# A Hybrid Network Anomaly and Intrusion Detection Approach Based on Evolving Spiking Neural Network Classification

Konstantinos Demertzis(✉) and Lazaros Iliadis

Department of Forestry and Management of the Environment
and Natural Resources, Democritus University of Thrace, 193 Pandazidou st.,
68200 N. Orestiada, Greece
{kdemertz,liliadis}@fmenr.duth.gr

**Abstract.** The evolution of network services is closely connected to the understanding and modeling of their corresponding traffic. The obtained conclusions are related to a wide range of applications, like the design of the transfer lines' capacity, the scalar taxing of customers, the security violations and the spotting of errors and anomalies. Intrusion Detection Systems (IDS) monitor and analyze the events in traffic, to locate indications for potential intrusion and integrity violation attacks, resulting in the violation of trust and availability of information resources. They act in a complimentary mode with the existing security infrastructure, aiming in the early warning of the administrator, offering him details that will let him reach proper decisions and correction actions. This paper proposes a network-based online system, which uses minimum computational power to analyze only the basic characteristics of network flow, so as to spot the existence and the type of a potential network anomaly. It is a Hybrid Machine Learning Anomaly Detection System (HMLADS), which employs classification performed by Evolving Spiking Neural Networks (eSNN), in order to properly label a Potential Anomaly (PAN) in the net. On the other hand it uses a Multi-Layer Feed Forward (MLFF) ANN to classify the exact type of the intrusion.

**Keywords:** Security · Network intrusion and anomalies · Machine learning · Evolving spiking neural networks · Multi-layer neural network

## 1 Introduction

An IDS [4] monitors network traffic for suspicious activity or anomalous behavior and alerts the system or network administrator accordingly. There are network based (NIDS) and host based (HIDS) intrusion detection systems. Some of them are looking for specific signatures of known threats, whereas others are spotting anomalies by comparing traffic patterns against a baseline. There are three basic approaches for designing and building IDS, namely: the Statistical, the Knowledge based and the Machine Learning one which has been employed in this research effort.

The concept of the Statistical-based systems (SBID) is simple: it determines "normal" network activity and then all traffic that falls outside the scope of normal is


A.B. Sideridis et al. (Eds.): E-Democracy 2013, CCIS 441, pp. 1–13, 2014.
DOI: 10.1007/978-3-319-11710-2_2

flagged as anomalous (abnormal). These systems attempt to learn network traffic patterns on a particular network. This process of traffic analysis continues as long as the system is active, so, assuming network traffic patterns remain constant, the longer the system is on the network, the more accurate it becomes. The Knowledge Based Intrusion Detection systems (KBIDES) classify the data vectors based on a carefully designed Rule Set or they use models obtained from past experience in a heuristic mode. The Machine Learning Anomaly Detection (MLAD) approach automates the analysis of the data vectors, and they result in the implementation of systems that have the capacity to improve their performance as time passes.

Artificial Intelligence and data mining algorithms have been applied as intrusion detection methods in finding new intrusion patterns [3, 11, 12, 17], such as clustering (unsupervised learning) [7, 13, 21] or classification (supervised learning) [5, 14, 18, 26]. Also, a few hybrid techniques were proposed like Neural Networks with Genetic Algorithms [23] or Radial Based Function Neural Networks with Multilayer Perceptron [1, 16]. Besides, other very effective methods exist such as Sequential Detection [22], State Space [15], Spectral Methods [27] and combinations of those.

This research effort aims in the development and application of an innovative MLAD approach towards the trace of anomalies in the network. The methodology will be using spiking (biologically inspired) Artificial Neural Networks (SANN). SANN are modular connectionist-based systems that evolve their structure and functionality in a continuous, self-organized, on-line, adaptive, interactive way from incoming information. Also, it can learn both data and knowledge in a supervised and/or unsupervised way. For the aforementioned reasons, many studies attempt to use SANN for practical applications, some of them demonstrating very promising results in solving complex real world problems [8, 19, 28].

The Hybrid Evolving Spiking Anomaly Detection Model (HESADM) that has been developed and discussed herein is based in the "*Thorpe*" neural model [24] which intensifies the importance of the spikes taking place in an earlier moment, whereas the neural plasticity is used to monitor the learning algorithm by using one-pass learning. In order to classify real-valued data sets, each data sample, is mapped into a sequence of *spikes* using the Rank Order Population Encoding (ROPE) technique [2, 25]. The topology of the developed eSNN is strictly feed-forward, organized in several layers and weight modification occurs on the connections between the neurons of the existing layers.

## 2 Rank Order Population Encoding

The ROPE method [2, 25] is an alternative to conventional rate coding scheme that uses the order of firing neuron's inputs to encode information which allows the mapping of vectors of real-valued elements into a sequence of spikes. Neurons organized into neuronal maps which share the same synaptic weights. Whenever the synaptic weight of a neuron is modified, the same modification is applied to the entire population of neurons within the map. Inhibition is also present between each neuronal map. If a neuron spikes, it inhibits all the neurons in the other maps with neighboring positions. This prevents all the neurons from learning the same pattern. When

propagating new information, neuronal activity is initially reset to zero. Then, as the propagation goes on, neurons are progressively desensitize each time one of their inputs fire, thus making neuronal responses dependent upon the relative order of firing of the neuron's afferents. More precisely, let A = $\{a_1, a_2, a_3 \ldots a_{m-1}, a_m\}$ be the ensemble of afferent neurons of neuron i and W = $\{w_{1,i}, w_{2,i}, w_{3,i} \ldots w_{m-1,i}, w_{m,i}\}$ the weights of the m corresponding connections; let mod $\in [0,1]$ be an arbitrary modulation factor. The activation level of neuron i at time t is given by Eq. (1):

$$\text{Activation}(i,t) = \sum_{j\in[1,m]} \text{mod}^{\text{order}(a_j)} w_{j,i} \tag{1}$$

where order($a_j$) is the firing rank of neuron $a_j$ in the ensemble A. By convention, order ($a_j$) = +8 if a neuron $a_j$ is not fired at time t, sets the corresponding term in the above sum to zero. This kind of desensitization function could correspond to a fast shunting inhibition mechanism. Whenever a neuron reaches its threshold, it spikes and inhibits neurons at equivalent positions in the other maps so that only one neuron will respond at any particular location. Every spike also triggers a time based Hebbian-like learning rule that adjusts the synaptic weights. Let $t_e$ be the date of arrival of the Excitatory PostSynaptic Potential (EPSP) at synapse of weight W and $t_a$ the date of discharge of the postsynaptic neuron.

$$\begin{aligned} &\text{if } t_e < t_a \text{ then} && dW = a(1-W)e^{-|\Delta o|\tau} \\ &\text{else} && dW = -aWe^{-|\Delta o|\tau} \end{aligned} \tag{2}$$

Where Δo is the difference between the date of the EPSP and the date of the neuronal discharge (expressed in term of order of arrival instead of time), a is a constant that controls the amount of synaptic potentiation and depression [2].

ROPE technique with receptive fields allows the encoding of continuous values by using a collection of neurons with overlapping sensitivity profiles [8, 28]. Each input variable is encoded independently by a group of one-dimensional receptive fields (Fig. 2). For a variable n, an interval $\left[I^n_{min}, I^n_{max}\right]$ is defined. The Gaussian receptive field of neuron i is given by its center μi:

$$\mu i = I^n_{min} + \frac{2i-3}{2} \frac{I^n_{max} - I^n_{min}}{M-2} \tag{3}$$

The width σ is given by Eq. (4):

$$\sigma = \frac{1}{\beta} \frac{I^n_{max} - I^n_{min}}{M-2} \tag{4}$$

where $1 \le \beta \le 2$ and the parameter β directly controls the width of each Gaussian receptive field.

Figure 1 depicts an example encoding of a single variable. For the diagram (β = 2) the input interval $\left[I^n_{min}, I^n_{max}\right]$ was set to [−1.5, 1.5] and M = 5 receptive fields were used. For an input value v = 0.75 (thick straight line in left figure) the intersection

points with each Gaussian is computed (triangles), which are in turn translated into spike time delays (right figure).

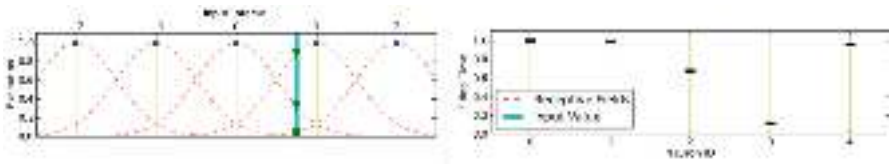

**Fig. 1.** Population encoding based on Gaussian receptive fields. Left Figure: Input Interval – Right Figure: Neuron ID [28]

## 3 One-Pass Learning

The aim of the one-pass learning method is to create a repository of trained output neurons during the presentation of training samples. After presenting a certain input sample to the network, the corresponding spike train is propagated through the SANN which may result in the firing of certain output neurons. It is also possible that no output neuron is activated and in this case the network remains silent and the classification result is undetermined. If one or more output neurons have emitted a spike, the neuron with the shortest response time among all activated output neurons is determined. The label of this neuron represents the classification result for the presented input sample. The procedure is described in detail in the following Algorithm 1 [8, 28].

**Algorithm 1.** Training an evolving Spiking Neural Network (eSNN) [28]

**Require:** $m_l$, $s_l$, $c_l$ **for a class label** l ∈ L

initialize neuron repository $R_l = \{\}$
**for all** samples $X^{(i)}$ belonging to class l **do**
  $w_j^{(i)} \leftarrow (m_l)^{order(j)}$, ∀ j | j pre-synaptic neuron of i
  $u_{max}^{(i)} \leftarrow \sum_j w_j^{(i)} (m_l)^{order(j)}$
  $\theta^{(i)} \leftarrow c_l u_{max}^{(i)}$
**if** $\min(d(w^{(i)}, w^{(k)})) < s_l, w^{(k)} \in R_l$ **then**
  $w^{(k)} \leftarrow$ merge $w^{(i)}$ and $w^{(k)}$ according to Equation 6
  $\theta^{(k)} \leftarrow$ merge $\theta^{(i)}$ and $\theta^{(k)}$ according to Equation 7
**else**
  $R_l \leftarrow R_l \cup \{w^{(i)}\}$
**end if**
**end for**

For each training sample i with class label l ∈ L a new output neuron is created and fully connected to the previous layer of neurons resulting in a real-valued weight vector $w^{(i)}$ *with* $w_j^{(i)} \in R$ denoting the connection between the pre-synaptic neuron j and the

created neuron i. In the next step, the input spikes are propagated through the network and the value of weight $w_j^{(i)}$ is computed according to the order of spike transmission through a synapse j: $w_j^{(i)} = (m_l)^{order(j)}$, $\forall j | j$ pre-synaptic neuron of i.

Parameter $m_l$ is the modulation factor of the Thorpe neural model. Differently labeled output neurons may have different modulation factors $m_l$. Function order(j) represents the rank of the spike emitted by neuron j. The firing threshold $\theta^{(i)}$ of the created neuron I is defined as the fraction $c_l \in R, 0 < c_l < 1$, of the maximal possible potential $u_{max}^{(i)}$:

$$\theta^{(i)} \leftarrow c_l u_{max}^{(i)} \tag{5}$$

$$u_{max}^{(i)} \leftarrow \sum_j w_j^{(i)} (m_l)^{order(j)} \tag{6}$$

The fraction $c_l$ is a parameter of the model and for each class label $1 \in L$ a different fraction can be specified. The weight vector of the trained neuron is then compared to the weights corresponding to neurons already stored in the repository. Two neurons are considered too "similar" If the minimal *Euclidean* distance between their weight vectors is smaller than a specified similarity threshold $s_l$ (the eSNN object uses optimal similarity threshold s = 0.6). All parameters modulation factor $m_l$, similarity threshold $s_l$, PSP fraction $c_l$, $1 \in L$ of ESNN which were included in this search space, are optimized according to the Versatile Quantum-inspired Evolutionary Algorithm (vQEA) [19]. In this case, both the firing thresholds and the weight vectors are merged according to Eqs. 7 and 8:

$$w_j^{(k)} \leftarrow \frac{w_j^{(i)} + N w_j^{(k)}}{1 + N}, \forall j \mid j \text{ pre-synaptic neuron of i} \tag{7}$$

$$\theta^{(k)} \leftarrow \frac{\theta^{(i)} + N\theta^{(k)}}{1 + N} \tag{8}$$

It must be clarified that integer N denotes the number of samples previously used to update neuron k. The merging is implemented as the (running) average of the

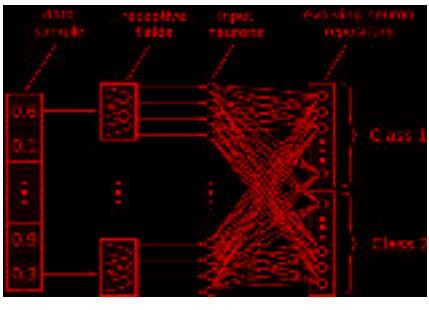

**Fig. 2.** The Evolving Spiking Neural Network (eSNN) architecture [28]

connection weights, and the (running) average of the two firing thresholds. After the merging, the trained neuron i is discarded and the next sample processed. If no other neuron in the repository is similar to the trained neuron i, the neuron i is added to the repository as a new output neuron.

Pattern recognition aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. Methods like classification, regression and clustering according to the type of learning procedure are used to generate the output value based on template matching, statistical classification, syntactic or structural matching and neural networks. The HESADM uses a two-layer feedforward neural network with sigmoid function both in hidden and output layer, scaled conjugate gradient backpropagation as the learning algorithm. The performance metric used is the Mean Squared Error (MSE).

## 4 Description of the HESADM Methodology

The HESADM methodology uses eSNN classification approach and Multi-Layer Feed Forward ANN in order to classify the exact type of the intrusion or anomaly in the network with minimum computational power.

The general methodology is described in detail below:

**Step 1:** We choose to use the traffic oriented data, which is related to only 9 features. We import the required classes that use the variable ***Population Encoding***. This variable controls the conversion of real-valued data samples into the corresponding time spikes. The encoding is performed with 20 Gaussian receptive fields per variable (Gaussian width parameter beta = 1.5). We also normalize the data to the interval [−1, 1] and so we indicate the coverage of the Gaussians using i_min and i_max. For the normalization processing the following function 9 was used:

$$x_{1_{norm}} = 2 * \left( \frac{x_1 - x_{min}}{x_{max} - x_{min}} \right) - 1, \quad x \in R \tag{9}$$

The data is classified in two classes namely: class 0 which contains the normal results and class 1 which comprises of the abnormal ones (DoS, r2l, u2r and probe). The eSNN object using modulation factor m = 0.9, firing threshold ratio c = 0.7 and similarity threshold s = 0.6 in agreement with the vQEA algorithm [19, 28].

**Step 2:** We train the eSNN with 70 % of the dataset vectors (*train_data) a*nd we test the eSNN with 30 % of the dataset vectors *(test_data).*

**Step 3:** If the result of the classification is normal, the eSNN classification process is repeated but this time the relevant normal data vectors are used. These vectors are comprised of 11 features [9]. If the result is normal then the process is terminated. If the result of the classification is abnormal, a two-layer feedforward neural network is used to perform pattern recognition of the attack type with all features of KDD dataset

(41 inputs and 5 outputs). In the hidden layer 33 neurons are used, based on the following empirical function 10 [6]:

$$\left(\frac{2}{3} * \text{Inputs}\right) + \text{Outputs} = \left(\frac{2}{3} * 41\right) + 5 = 33 \tag{10}$$

The outcome of the pattern recognition process is submitted in the form of an *Alert* signal to the network administrator. A Graphical display of the complete HESADM methodology can be seen in Fig. 3.

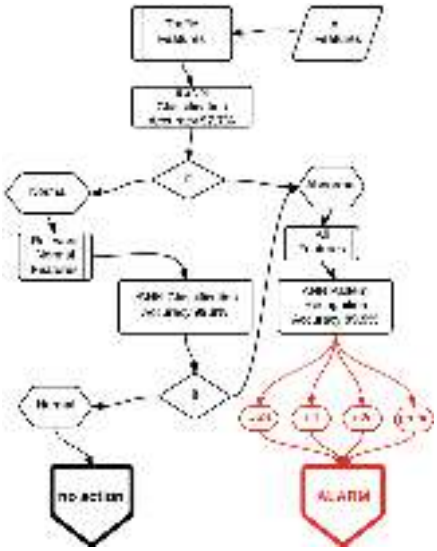


**Fig. 3.** The Hybrid Evolving Spiking Anomaly Detection Model (HESADM) methodology

## 5 Data

The *KDD Cup* 1999 data set [20] was used to test the herein proposed approach. This data set was created in the LincolnLab of MIT [20] and it is the most popular free data set used in evaluation of IDS. It contains recordings of the total network flow of a local network which was installed in the Lincoln Labs and it simulates the military network of the USA air force. The method of events' analysis includes a connection between a source IP address and a destination IP, during which a sequence of TCP packages is exchanged, by using a specific protocol and a strictly defined operation time. The KDD Cup 1999 data includes 41 characteristics which are organized in the following 4 basic categories: *Content Features, Traffic Features, Time-based Traffic Features, Host-based Traffic Features.* Also the attacks are divided in four categories, namely: DoS, r2l, u2r and probe.

## 6 Results

The analysis of the data set, the eSNN classifications and the pattern recognition was performed on a dual boot laptop machine with an AMD Phenom X3 N830 at 2.1 GHz CPU and 4 GB RAM.

In the first classification the data classified as normal or abnormal. The dataset Traf_Red_Full.data has 145,738 records and the 70 % (102,016 rec.) used as train_data and the 30 % (43,722 rec.) used as test_data. The results are shown below:

Classification Accuracy: 97.7 %.
No. of evolved neurons: Class 0/794 neurons - Class 1/809 neurons.
Elapsed time: 2068.23 s.

In order to perform comparison with different learning algorithms the Weka version 3.7 software was used (http://www.cs.waikato.ac.nz/ml/weka). Table 1 reports the results obtained with 10 different classifiers (*NaiveBayes, RBFNetwork, MLP, LibSVM, k-NN, J48, RandomForest, LogisticRegression, BayesNet, AdaBoost*) (Table 2). AQ1

**Table 1.** The Training Accuracy reports the average accuracy computed over 10-fold cross-validation. The testing accuracy refers to the percentage of data that were correctly detected by each classifier in the Traf_Red_Full_Dataset.

| Traf_Red_Full Dataset | | |
|---|---|---|
| Classifier | Train Accuracy | Test Accuracy |
| NaiveBayes | 96.387 % | 95.3981 % |
| RBFNetwork | 94.9734 % | 93.3281 % |
| MLP | 97.9475 % | 97.3743 % |
| LibSVM | 98.9691 % | 97.0335 % |
| k-NN | 97.5435 % | 97.4452 % |
| J48 | 97.619 % | 97.4909 % |
| RandomForest | 97.57 % | 97.5046 % |
| LogisticRegression | 97.8937 % | 96.9008 % |
| BayesNet | 97.9025 % | 96.9237 % |
| AdaBoost | 96.0311 % | 95.947 % |
| eSNN | 98.9 % | 97.7 % |

In the second classification case, the relevant normal features comprising of 11 features were used. The data were classified as normal or abnormal. The dataset normalFull.data has 145,738 records and the 70 % (102,016 rec.) used as train_data and the 30 % (43,722 rec.) used as test_data. The results are shown below:

Classification Accuracy: 99.9 %.
No. of evolved neurons: Class 0/646 neurons - Class 1/136 neurons.
Elapsed time: 1345.25 s.

**Table 2.** The Training Accuracy reports the average accuracy computed over 10-fold cross-validation. The testing accuracy refers to the percentage of data that were correctly detected by each classifier in the normalFull_Dataset.

| normalFull Dataset | | |
|---|---|---|
| Classifier | Train Accuracy | Test Accuracy |
| NaiveBayes | 99.5112 % | 98.895 % |
| RBFNetwork | 99.9351 % | 99.4412 % |
| MLP | 99.9818 % | 99.8992 % |
| LibSVM | 99.673 % | 99.1088 % |
| k-NN | 99.2554 % | 98.9278 % |
| J48 | 99.7751 % | 99.719 % |
| RandomForest | 99.8463 % | 98.9561 % |
| LogisticRegression | 98.998 % | 98.9855 % |
| BayesNet | 98.9933 % | 98.9718 % |
| AdaBoost | 99.2784 % | 98.9357 % |
| eSNN | 99.999 % | 99.9 % |

We can consider the Testing Accuracy as an estimate of the generalization ability of our classifiers. The best results on the testing dataset were obtained by using the eSNN classifier.

A MLFF ANN was developed with 41 input neurons, corresponding to the 41 input parameters of the KDD cup 1999 dataset, 33 neurons in the Hidden Layer and 5 in the output one corresponding to the following output parameters: DoS, r2l, u2r, Probe, normal. The KDD cup 1999 dataset was divided randomly in 70 % (102,016 rec.) the train_data, 15 % (21,861 rec.) as test_data and the rest 15 % (21,861 records) as validation_data. The training process finished in 11 min 54 s and 178 iterations were performed. The performance of the classification is shown in the following matrices and it supports the validity of the model:

**ROC analysis:** The ROC curve is a plot of the true positive rate (sensitivity) versus the false positive rate (1 - specificity) as the threshold is varied. A perfect test would show points in the upper-left corner, with 100 % sensitivity and 100 % specificity. For this problem, the network performs very well (Fig. 4).

**Performance analysis:** Mean Squared Error gives the difference between observation and simulation. The lower the better. In this case all curves converging to the same point mean that network performs perfect (Fig. 5).

**Training State:** The Figure shows variation in gradient coefficient with respect to number of epochs. Minimum the value is better will be training and testing of networks. From figure it can be seen that gradient value goes on decreasing with increase in number of epochs (Fig. 6).

**Error histogram:** this shows how the error sizes are distributed. Typically most errors are near zero, with very few errors far from that (Fig. 7).

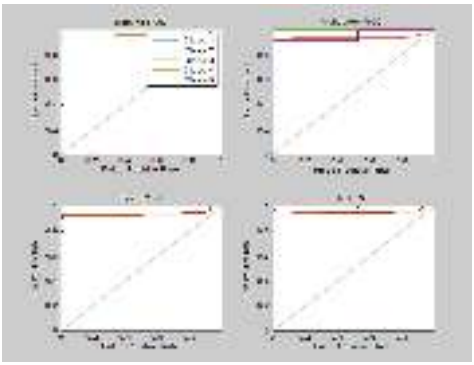

**Fig. 4.** ROC analysis

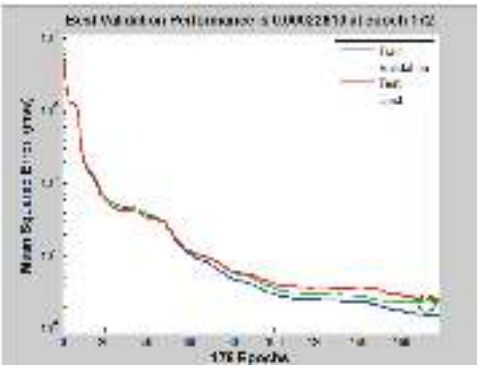

**Fig. 5.** Performance analysis

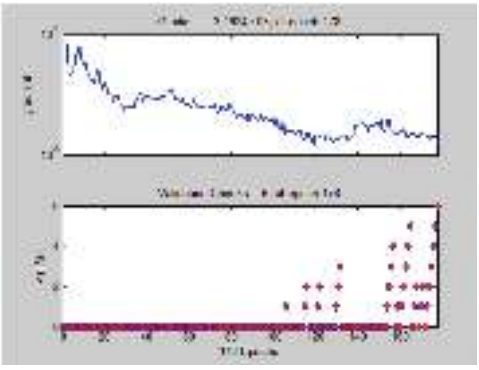

**Fig. 6.** Training State

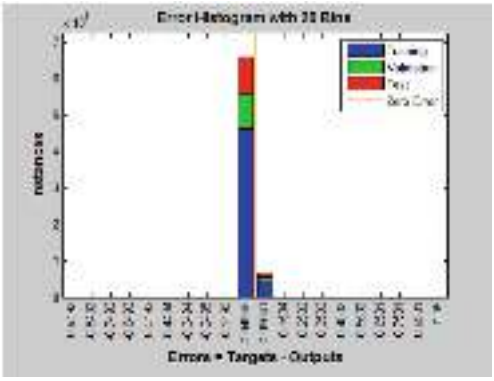

**Fig. 7.** Error histogram

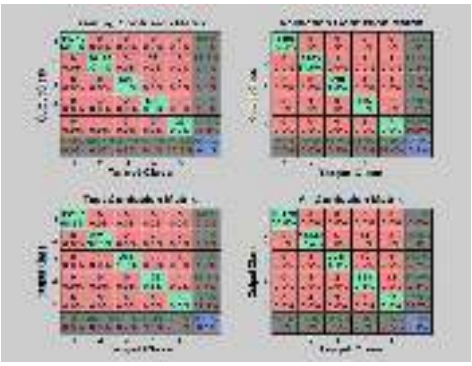

**Fig. 8.** Confusion Matrix

**Confusion Matrix**: The network outputs are very accurate, by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies (Fig. 8).

## 7 Conclusion

In this paper we have proposed a Hybrid Evolving Spiking Anomaly Detection Model which intended to classify the normal and attack patterns in a computer network. This was based on an evolving Spiking Neural Network model and on MLFF ANN techniques. An effort was done to use minimum computational power and resources. The classification performance of eSNN and the accuracy of MLFF ANN were experimentally explored based on KDD cup 1999 dataset. The topology of the eSNN model consists of two layers. The first layer receives an input stimulus obtained from the mapping of a real-valued data sample into spike trains using a rank order population

encoding based on Gaussian receptive fields. As a consequence of this transformation input neurons emit spikes at pre-defined firing times, invoking the one pass learning algorithm. The learning iteratively creates repositories of neurons, one repository for each class. Finally, the output of the second neural layer determines the class label of the presented input stimulus. The eSNN model was investigated in a number of scenarios and reported promising results. Moreover the MLFF ANN system is a pattern recognition system which detects the attacks and classifies them with high accuracy and adds a greater degree of integrity to the rest of security infrastructure of HESADM.

As a future direction, aiming to improve the efficiency of biologically realistic neural networks for pattern recognition, it would be important to extend the eSNN model with ROC analysis. In addition, the model needs to be evaluated further, with respect to parameter optimization in consideration of minimum processing time. Finally, other coding schemes could be explored and compared on the same security task.

## References


1. Dahlia, A., Zainaddin, A., Hanapi, Z.M.: Hybrid of fuzzy clustering neural network over NSL dataset for intrusion detection system. J. Comput. Sci. **9**(3), 391–403 (2013)
2. Delorme, A., Perrinet, L., Thorpe, S.J.: Networks of integrate-and-fire neurons using rank order coding B: spike timing dependant plasticity and emergence of orientation selectivity. Neurocomputing **38–40**(1–4), 539–545 (2000)
3. Denning, E.D.: An Intrusion-Detection model. IEEE Trans. Softw. Eng. **13**, 222–232 (1987). doi:10.1109/TSE.1987.232894
4. Garcıa, P., Verdejo, J., Fernandez, G., Vazquez, E.: Anomaly-based network intrusion detection: techniques, systems & challenges. Comput. Secur. **28**, 18–28 (2009). Elsevier
5. George, H.J.: Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the UAI' 95, pp. 338–345. Morgan Kaufmann Publishers Inc., San Francisco (1995)
6. Heaton, J.: Introduction to Neural Networks with Java (2008). ISBN 097732060X
7. Jakir, H., Rahman, A., Sayeed, S., Samsuddin, K., Rokhani, F.: A modified hybrid fuzzy clustering algorithm for data partitions. Aust. J. Basic Appl. Sci. **5**, 674–681 (2011)
8. Kasabov, N.: Evolving Connectionist Systems: The Knowledge Engineering Approach. Springer, New York (2006)
9. Günes, K.H., Heywood, A.N.Z., Heywood, M.I.: Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets, Natural Sciences and Engineering Research Council of Canada (1999)
10. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: 14th International Joint Conference on Artificial Intelligence, vol. 2, no. 12, pp. 1137–1143 (1995) AQ2
11. Bharti, K., Shweta, J., Sanyam, S.: Fuzzy K-mean clustering via random forest for intrusion detection system. Int. J. Comput. Sci. Eng. **02**(06), 2197–2200 (2010)
12. Mehdi, B., Mohammad, B.: An overview to software architecture in intrusion detection system. Int. J. Soft Comput. Softw. Eng. (2012). doi:10.7321/jscse.v1.n1.1
13. Muna, M., Jawhar, T., Mehrotra, M.: Design network intrusion system using hybrid fuzzy neural network. Int. J. Comput. Sci. Secur. **4**(3), 285–294 (2009) AQ3

14. Mehdi, M., Zulkernine, M.: A neural network based system for intrusion detection and classification of attacks. In: IEEE International Conference on Advances in Intelligent Systems - Theory and Applications (2004)
15. Mukhopadhyay, I.: Implementation of Kalman filter in intrusion detection system. In: Proceeding of International Symposium on Communications and IT, Vientiane (2008)
16. Novikov, D., Yampolskiy, R.V., Reznik, L.: Anomaly detection based intrusion detection. In: Proceedings of the Third International Conference on IT: New Generations, 10–12 April. IEEE (2006)
17. Puketza, N., Zhang, K., Chung, M., Mukherjee, B., Olsson, R.A.: A methodology for testing intrusion detection system. IEEE Trans. Softw. Eng. **22**, 719–729 (1996)
18. Han, S.-J., Cho, S.-B.: Evolutionary neural networks for anomaly detection based on the behavior of a program. IEEE Trans. Syst. Man Cybern. **36**, 559–570 (2005)
19. Schliebs, S., Defoin-Platel, M., Kasabov, N.: Integrated feature and parameter optimization for an evolving spiking neural network. In: 15th International Conference, ICONIP 2008 (2009)
20. Stolfo, S.J., Wei, F., Wenke, L., Prodromidis, A., Chan, P.K.: Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection: results from the JAM project. In: DISCEX '00 (2000)
21. Suguna, J., Selvi, A.M.: Ensemble fuzzy clustering for mixed numeric and categorical data. Int. J. Comput. Appli. **2012**(42), 19–23 (2012). doi:10.5120/5673-7705
22. Tartakovskya, A.G., Rozovskii, B.L., Blazek, R.B., Hongjoong, K.: A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. IEEE **54**(9), 3372–3382 (2006)
23. Zhou, T.-J.: The research of intrusion detection based on genetic neural network. IEEE Xplore Press, Hong Kong, pp. 276–281 (2008). doi:10.1109/ICWAPR.2008.4635789
24. Thorpe, S.J., Delorme, A., van Rullen, R.: Spike-based strategies for rapid processing. Neural Networks **14**(6–7), 715–725 (2001)
25. Thorpe, S.J., Gautrais, J.: Rank order coding. In: CNS '97, pp. 113–118 (1998)
26. Vapnik, V.: The Nature of Statistical Learning Theory, 2nd edn, p. 188. Springer, New York (1995). ISBN 10:0387945598
27. Wei, L., Ghorbani, A.A.: Network anomaly detection based on wavelet analysis. EURASIP **2009**, 1–16 (2009). (Article No. 4, Hindawi Publishing Corp., New York)
28. Wysoski, S.G., Benuskova, L., Kasabov, N.: Adaptive learning procedure for a network of spiking neurons and visual pattern recognition. In: Blanc-Talon, J., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2006. LNCS, vol. 4179, pp. 1133–1142. Springer, Heidelberg (2006)