Journal of Mathematical Modeling
Vol. 9, No. 1, 2021, pp. 437-450

JMM

An intrusion detection system with a parallel multi-layer neural network

Mohammad Hassan Nataj Solhdar[†], Mehdi Janinasab Solahdar[‡], Sadegh Eskandari[§*]

[†] *Shohadaye Hoveizeh University of Technology, Dasht-e Azadegan, Khuzestan, Iran*
[‡] *Islamic Azad University, Mahalat Branch, Mahalat, Iran*
[§] *Department of Computer Science, University of Guilan, Rasht, Iran*

Email(s): nataj.solhdar@gmail.com, mehdi_janinasab@yahoo.com, eskandari@guilan.ac.ir

Abstract. Intrusion detection is a very important task that is responsible for supervising and analyzing the incidents that occur in computer networks. We present a new anomaly-based intrusion detection system (IDS) that adopts parallel classifiers using RBF and MLP neural networks. This IDS constitutes different analyzers each responsible for identifying a certain class of intrusions. Each analyzer is trained independently with a small category of related features. The proposed IDS is compared extensively with existing state-of-the-art methods in terms of classification accuracy . Experimental results demonstrate that our IDS achieves a true positive rate (TPR) of 98.60% on the well-known NSL-KDD dataset and therefore this method can be considered as a new state-of-the-art anomaly-based IDS.

Keywords: Intrusion detection, computer security, neural network, parallel processing.
AMS Subject Classification 2010: 68T05, 68T10.

1 Introduction

Intrusion Detection Systems (IDS) are currently the most basic components of a networks monitoring system [1,35]. The term intrusion is applied to an operation which tries to pass through the systems security in order to have illegal access to network or computer system [24]. This operation is performed by external and internal intruders. Intrusion detection is considered as a tool for supervising and analyzing the incidents which occur in a computer system in order to identify the signs of the security issues [18].

IDSs can be classified into network-based, host-based and distributed-based systems. Network-based systems detect the attacks by analyzing network-related incidents, such as traffic congestion, IP addresses and port services [28]. Host-based systems protect a host by monitoring the

*Corresponding author.
Received: 12 August 2020 / Revised: 30 December 2020 / Accepted: 21 January 2021
DOI: 10.22124/jmm.2021.17362.1502

network interface and controlling external accesses to the system [29]. Distributed-based systems consist of several network-based or host-based, or a combination of the two, with a central managing station [22]. In these systems each IDS sends its own reports to the central managing station and the central station is responsible for studying the reports and informing the security unit(s) of the system [29].

Considering detection methods, IDSs are also classified into signature-based and anomaly-based methods. Signature-based methods continuously compare current behaviors with the previously observed data resulted from unexpected behaviors. Therefore, these methods require a dataset, corresponding to the signature that identifies the attacks [10, 25]. Anomaly-based methods try to learn the model of normal behaviours. Any behaviour that does not follow the normal model, is considered as an anomalous behaviour. Typically, an incident with more than two deviations from standard behavior is assumed to be anomalous [30]. Although the anomaly-based IDSs would cause a number of false alarms, they are able to discover new attack types. Therefore, these methods attracted much of interest in recent years.

Several anomaly-based IDSs are proposed in the literature [2,3,8,11,12,20,24,33]. Although these systems are able to handle most of the detection tasks, the accuracy of detecting abnormal behaviours is still an open problem and we need more accurate methods with better true positive rates. In this paper, to increase the accuracy, we present a new IDS. The proposed IDS adopts four neural networks-based classifiers to analyze four different intrusion types in a parallel manner. Besides the parallel classifiers, our method uses an information-theoretic based feature selection to increase the classification speed and generalization performance. Our IDS achieves a true positive rate (TPR) of 98.60% on the well-known NSL-KDD dataset and therefore this method can be considered as a new state-of-the-art anomaly-based IDS.

The rest of the paper is organized as follows. Section 2 discusses the related work. MLP and RBF networks are defined in section 3. Section 4 introduces the datasets we used for training and testing our proposed IDS. Section 5 presents the proposed IDS. Section 6 demonstrates the experimental results and Section 7 concludes the paper.

2 Related work

Due to the fast development of machine learning and its success in solving many real world hard problems, anomaly-based IDSs have attracted much of interest in recent years. Sharma and Mukherjee [24] proposed layered (serial) IDS by combining nave bayes classifier (NBC) and naive bayes tree (NBTree). Chebrolu et al. [6] demonstrated the effects of removing redundant and irrelevant features on the accuracy of IDSs and then proposed a hybrid feature selection method based on Bayesian networks (BN) and classification and regression trees (CART). Liu et al. [17] presented a hierarchical IDS model using principal component analysis (PCA) neural networks that detects both anomaly and misuse attacks. In their model, PCA is applied for classification and neural networks are used for online computing. Panda et al. [19] presented a combinational intelligent method in addition to a combination of classifiers for intelligently making decisions. Yuan et al. [33] proposed a two layers multi-class detection method, called TLMD, that uses the C5.0 and naive Bayes methods for adaptive network intrusion detection. Almiani et al. [2] proposed a layered hybrid intrusion detection model that uses cascaded layers

of clustered selfOrganized map (CSOM) and radial basis function (RBF) neural networks. They adopted a Kmeans clustered SOM as the first layer and an RBFbased neural network as the second filtering layer. Chowdhury et al. [8] proposed a method that uses features extracted from different layers of a pre-trained deep network as inputs for an SVM+1NN classifier. Yaseen et al. [3] proposed a multi-level hybrid intrusion detection model that uses support vector machine and extreme learning machine. Their method adopts a modified K-means algorithm to build new small training datasets representing the entire original training dataset. Papamartzivanos et al. [20] proposed a method called Dendron, that uses a combination of genetic algorithm (GA) and decision tree (DT). Using GA, Dendron is able to deal with the challenging nature of the network traffic, which generally biases machine learning techniques to neglect the minority classes of a dataset. Ghanem and Jantan [12] proposed an IDS that uses an enhanced Bat algorithm (EBat) for training an artificial neural network.

3 Algorithms in neural networks

As mentioned before, our method adoptes two types of supervised neural networks, namely MLP [4] and RBF [31]. This section describes the two networks in detail

3.1 MLP neural network

MLP is a well-known neural network which performs the classification operation desirably. An MLP, is a feed forward neural network [31] with one or more hidden layer(s). This network consists of an input layer, at least one middle or hidden layer for computations and an output layer. Input signals are distributed forwards from one layer to another. In multilayer networks, layers are connected to each other one after the other in such a way that the outputs of the first layer form the inputs of the second layer and so forth to the end where the outputs of the last layer form the main outputs and real answer of the network. In other words, network signals proceed in a feed forward path which starts with an input layer and ends with an output layer. Typically, back propagation training algorithm is used for training MLP neural network [4].

3.2 RBF neural network

RBF neural network is a network with unidirectional architecture in the first layer of which a linear mapping with Gaussian function is carried out, and the classification (usually linear) is performed in its second layer. The main architecture of RBF neural network consists of a three-layer network that is shown in Figure 1.

The first layer is just an input layer in which there is no processing operation. The second layer or the hidden layer forms a nonlinear correspondence between the input space and a space with typically wider dimensions and plays a more important role in changing nonlinear patterns into linear separable patterns. Finally, the third layer generates the weighted sum with a linear output. An output like this will be efficient if RBF is used for approximation function. RBF requires more neurons as compared with MLP, but it is often possible to be designed in such a way that it could take the standard feed-forward network in a fraction of time for training [9]. Unlike MLP network which applies Logsig or Tansig activation functions,

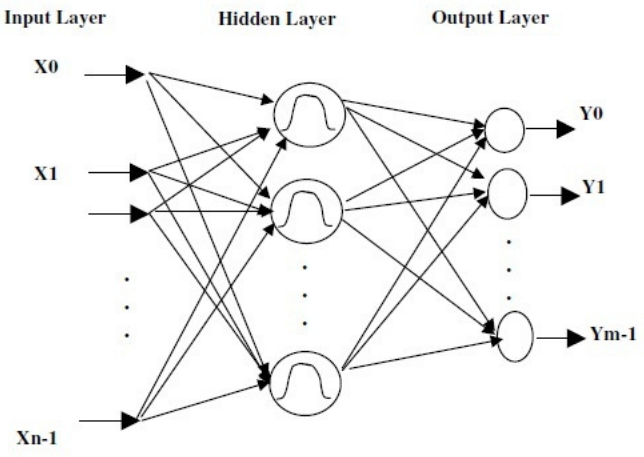

Figure 1: The structure of a RBF network [34].

the activation function in RBF network is a nonlinear Gaussian function. In this network, the connection between neurons of input and hidden layers are not as simple as in MLP. Neurons of the hidden layer are multidimensional units and the dimension of these neurons is equal to the number of the networks inputs. Simply speaking, due to Gaussian functions specific shape, each neuron of the hidden layer has the bigger output only when the networks input vector is closer to Gaussian function center of that neuron; and the more the distance of the input vector from the center of this nonlinear function, the less the output neuron. This distance can be defined by the criterion of Euclidean distance.

4 The NSL-KDD nataset

NSL-KDD is a standardized KDD datasets in which some problems of KDD99 dataset such as duplicate connections have been fixed [26]. The most important properties of NSL-KDD are as follows:

1. There are no duplicate records in the training set, so in training, we will not have duplicate data.
2. There are no duplicate records in the testing set, so the IDS performance of the will be estimated more accurately.
3. The distribution of records in the NSL-KDD training set and testing set is acceptable.

For example, there are four known categories of intrusions in the dataset proportionally. The dataset contains 125973 training and 22544 testing samples which are distributes as Table 1.

5 The proposed intrusion detection system

Our layered and parallel IDS, uses RBF and MLP neural networks for detecting majority and minority attacks, respectively. Figure 2, represents the general structure of our proposed IDS

Table 1: Distribution of the training and testing samples in NSL-KDD dataset.

Type of Sample	Number (Percentage) of Train Samples	Number (Percentage) of Test Samples
Normal	67343 (53%)	9712 (43 %)
Dos	45948 (36%)	8402 (37%)
Prob	11657 (9.2%)	2422 (10%)
U2R	53 (0.05%)	68 (0.3%)
R2L	976 (0.07%)	1941 (0.8%)

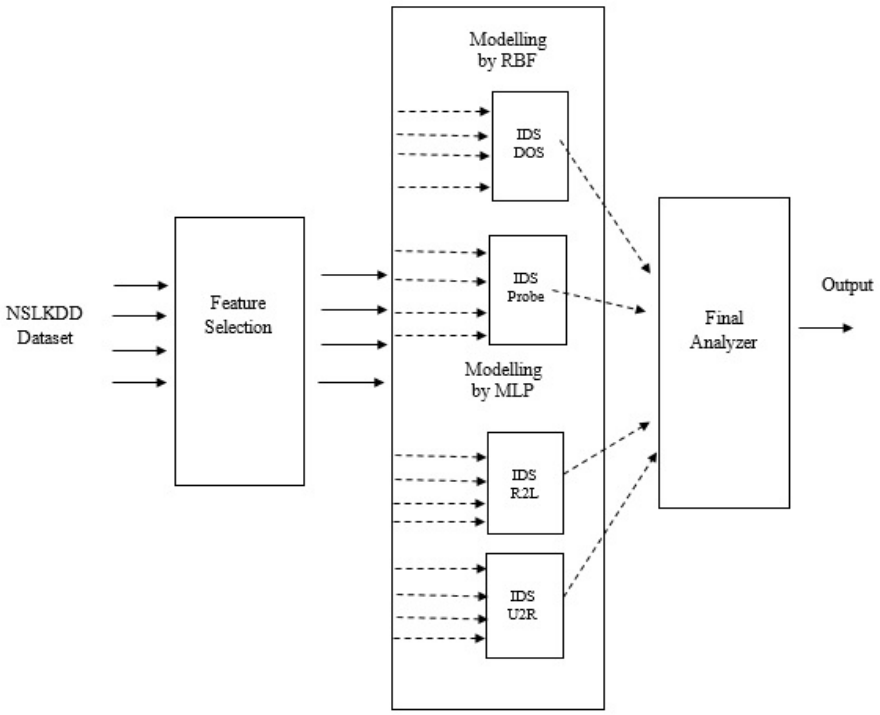

Figure 2: Proposed intrusion detection system.

framework. The framework constitutes three main steps, 1) feature selection, 2) parallel intrusion detection and 3) final analysis. The task of feature selection is to select the most important and discriminative features among all the feature space. The parallel intrusion detection step constitutes four independent IDSs, each responsible to analyze a sub-class of intrusions. The last step makes the final decision about the intrusion type based on the results reported by the four IDSs.

5.1 Feature selection

In complex machine learning domains, the feature space may contain several irrelevant and redundant features. The irrelevant features contain false correlations, which hinder the process of detecting intrusions. On the other hand, the redundant features add information that is

contained in other features. These extra features can increase computation time, and can impact the accuracy of IDS [7]. Feature selection improves the learning task by searching for the subset of features, which best subsumes the training data. Our proposed IDS adopts a well-known information theoretic-based features selection algorithm called mRMR [21]. Let F and $S \subseteq F$ represent the set of all the input features (feature space) and the set of the selected features, respectively (S is initially empty). The mRMR algorithm selects the next feature among the set of unselected features ($F - S$) using the following criteria:

$$f^* = \arg \max_{f \in F-S} I(f;Y) - \frac{1}{|S|-1} \sum_{f' \in S} I(f;f'), \tag{1}$$

where, Y is the class label and $I(X;Y)$ represents the mutual information between X and Y:

$$I(X;Y) = \int_Y \int_X p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{2}$$

Eq. (1) tries to select the most informative (the term $I(f,Y)$) and least redundant feature among unselected feature subset.

5.2 Parallel intrusion detection

In order to reach a higher true intrusion detection rate, we designed four independent IDSs, namely IDS-DOS, IDS-PROBE, IDS-R2L and IDS-U2R which are responsible for identifying DOS, probing, R2L and U2R attacks, respectively. Such a design enables us to keep the detection sub-models as small as possible and hence avoiding the over-fitting problem. Moreover, the parallel nature of the model makes it possible to perform the sub-models on different processors and consequently improve the speed.

The IDS-DOS is an RBF containing a single hidden layer with 15 neurons. This network has six inputs corresponding to the six features suggested in [16] (the features are listed in Appendix 1). The ISD-PROBE is also an RBF with 24 neurons in its hidden layer and six inputs. The input features for this network are the same as for IDS-DOS. R2L and U2R are both MLPs, each with two hidden layers. R2L is fed by 19 input features and contains 5 and 9 neurons in its first and second hidden layers, respectively. On the other hand, U2R is fed by 12 input features and has 8 and 5 neurons in its hidden layers.

5.3 Final Analysis

The proposed intrusion detection system has a final analyzer which receives the results generated by the four IDSs and determines the type of the attack if there is any. This analyzer is a priority encoder that generates the output based on Table 2. In this table, X represents a don't care value. The priorities of attacks are defined based on their incidence rate. For example, DOS is the most frequent and severe one [27] and hence has the highest priority in our IDS. On the other hand, ,U2R is the most rare attack and therefore has the lowest priority.

Table 2: The results generated by the final analysis phase in the proposed IDS.

IDS DOS	IDS PROBE	IDS R2L	IDS U2R	Output
1	X	X	X	DOS
0	1	X	X	PROBE
0	0	1	X	R2L
0	0	0	1	U2R
0	0	0	0	No Attack

6 Experimental results

In this section, we provide experimental results to illustrate the performance of the proposed method. In this regard, we provide three sets of experiments. Firstly, as preliminary experiments, we explore the effects of parallelization and feature selection on the accuracy and efficacy of the proposed IDS. Secondly, we compare our proposed IDS with nine state-of-the-art IDSs from the literature. Finally, we explore the optimal number of processors for parallel execution of the proposed IDS. All the NSL-KDD data is normalized using the following equation:

$$X_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

where, $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of training date, respectively. Using such a normalization, different features become independent of scale and play equal role in the classification. All the methods in this paper are implemented in MATLAB 7.10.0 version (2010a) running on a computer system with a 2.53 GHz two-core processor and 4G RAM. The neural networks are trained using stochastic gradient descent (SGD) and minimum squared error (MSE) as optimization and loss function, respectively. For evaluating the different IDSs, two criteria, namely false positive rate (FPR) and true positive rate (TPR) are reported. These criteria are defined as follows:

$$FPR = \frac{FP}{TN + FP}, \qquad TPR = \frac{TP}{TP + FN}$$

where

True Positive (TP): the number of intrusion samples classified as intrusions,

True Negative (TN): the number of normal samples classified as normal,

False Positive (FP): the number of normal samples classified as intrusions,

False Negative (FN): the number of intrusion samples classified as normal.

It is worth noting that TPR is a more important criterion than FPR for intrusion detection tasks. This is because of the fact that classifying an intrusion as normal will have more serious implications than classifying a normal package as an intrusion.

6.1 Preliminary experiments

Here, we explore the effects of parallelization and feature selection on the accuracy and efficacy of the proposed IDS. In this regard the proposed IDS is compared with it's two relaxed versions:

Table 3: TPR, FPR and speed comparisons for the three different IDS implementations.

IDS Version	Classification Accuracy		Running Times (Minutes)	
	TPR	FPR	Single Processor	Multi-Processors
Single MLP-Based IDS	87.3	0.36	89.57	–
Parallel IDSs Without Feature Selection	95.5	0.5	97.35	47.39
Parallel IDSs With Feature Selection (Proposed IDS)	**98.6**	0.82	**63.47**	**29.11**

1. **A single MLP-based IDS**: This version constitutes three phases (similar to our IDS) but, instead of the four parallel small IDSs, it uses a single large MLP for detecting all the intrusions. This MLP contains three hidden layers with 12, 7 and 5 neurons in its first, second and third layers, respectively. Moreover, this network is fed by 17 input features and generates 4 outputs (each output corresponds to an intrusion type).

2. **Parallel IDSs without feature selection**: This version does not apply feature selection and hence has two phases. The first phase constitutes four parallel IDSs, similar to those used in our proposed IDS, but each IDS is fed by all the 41 input features.

Table 3 reports the TPR, FPR and running times (in minutes) for the three IDS implementations. For running times, we reported the execution times in two running modes: single processor and multi-processors. For multi-processors mode, we used four processors, each running an IDS for parallel IDSs (for the single MLP-based IDS we didn't apply this mode). Comparing TPR results in the table, we can see an increase of 8.2% when we us parallelization. This can be attributed to the fact that simpler learning models do not over-fit to the training data and therefore they show more generalization performance than a single large model. On the other hand, comparing TPR results for the two parallel IDSs, we can see the impact of features selection on classification accuracy. Using feature selection, we are able to remove redundant and irrelevant features and therefore, we are able to avoid a problem called "*curse of dimensionality*" [15] and hence improve the classification accuracy (an increase of 3.1% in TPR value).

Comparing the running times in single processor mode, again we see the impact of feature selection. Although the parallelization increases the running time about 8 minutes, feature selection decreases it more than 30 minutes. Moreover, considering the multi-processors mode, we can see the impact of parallelization on the performance of the proposed IDS. The independent behavior of small IDSs enables us to run each of them on a separate processor, which improves the speed of the detection process considerably.

6.2 Comparison with state of the art IDSs

Here we compare the proposed method with eight state of the art IDSs, namely K-means-NN [11], NBC-NBTree [24], LTMD [33], CSOMRBF [2], DNN-SVM [8], SVM+ELM [3], Dendron [20] and EBAT-MLP [12]. For the K-means-NN, we set the number of clusters for all the attack types to be $K = 6$ and the number of selected samples from each cluster to be $S = 200$. In the NBC-NBTree, we used the NBC for major attacks detection and NBTree for minor attack detection. For the SVM+ELM, the distance threshold for the modified K-means algorithm is set to 0.5. For the Dendron algorithm, we set $P_c = 0.95$ (crossover probability), $P_m = 0.1$ (mutation probablity), $\lambda = 15$ (individuals per class), $\beta = 40$ (beta weight) and $\theta = 2\%$

Table 4: Comparison of the proposed IDS with other SOA methods.

IDS	TPR	FPR
K-means-NN	93.83	9.88
NBC-NBTree	93.41	0.275
LTMD	93.32	0.06
CSOMRBF	95.27	0.23
DNN-SVM	94.62	0.235
SVM+ELM	95.75	0.102
Dendron	95.97	1.05
EBAT-MLP	97.20	0.022
Our IDS	**98.60**	0.82

(additional individuals). For the EBAT-MLP method, we set $A = 0.95$ (loudness), $r = 0.1$ (pulse rate), $limit_1 = 0.8$ and $limit_2 = 0.5$. The classification results, presented in Table 4, show that the proposed IDS performs very well and shows a significant increase in TPR value. Although the FPR value is not the best one for our method, we should note again that TPR is a more important criterion than FPR for IDS tasks. Classifying a normal package as intrusion is an error that can be corrected by the operator in the next steps. However, classifying an intrusion as a normal package can have irreparable consequences for the system.

6.3 The effects of the number of processors on the running times

Parallel nature of the proposed IDS enables us to perform the individual sub-models on different processors and consequently improve the speed. Here, we investigate the effect(s) of the number of processors on the running time of the proposed IDS. In this regard, we vary the number of processors in the interval 1-8, recording the running time of our IDS. Figure 3, demonstrates the recorded running times. Comparing with serial configuration (single processor), we can see an speedup of 32%, 54% and 58% using two, three and four processors, respectively. However, using more than four processors, does not affect the running time significantly. This can be attributed to the number of sub-models in our IDS. As the number of processors increases from one to four, the sub-models are able to be distributed on the processors and run in parallel. This greatly increases the execution speed. But when the number of processors exceeds 4, there is no proper distribution on processors and as a result, new processors are practically idle. Therefore, the increase in speed is significantly reduced.

7 Conclusions

In this paper, we presented a new anomaly-based IDS that constitutes three main steps, 1) feature selection, 2) parallel intrusion detection and 3) final analysis. For feature selection we adopted the well known mRMR algorithm. For the parallel intrusion detection step we used four independent IDSs, each responsible to analyze a sub-class of intrusions. For the last step, we adopted a priority encoder that generates the output based on the results of the parallel IDSs and

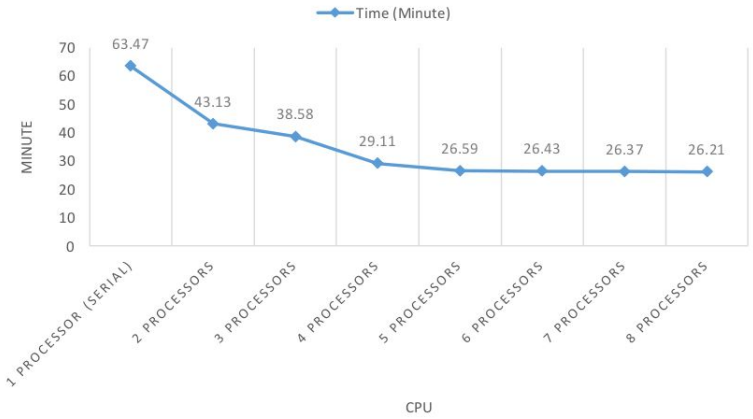

Figure 3: The effect(s) of number of processors on the running time of the proposed IDS.

their importance. According to experiments we concluded that the feature selection step plays an important role in accuracy and running time of the proposed IDS. Moreover, we concluded that the independent behavior of small IDSs enables us to run each of them on a separate processor, which improves the speed of the detection process considerably. More experiments demonstrated that our IDS achieves a TPR of 98.60% on the well-known NSL-KDD dataset, which is a new record for this dataset.

References

[1] S. Aljawarneh, M. Aldwairi, M. B. Yassein, *Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model*, J. Comput. Sci-Neth. **25** (2018) 152–160.

[2] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, A. Razaque, *Cascaded hybrid intrusion detection model based on SOM and RBF neural networks*, Concurr. Com. Pract. E. **32** (2020) 5233.

[3] W.L. Al-Yaseen, Z.A. Othman, M.Z.A. Nazri, *Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system*, Expert. Syst. Appl. **67** (2017) 296–303.

[4] R. Beghdad, *Critical study of neural networks in detecting intrusions*, Comput. Secur. **27** (2008) 168–175.

[5] J. Cannady, *Artificial neural networks for misuse detection*, In Proceedings of the 1998 National Information Systems Security Conference (NISSC'98), pages 443-456, Arlington, VA, 1998.

[6] S. Chebrolu, A. Abraham, J.P. Thomas, *Feature deduction and ensemble design of intrusion detection systems*, Comput. Secur. **4** (2005) 295–307.

[7] S. Chebrolu, A. Abraham, J.P. Thomas, *Feature deduction and ensemble design of intrusion detection systems*, Comput. Secur. **24** (2005) 295–307.

[8] M.M.U. Chowdhury, F. Hammond, G. Konowicz, C. Xin, H. Wu, J. Li, October, *A few-shot deep learning approach for improved intrusion detection*, In Proceedings of the 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, USA, pp. 456462, 2017.

[9] H. Demuth, M. Beale, *Neural network toolbox for use with matlabusers guide verion 3.0*, 1993.

[10] D.E. Denning, *An intrusion-detection model*, IEEE. T. Software. Eng. **2** (1987) 222–232.

[11] K. Faraoun, A. Boukelif, *Neural networks learning improvement using the kmeans clustering algorithm to detect network intrusions*, Infocomp. J. Comput. Sci. **5** (2006) 28–36.

[12] W.A. Ghanem, A. Jantan, *A new approach for intrusion detection system based on training multilayer perceptron by using enhanced Bat algorithm*, Neural. Comput. Appl. **32** (2019) 1–34.

[13] M.M. Javidi, M.H. Nattaj, *A new and quick method to detect dos attacks by neural networks*, J. Math. Comput. Sci. JM. **6** (2013) 85–96.

[14] M.M. Javidi, M.H. Nattaj, *Network attacks detection by hierarchical neural network*, Com. Eng. App. **2** (2015) 119–132.

[15] M.M. Javidi, S. Eskandari, *Online streaming feature selection: a minimum redundancy maximum significance approach*, Pattern. Anal. Appl. **3** (2019) 949–963.

[16] H.G. Kayacik, A.N. Zincir-Heywood, M.I. Heywood, *A hierarchical SOM-based intrusion detection system*, Eng. Appl. Artif. Intell. **20** (2007) 439–451.

[17] G. Liu, Z. Yi, S. Yang, *A hierarchical intrusion detection model based on the pca neural networks*, Neurocomputing. **70** (2007) 1561–1568.

[18] S.-N. Nguyen, V.-Q. Nguyen, J. Choi, K. Kim, *Design and implementation of intrusion detection system using convolutional neural network for dos detection*, In Proceedings of the 2nd International Conference on Machine Learning and Soft Computing, CMLSC 2018. 34–38 ACM, 2018.

[19] M. Panda, A. Abraham, M.R. Patra, *A hybrid intelligent approach for network intrusion detection*, Procedia. Eng. **30** (2012) 1–9.

[20] D. Papamartzivanos, F.G. Marmol, G. Kambourakis, *Dendron: Genetic trees driven rule induction for network intrusion detection systems*, Future. Gener. Comput. Syst. **79** (2018) 558–574.

[21] H. Peng, F. Long, C. Ding, *Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy*, IEEE. T. Pattern. Anal. Mach. Intell. **27** (2005) 1226–1238.

[22] C. Ranjithkumar, I. Jubitha, G. Dalin, *An analysis of intrusion detection system and security issues*, Int. J. Eng. Technol. Comput. Res. **5** (2017) 186-191 .

[23] A. Rapaka, A. Novokhodko, D. Wunsch, *Intrusion detection using radial basis function network on sequences of system calls*, IEEE IJCNN. **3** (2003) 1820–1825.

[24] N. Sharma, S. Mukherjee, *A novel multi-classifier layered approach to improve minority attack detection in ids*, Procedia. Technol. **6** (2012) 913–921.

[25] T. Sproull, J. Lockwood, *Distributed instrusion prevention in active and extensible networks*, IFIP International Working Conference on Active networks, IWAN. 54–65. Springer, 2004.

[26] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, *A detailed analysis of the KDD CUP 99 data set*, In 2009 IEEE symposium on computational intelligence for security and defense applications (IEEE CISDA 2009) 1–6, July, 2009.

[27] A.N. Toosi, M. Kahani, *A new approach to intrusion detection based on an evolutionary soft computing model using neuro-fuzzy classifiers*, Comput. Commun. **30** (2007) 2201–2212.

[28] M. Toulouse, H. Le, C.V. Phung, D. Hock, *Defense strategies against byzantine attacks in a consensus-based network intrusion detection system*, Informatica **41** (2017) 193–207.

[29] N.N. Tran, R. Sarker, J. Hu, *An approach for host-based intrusion detection system design using convolutional neural network*, In International Conference on Mobile Networks and Management (9th International Conference, MONAMI), 116126. Springer, 2017.

[30] P.R.K. Varma, V.V. Kumari, S.S. Kumar, *A survey of feature selection techniques in intrusion detection system: A soft computing perspective*, Progress in Computing, Analytics and Networking (ICCAN 2017), 785793. Springer, 2018.

[31] S.X. Wu, W. Banzhaf, *The use of computational intelligence in intrusion detection systems: A review*, Appl. Soft. Comput. **10** (2010) 1–35.

[32] Z. Xue-qin, G. Chun-hua, L. Jia-jun, *Intrusion detection system based on feature selection and support vector machine*, Communications and Networking in China, 2006. ChinaCom06, First International Conference, 15. IEEE, 2006.

[33] Y. Yuan, L. Huo, D. Hogrefe, *Two layers multi-class detection method for network intrusion detection system*, IEEE Symposium on Computers and Communications. IEEE ISCC 2017. 767-772 IEEE, July, 2017.

[34] C. Zhang, J. Jiang, M. Kamel, *Intrusion detection using hierarchical neural networks*, Pattern. Recogn. Lett. **26** (2005) 779–791.

[35] Y. Zhou, G. Cheng, S. Jiang, M. Dai, *Building an efficient intrusion detection system based on feature selection and ensemble classifier*, Comput. Netw. **174** (2020) 107247.

Appendix: NSL-KDD Features

Tables 5 and 6 list the NSL-KDD dataset features. The features that are used to feed each of the sub-models (in our proposed IDS) are ticked.

Table 5: The list of NSL-KDD dataset features (part 1).

Name of network features	Descriptions	Feature label	IDS-DOS	IDS-PROBE	**IDS-R2L**	IDS-U2R
protocol_type	Type of protocol, for example, udp, tcp, etc.	A	✓	✓		✓
Service	Network service at the destination, for example, http, telnet, etc.	B	✓	✓		✓
Flag	Type of connection to the network	C	✓	✓	✓	
src_bytes	The number of bytes of data sent from source to destination	D	✓	✓		
dst_bytes	Number of bytes sent from destination to source	E	✓	✓	✓	✓
Land	1 if the connection from/to host/port is the same, otherwise zero	F	✓	✓	✓	✓
wrong_fragment	Number of wrong fragments	G				✓
Urgent	Number of urgent packages	H				✓
Hot	Number of host indexes	I				
num_failed_logins	Number of failed logins	J				
logged_in	1 if logged in; otherwise zero	K				✓
num_compromised	Number of compromised connections	L			✓	✓
root_shell	1 if root shell is obtained; otherwise zero	M				
su_attempted	1 if attempts are made to implement the su root instruction, otherwise zero	N				
num_root	Number of root access	O				
num_file_creations	Number of file creation operations	P				
num_shells	Number of shell command creations	Q				✓
num_access_files	Number of file access commands	R				
num_outbound_cmds	Number of outbound commands for access to ftp protocol	S				✓
is_host_login	1 if the input belongs to the host list, otherwise zero	T				✓
is_guest_login	1 if the user logins as a guest; otherwise zero	U				
Count	The number of connections to a host in the latest connections in the last two seconds	V				✓

Table 6: The list of NSL-KDD dataset features (part 2).

Name of network features	Descriptions	Feature label	IDS-DOS	IDS-PROBE	**IDS-R2L**	IDS-U2R
srv_serror_rate	The rate of connections with SYN error (server)	Y			✓	✓
rerror_rate	The rate of connections that have REJ error	Z				
srv_rerror_rate	The rate of connections that have REJ error (server)	AA				
same_srv_rate	Percentage of connections that have the same service	AB			✓	✓
diff_srv_rate	Rate of connections that have different service	AC				
srv_diff_host_rate	Rate of connections that have different hosts	AD				✓
dst_host_count	The number of connections from a host to the destination during a specific time	AE			✓	
dst_host_srv_count	The number of connections from a host to destination for access to service	AF			✓	✓
dst_host_same_srv_rate	The rate of connections from a host to the destination to have access to the same service	AG			✓	✓
dst_host_diff_srv_rate	The rate of connections from a host to the destination to have access to various services	AH				
dst_host_same_src_port_rate	The rate of connections from a host to the destination from the same port	AI			✓	✓
dst_host_srv_diff_host_rate	The rate of connections from a host to various destinations to have access to service	AJ				
dst_host_serror_rate	The rate of connections from a host to a specific destination that has SYN error	AK				
dst_host_srv_serror_rate	The rate of connections from a host with a specific service to a destination that has SYN error	AL				
dst_host_rerror_rate	The rate of connections from a host to a specific destination that has REJ error	AM				
dst_host_srv_rerror_rate	The rate of connections from a host with a specific service to a destination that has REJ error	AN				
Duration	Duration of connection in seconds	AO				