetic algorithm and bp neural network. In *Frontiers of Manufacturing Science and Measuring Technology IV*, volume **599** of *Applied Mechanics and Materials*, pages 726–730. Trans Tech Publications, 10 2014.

[10] C Qiu and J Shan. Research on intrusion detection algorithm based on bp neural network. *International Journal of Security and Its Applications*, **9**(6):247–259, 2015.

[11] W Feng, Q l Zhang, G zh Hu, and J Huang. Mining network data for intrusion detection through combining svms with ant colony networks. *Future Generation Computer Systems*, **37**(Supplement C):127 – 140, 2014.

[12] M E Aminantoa and K Kimb. Deep learning in intrusion detection system: An overview. 2016.

[13] L Rere, M I Fanany, and A M Arymurthy. Simulated annealing algorithm for deep learning. *Procedia Computer Science*, **72**:137–144, 2015.

[14] T Nikoskinen. From neural networks to deep neural networks. 2015.

[15] H Shi and W Li. Evolving artificial neural networks using simulated annealing based hybrid genetic algorithms. *JSW*, **5**(4):353–360, 2010.

[16] A Koutsoukas, J. Monaghan, X l Li, and J Huan. Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of Cheminformatics*, **9**(1):42, Jun 2017.

[17] S Kirkpatrick, C Gelatt, and M Vecchi. Optimization by simulated annealing. *Science*, **220**(4598):671–680, 1983.

[18] Y Y Chung and N Wahid. A hybrid network intrusion detection system using simplified swarm optimization (sso). *Applied Soft Computing*, **12**(9):3014 – 3022, 2012.

[19] M Tavallaee, E Bagheri, W Lu, and A Ghorbani. A detailed analysis of the kdd cup 99 data set, CISDA'09, pages 53 –58, Piscataway, NJ, USA, 2009. IEEE Press.