EasyChair Preprint

№ 3452

An Effective Deep Learning Algorithm for Intrusion Detection

Yi-Lin Chen, Huan Chen and Chun-Wei Tsai

May 21, 2020

An Effective Deep Learning Algorithm for Intrusion Detection

Yi-Lin Chen[1], Huan Chen[1], and Chun-Wei Tsai[2]

[1] Computer Science and Engineering, National Chung Hsing University, Taiwan
a0989735018@gmail.com, huan@nchu.edu.tw
[2] Computer Science and Engineering National Sun Yat-sen University, Taiwan
cwtsai@cse.nsysu.edu.tw

Abstract. Although many mature technologies can be used to prevent cyberattacks, we still have to redesign or adjust these detection or defense systems for unknown cyberattacks today. In fact, the speed of update for these systems may be slower than the production of cyberattacks. To solve this problem, the technology of intrusion detection and prevention system (IDPS) is indispensably required for network because it is capable of not only detecting the unknown attacks but also preventing attacks. More recently, using deep learning technology for the IDPS to precisely detect the attacks in a network is a promising research topic. However, because the parameter setting in deep learning still relies on manual operation by human, setting parameters, such as the number of layers for a neural network and the number of neurons in each layer, to reach a higher accuracy still is a critical issue in this research domain. In this paper, we will present an effective algorithm, which combines a metaheuristic algorithm with deep learning to dynamically adjust the parameters of deep learning (i.e., the number of neurons in each hidden layer) to enhance the performance of deep learning for IDPS. The experimental results show that the accuracy of our proposed method is outperform than current deep learning methods compared in this research for IDPS.

Keywords: Intrusion Detection System · Deep Learning · Metaheuristic.

1 Introduction

The Internet has developed rapidly due to advances in technology it has gradually become an integral part of our daily life. For example, people, families, companies, and even governments exchange information and communicate through the Internet anytime and anywhere. Although the progress of the Internet has brought a lot of convenience, it exposed internet users under the risk of many cyber attacks. In order to defend against all kinds of cyber attacks, there are many current technologies which play important roles in network security developed, such as firewalls, antivirus software, encryption of data, honeypots and intrusion detection systems (IDS). The IDS [6,7] typically can be roughly divided into network-based IDS (NIDS) and host-based IDS (HIDS). NIDS is mainly used to

analyze and monitor whether there exists abnormal attack behavior in packets on the network. HIDS is a system placed on the host, immediately monitoring whether log files are edited or whether there are malicious programs.

Several successful results [5, 17] show deep learning can provide an effective way to solve complex problem in every research field. In particular, the prediction of image and sequence in time is very successful, but it requires a lot of domain knowledge for setting parameters. In the research [2], there are three ways to adjust the parameters at this stage, including manual search, grid search and random search. These three methods need plenty of time to adjust, and it adjusts for Rule of thumb. Typically, a deep learning algorithm has many parameters that can be adjusted. For example like Neurons, Number of layers, weight values of Neural network and learning rates in the optimizer.

Based on the above defect, using metaheuristics algorithm is a feasible way to adjust parameters because metaheuristic algorithm able to find an approximate solution within the reasonable time. That is why it is applied to several NP problems such as travelling salesman problem (TSP). This is also the reason why metaheuristic algorithm is suitable for solving adjustment parameters. In research [13, 14], the metaheuristic algorithm which is combined with machine learning into new architectures helps to adjust the parameters of machine learning. These methods use the characteristics of metaheuristic algorithm to find the approximate optimal solution in the solution space. The great performance improvement of machine learning enables researchers to quickly adjust optimal state of the system. Because of the successful research on optimization of machine learning parameters, the study of [3] mentioned that using metaheuristic algorithm to improve the performance of deep learning or to adjust its parameters is a promising research topic in recent years.

Generally speaking, using metaheuristic to improve performance of deep learning currently can be divided into two categories. The first one is updating weights in deep learning network by using a hybrid method of metaheuristic algorithm and original update method (backpropagation). For example, in the studies [1, 8, 9], the simulated annealing (SA), differential evolution (DE), harmony search (HS) and particle swarm optimization (PSO) are used to find the best weight value and bias. In study of [12], Tian and Fong provide a brief review of hybrid updating methods for deep learning. These hybrid methods aimed at using both characteristics of metaheuristic algorithm and backpropagation when updating weight and bias. As for the feature of metaheuristic algorithm, it narrows down the search space of solution because it performs well in global search of solution. As for the feature of backpropagation, it performs well in searching best result in small region. The advantage of combing metaheuristic algorithm and backpropagation can reduce the number of searches, search time, and computing resources especially. The second one is adjusting parameter for architecture of deep learning. Great parameters can effectively improve the performance of the neural network. Young in the study of [16] utilized evolutionary algorithm to adjust parameters in architecture of convolutional neural networks (CNN). The number of layers is configured and fixed at the first, and then the number of

neurons in each layer will be adjusted dynamically. In this method, the objective value is used to find the loss value then find the best combination solution. In the study of [15], Yin et al. used the recurrent neural network (RNN) model to detect network attacks. The experimental show that deep learning method can provide higher accuracy than other machine learning algorithms. However, the method presented by Yin et al. [15] is by manual adjustment or grid adjustment for the parameters. It lacks autonomy and requires adjustment of the rule of thumb.

For this reason, in this research, we will attempt to develop a more useful way to let the deep learning algorithm can able to adjust its parameters by the metaheuristic algorithm for the detection module of IDPS. It can be expected that the proposed algorithm may able to reduce the error of manual interference and then able to find the best solution. The rest of the paper is organized as follows: The rest of this paper is organized as follows. Section 2 starts with the basic idea of the proposed system and then provides the details of the proposed algorithm. Section 3 first gives the descriptions experiment environments, datasets, and then the comparison results of the proposed algorithms and other classification algorithms compared in this paper. Finally, Section 4 draws the conclusion and gives some future research directions of this research.

2 The Proposed System

2.1 The System Architecture

The method proposed in this paper to divided into three parts, one is data preprocessing, another is metaheuristic algorithm, and the other is deep learning model. The training model is divided into four steps. The first step is to preprocess the data, the second step is using the deep learning model for prediction, the third step is using metaheuristic algorithm to dynamically adjust the number of neurons in hidden layer, and the last step is to save the best model. The detection method is divided into two steps. The first step is to preprocess the data, and the second step is to put the data into intrusion detection system for detection, as shown in Figure 1.

2.2 The Proposed Algorithm

In this research, we first consider the parameter settings of deep learning algorithm as a multi-dimensional optimization problem. After then, we used differential evolution (DE) [11] for solving this optimization problem.

The design of proposed algorithm will based on the study of [4] to solve this multi-dimensional optimization problem. Different to the simple DE, the proposed algorithm contains initialization, mutation, crossover, selection, and update operators, as shown in Algorithm 1. The s represents the initialization of the population solution, and each population solution represents the number of hidden layer neurons. In the initial operator, the number of neurons is randomly

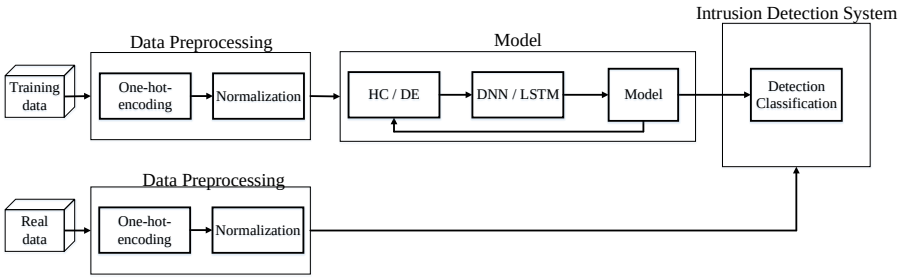


Fig. 1. System Process Chart

Algorithm 1: Differential Evolution Algorithm

```
Differential Evolution
{
    S = Initialization()
    For each iteration do
        M = Mutation(S)
        C = Crossover(S, M)
        S = Selection(S, C)
        S = Update(S)
    End for
    Output S
}
```

generated and the input layer is reduced by one and the number of output layers is increased by one.

As for the mutation operator, M represents the mutated solution; S represents the original solution; $r1$ and $r2$ are randomly selected and unequal populations; F is a crossover probability, as shown in Eq. (1). In this operator, each population solution must be mutated. The mutation operator will subtract the populations of $r1$ and $r2$ and multiplying them by F, and it adds the result of the current solution to become the solution of next iteration.

$$M_{current}(t+1) = (S_{r1}(t) - S_{r2}(t)) \times F + S_{current}(t) \quad . \quad (1)$$

As for the crossover operator, C_r represents the exchange rate; C_{ij} represents the j^{th} dimension in the i^{th} solution after crossover operator, as shown in Eq. (2). In this operator, each solution and dimension will be crossovered. First, a floating point number $r \in [0, 1]$ will be randomly generated, and the determine to use $M_{ij}(t+1)$ or $S_{ij}(t)$.

$$C_{ij}(t+1) = \begin{cases} M_{ij}(t+1) & \text{if } r \leq C_r, \\ S_{ij}(t) & otherwise. \end{cases} \tag{2}$$

The selection operator will use a greedy method. This greedy method is to select the fitness (f) superior individual as a new individual. The detailed method is as shown in Eq. (3).

$$S_i(t+1) = \begin{cases} C_i(t+1) & \text{if } f(C_i(t+1)) \geq f(S_i(t)) \\ S_i(t) & otherwise \end{cases} \tag{3}$$

In order to prevent falling into the local solution, an improved method presented Lee et al. [4] is also used in the proposed algorithm. In the update operator, the proposed algorithm will judge whether the optimal solution for the five consecutive iterations is the same. If they are the same, we will resetting the solution of a certain dimension in a population, and the actual solution should be divided into two steps in this paper. The first step is that if the best solution of the five iterations are the same, the best solution from the first iteration to the latest iteration will be added and divided by the current number of iterations. After the first step, it may not be an integer, but the number of neurons must be an integer, so we force the conversion to an integer. The second step is that a population is randomly produced and replaced with the value generated by the first step, as shown in Figure 2.

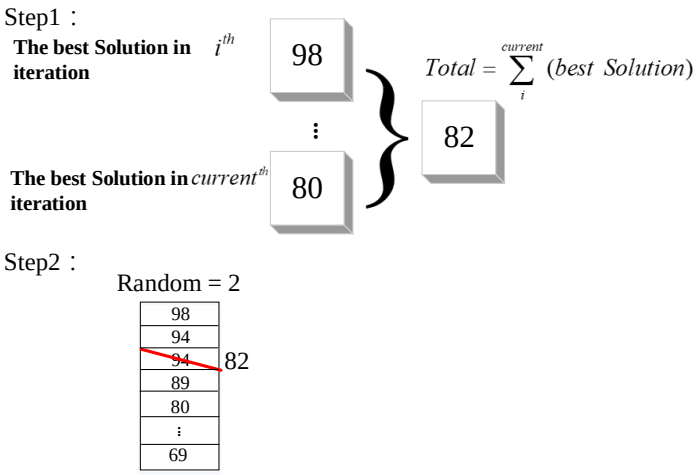


Fig. 2. DE update step

3 EXPERIMENTAL RESULTS

3.1 Experimental Environment

In this chapter, we conducted two sets of experiments, one of which is to divide the data into two categories, and the other one is to divide the various attacks into five categories. The experimental results are processed by one PC with Intel i9-7900XE CPU (2.60 GHz, 13.75 MB cache, and 10 cores), Nvidia Titan XP GPU and 16 GB of memory running in Ubuntu 18.04. The programming language used is python. Because of the large number of data and a large number of operations required for deep learning neural networks, GPU is used to speed up the experiment. To be fair, our deep learning model use the same architecture, so the neural network architecture will be an input layer, a hidden layer, and an output layer. The number of neurons in the input layer equals to the number of features of each data, and the output layer is divided into two output neurons and five output neurons by experiment. The number of neurons in the hidden layer will be dynamically adjusted by metaheuristic to achieve the best accuracy.

Table 1. Small attacks in the four major attacks

Label	Attack
Probe	ipsweep, mscan, nmap, portsweep, saint, satan
DOS	apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm
U2R	buffer-overflow, httptunnel, loadmodule, perl, ps, rootkit, sqlattack, xterm
R2L	ftp-write, guess-passwd, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, worm, xlock, xsnoop

3.2 The Descriptions of Dataset

In current research papers with intrusion detection systems, almost all of the NSL-KDD datasets are used to test the performance of the intrusion detection system. There are two main characteristics in the NSL-KDD dataset. One is to delete the redundant records in the KDD CUP 99 dataset so that the classifier does not favor more frequent records, and the other one is to remove the duplicate records in the KDD CUP 99 dataset to make the detection rate more accurate. The training data and the testing dataset have also been separated in the NSL-KDD dataset, so that different accuracy rates due to the segmentation of the dataset can be avoided. In NSL-KDD, there are two classifications. One is a two-category which divides the data into normal and attack behaviors, and the other one is the attack behaviors in two classifications which are divided into four types. The four large attacks can be subdivided into 39 subcategories. As shown in the Table 1, each type of attack will be recorded in the last column of

each piece of data. Each data in the NSL-KDD dataset represents the data of each network connection, and there are 41 features and 1 label in each data.

There are three features in the NSL-KDD dataset, and the data type in one of the features is string. Therefore, the dataset needs to be preprocessed. The second feature in the dataset represents the communication protocol type, and there are a total of 3 in this feature; the third feature represents the network service type, and there are a total of 70 in the feature; the fourth feature represents the normal or wrong state of the connection, and there are a total of 11 in this feature. So all of the above three features need to be converted by one hot encoding, and after conversion, each dataset will change from 41 features to 122 features. It is also observed that the eigenvalue difference is too large, so the feature value is reduced to the interval of 0 to 1, using the MinMaxScaler function. The final classification class turns normal behavior into 0, and the major attack behaviors are converted to 1, 2, 3, and 4, respectively.

3.3 Neural Network and Metaheuristic Parameter Settings

Recently, the long short-term memory (LSTM) [10] has been proved to show better results than RNN in solving problems. In order to make comparisons, the deep learning algorithm of this paper adopts DNN and LSTM. There are many parameters in the neural network, such as optimizer, learning rate, number of neurons and the number of hidden layers. As for the optimizer, it is used in solving various datasets is not the same. So after the experiment, it is found that the accuracy of using RMSProp in LSTM is better than using Adam. However, the accuracy of using Adam is better than RMSProp in DNN. Therefore, LSTM's optimizer uses RMSProp , and DNN uses Adam.

As for the learning rate, if the value is too large, it will cause back and forth oscillation. If it's too small, the convergence speed will be too slow, so we tested the learning rate from 0.01 to 0.1. In order to avoid the learning rate changing too much, the learning rate will be averaged in five times. After the experiment, it was found that the best result of learning rate of LSTM is 0.05; the best accuracy of DNN is 0.01. Therefore, 0.05 and 0.01 would be used in the learning rate of LSTM and DNN. In metaheuristic, DE parameters are an iteration, population, F, and CR, which are set to 40, 20, 0.5, and 0.3, respectively.

3.4 Experimental Results

The dataset used in this paper is part of NSL-KDD dataset. And it is divided into 1 training data and 2 testing data. In training data, there is dataset labeled as "full NSL-KDD train", and in testing data, there are dataset labeled as "full NSL-KDD test" **(DS1)** and "subset of the KDDTest" **(DS2)** in Table 2. In this paper, there are two methods, and one is the deep learning, and the other one is a heuristic algorithm for dynamically adjusting parameters. In the experiment, it was found that the results obtained by the deep learning were often different, and the methods were performed three times and averaged in order to be fair. In the dataset (DS1) dichotomy, the accuracy of the original DNN model is found

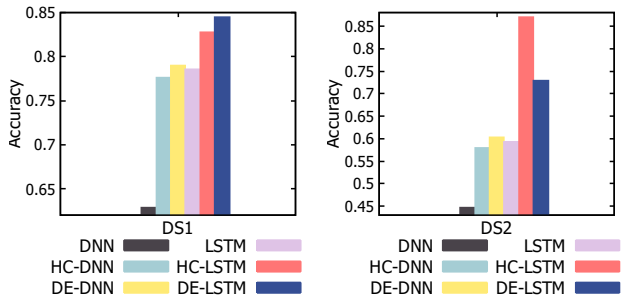
Table 2. Number of record in dataset

DS Name	Label	Number of record
Training	Normal	67343
	Probe	11656
	DOS	45927
	U2R	52
	R2L	995
Testing DS1	Normal	9711
	Probe	2421
	DOS	7458
	U2R	200
	R2L	2754
Testing DS2	Normal	2152
	Probe	2402
	DOS	4342
	U2R	200
	R2L	2754

to be about 62% lower, and the accuracy of using the original LSTM model is about 78.5% which is much higher than the original DNN model. It proves that LSTM with time series is better than DNN. In the above model, we added a heuristic algorithm, and the accuracy also increased significantly. After the DE adjustment of the DNN, the accuracy can reach about 79%, and the accuracy of the DE adjustment LSTM can reach 84.5%, as show in Figure 3 (a). In the dataset (DS2), since the two classifications are only a simple distinction, the accuracy of using the HC to adjust the LSTM is about 87% higher than that of the DE adjustment LSTM, and the result is as shown in Figure 3 (b). We also apply our proposed model to multiple classifications. In the multi-category, we can find that the accuracy of using DE to adjust DNN parameters cannot be exceeded using LSTM alone. The accuracy of using LSTM alone is about 76.4%. However, the proposed algorithm for LSTM can improve the accuracy rate up to about 81.4%, as shown in Figure 4.

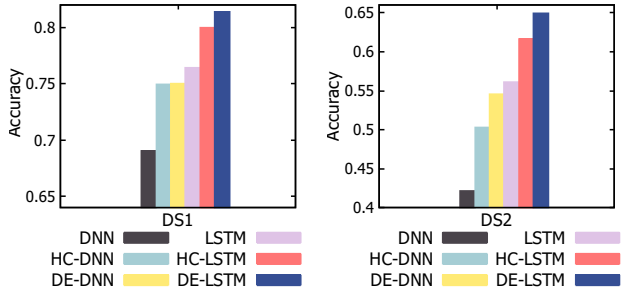
4 CONCLUSION

In this paper, we proposed a method of adjusting parameters in deep learning and used the most common NSL-KDD dataset in the intrusion detection system for verification. In the experimental results, we can find that the proposed method can bring higher accuracy of the intrusion detection system. Although the use of deep learning models alone does not have the same high accuracy as [15], the final accuracy of our method is indeed higher than that of pure deep learning model. Since this paper only dynamically adjusts the number of hidden layer neurons, there are still many parameters require for setting in the deep learning, such as learning rate and the number of hidden layers. The main advantage of the proposed algorithm, DE-LSTM, is that the deep learning parameters



(a) The accuracy of DS1 (b) The accuracy of DS2

Fig. 3. The result of experiments in dataset-2 categories



(a) The accuracy of DS1 (b) The accuracy of DS2

Fig. 4. The result of experiments in dataset-5 categories

can be dynamically adjusted in various environments to achieve more accuracy. It is anticipated that the method proposed in this paper can be applied to various fields to reduce the manual set the accuracy of the parameter reduction. In future work, there are two direction for the development of the proposed method, including the setting of parameter and the selection of metaheuristic algorithm. First part is the setting of parameter. To compare with essay [15], the network architecture in this paper focus on a simple neural network with single hidden layer. The proposed method utilizes a metaheuristic algorithm to tune the number of neurons in individual layer, and the result show the enhancement of accuracy in several dataset. In despite of the improvement of accuracy in the proposed method, there are still many parameters that can be adjusted in different deep learning neural network. Take a simple multilayer perceptron (MLP) for instance, the number of hidden layers, learning rate, and activation function also require for adjustment. Second part is the selection of metaheuristic

algorithm. DE algorithm is widely used to solve a variety of problem in many papers because of its strength in find better solution, and this is the reason why DE is adopted in the proposed method. Regardless of the advantage of DE, there are still many metaheuristic algorithm that reach to a better result in solving different problem, such as genetic algorithm (GA), search economics (SE), and ant lion optimizer (ALO), etc.

Acknowledgment

This work was supported in part by the Ministry of Science and Technology of Taiwan, R.O.C., under Contracts MOST 107-2221-E-110-078, MOST107-2221-E-005-022, and MOST 107-2218-E-005-018.

References

1. Ali, M.H., Mohammed, B.A.D.A., Ismail, A., Zolkipli, M.F.: A new intrusion detection system based on fast learning network and particle swarm optimization. IEEE Access **6**, 20255–20261 (2018)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Machine Learning Research pp. 281–305 (2012)
3. Fong, S., Deb, S., Yang, X.S.: How Meta-heuristic algorithms contribute to deep learning in the hype of big data analytics. In: Proceedings of the Progress in Intelligent Computing Techniques: Theory, Practice, and Applications. pp. 3–25 (2018)
4. Lee, W.P., Chien, W.J.: A novel differential evolution algorithm with co-evolution strategy. Journal of Computers **6**(3), 594–602 (2011)
5. Li, H., Ota, K., Dong, M.: Learning iot in edge: Deep learning for the internet of things with edge computing. IEEE Network **32**(1), 96–101 (2018)
6. Liao, H.J., Lin, C.H.R., Lin, Y.C., Tung, K.Y.: Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications **36**(1), 16 – 24 (2013)
7. Lunt, T.F.: A survey of intrusion detection techniques. Computers and Security **12**(4), 405–418 (1993)
8. Rere, L.M.R., Fanany, M.I., Arymurthy, A.M.: Simulated annealing algorithm for deep learning. Procedia Computer Science **72**, 137–144 (2015)
9. Rere, L.M.R., Fanany, M.I., Arymurthy, A.M.: Metaheuristic algorithms for convolution neural network. Computational Intelligence and Neuroscience pp. 1–13 (2016)
10. Sak, H., Senior, A.W., Beaufays, F.: Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. Computer Research Repository **abs/1402.1128** (2014)
11. Storn, R., Price, K.: Minimizing the real functions of the icec'96 contest by differential evolution. In: Proceedings of IEEE International Conference on Evolutionary Computation. pp. 842–844 (1996)
12. Tian, Z., Fong, S.: Survey of meta-heuristic algorithms for deep learning training. In: Baskan, O. (ed.) Optimization Algorithms, chap. 9. IntechOpen, Rijeka (2016). https://doi.org/10.5772/63785, https://doi.org/10.5772/63785

13. Tuba, E., Tuba, M., Simian, D.: Adjusted bat algorithm for tuning of support vector machine parameters. In: Proceedings of the IEEE Congress on Evolutionary Computation. pp. 2225–2232 (2016)
14. Yang, X.S., Deb, S., Fong, S.: Accelerated particle swarm optimization and support vector machine for business optimization and applications. In: Proceedings of the Networked Digital Technologies. pp. 53–66 (2011)
15. Yin, C.L., Zhu, Y.F., Fei, J.L., He, X.Z.: A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access **5**, 21954–21961 (2017)
16. Young, S.R., Rose, D.C., Karnowski, T.P., Lim, S.H., Patton, R.M.: Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments. pp. 1–5 (2015)
17. Zhang, Q., Yang, L.T., Chen, Z., Li, P.: A survey on deep learning for big data. Information Fusion **42**, 146–157 (2018)