Pairing an Autoencoder and a SF-SOINN for Implementing an Intrusion Detection System*

Gabriele Voltan, Gian Luca Foresti and Marino Miculan

Department of Mathematics, Computer Science, and Physics
University of Udine
Via delle Scienze 206, Udine, 33100, Italy

Abstract

Intrusion Detection System are systems aiming to detect intrusions within individual computers or networks. These systems are of fundamental importance nowadays, as the number of attacks on networks is ever increasing. In this paper, a prototype of a new Intrusion Detection System is presented. The key novelty is the architecture of this system, pairing an Autoencoder and a Soft-Forgetting Self-Organizing Incremental Neural Network. A fusing scheme is applied to exploit the classification capabilities of the two approaches. The proposed system, tested in different conditions using the NSL-KDD dataset, has achieved excellent performance in detecting attacks, demonstrating its ability to evolve its knowledge and to recognize attacks never seen before.

Keywords

Cybersecurity, intrusion detection systems, anomaly detection, continuous learning, NSL-KDD, Autoencoder, SF-SOINN

1. Introduction

The exponential growth of use of Internet and increasingly complex computer systems has led to an increase in computer crime. Given the presence of information systems now extended to every area, the fight against this phenomenon becomes very important, both for the security and protection of data, and for the safety and health of people who work in contact with industrial machinery controlled by IT systems. To do this, we need to have cyber security systems that can support humans. Intrusion Detection Systems are the right tools for this purpose [1].

There are two ways to classify the various types of IDSs: the first way analyzes the source from which the information comes, classifying the IDS into host-based and network-based; the second way analyzes the intrusion detection technique, classifying the IDS into signature-based and anomaly-based [1]. This paper presents a hybrid anomaly-based IDS (ASFSOINN) that aims to solve the problem of recognizing attacks never seen before.

Nowadays, many IDSs have been created, using various machine learning techniques.

Nirupama and Niranjanamurthy, in [2], achieve an Accuracy higher than 97%, using Decision Tree and Random Forest. This proposed model makes it possible to exploit the advantages of both techniques, also avoiding situations of overfitting.

Serkani, in [3], proposes a model, called DT-LSSVM, which uses Decision Trees to perform a feature selection and a Least-Squares Support-vector machine for anomaly detection. This model, tested on the KDD Cup 99 dataset, reaches an accuracy higher than 98%, on all four classes of attacks (DoS, Probe, R2L, U2R).

Naseer, in [4], proposes a Deep Convolutional Neural Network which obtains excellent results by testing it on the NSL-KDD dataset, demonstrating that these types of neural networks are also suitable for detecting anomalies in computer networks.

In [5], Jianliang shows, by testing the model on the KDDCUP99 dataset, how the k-Means algorithm reaches a detection rate higher than 96% for all four classes of attacks (DoS, Probe, R2L, U2R). The advantage of this model is its unsupervised learning mode. However, this algorithm requires knowing in advance the number of classes k. This makes it inefficient in a real-world scenario where new attacks emerge daily.

In [6], Huang uses the Extending Isolation Forest in combination with the k-Means, obtaining excellent performances on various datasets, including the KDDCUP99. The advantage of this model is the ability to detect anomalies quickly, thanks to the use of binary trees.

Xu, in [7], presents an autoencoder which, tested on the NSL-KDD, achieve an accuracy of 90.61% and an F1-score of 92.26%.

All the models just listed lack the ability to continuously evolve their knowledge. In order to address these issues, in this paper we present a novel system, aiming to integrate continuous learning with anomaly detection.

Ital-IA 2023: 3rd National Conference on Artificial Intelligence, organized by CINI, May 29–31, 2023, Pisa, Italy

*Work partially supported by the Department Strategic Project of the University of Udine within the InterDepartment Project on Artificial Intelligence (2020-25).

voltan.gabriele@spes.uniud.it (G. Voltan); gianluca.foresti@uniud.it (G. L. Foresti); marino.miculan@uniud.it (M. Miculan)

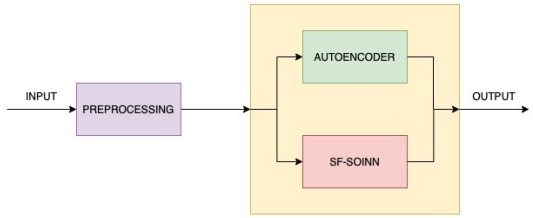

Figure 1: Logical architecture of the proposed ASFSOINN

More precisely, we integrate SF-SOINN with Autoencoder. The SF-SOINN is a neural network, previously presented in [8] capable to implement continuous learning in an efficient way. The Autoencoder is a neural network with excellent ability to detect anomalies. The combination of these two approaches make the system optimal for use in a real, modern scenario.

The rest of the paper is structured as follows. In Section 2 we present ASFSOINN, our solution composed by an Autoencoder and SF-SOINN. In Section 3 we describe some experimental results of the application of ASFSOINN on NSL-KDD dataset. Finally, some conclusions and directions for further work are in Section 4.

2. ASFSOINN: a hybrid anomaly-based IDS

The solution proposed in this paper is a hybrid system, called ASFSOINN, composed of an Autoencoder and a Soft-Forgetting Self-Organizing Incremental Neural Network. The two subsystems work together (Fig. 1), both contributing to the decision of the final label to be assigned to the input data.

2.1. Autoencoder

Autoencoders are a particular type of artificial neural network that aims to reconstruct the input x, after an encoding phase. During this process, the input is compressed by a series of layers, called Encoder, and then decompressed by a series of layers, called Decoder [9]. Fig. 2 shows the logic architecture of the Autoencoder used in ASFSOINN. The architecture of the Autoencoder used in ASFSOINN, unlike many others present in literature, is very simple, to allow the system to work in real time. This feature allows the network to be fast in producing output data, making it usable in real scenarios.

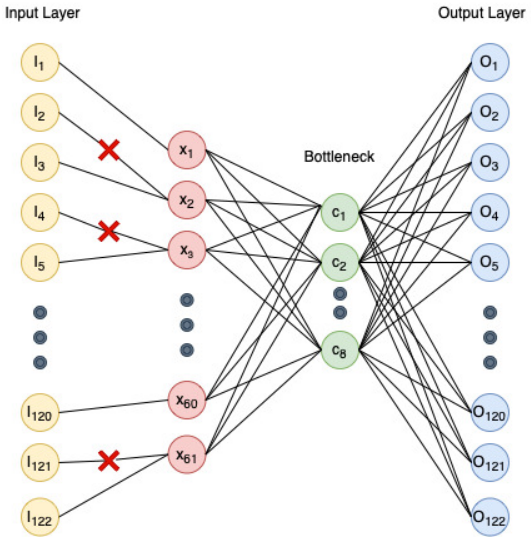

Figure 2: Logical architecture of the Autoencoder used by the ASFSOINN

2.2. Soft-Forgetting Self-Organizing Incremental Neural Network

SF-SOINN (Fig. 3), proposed by Foresti and Martina [8], is an evolution of the model called Enhanced Self-Organizing Incremental Neural Networks [10]. The main features of this neural networks are the capability to evolve its knowledge (continuous learning) and the speed in producing the output, possible as the model forgets what is no longer relevant [8].

This model takes as input a given x, with the respective label y, and using a distance function, calculates the two nodes closest to the one received as input. Next, it calculates a threshold to decide whether or not to add a node for the received data. After this phase, the model goes through an update phase, in which it eliminates any nodes and edges that are no longer considered useful.

How the SF-SOINN model works is explained in detail in [8].

2.3. Operating modes

The ASFSOINN proposed, has three operating modes: training mode, necessary to train the AE and SF-SOINN; testing mode, which allows the system to classify the input data; live mode, which allows an external operator to manual classify a specific data.

The algorithms are shown below.

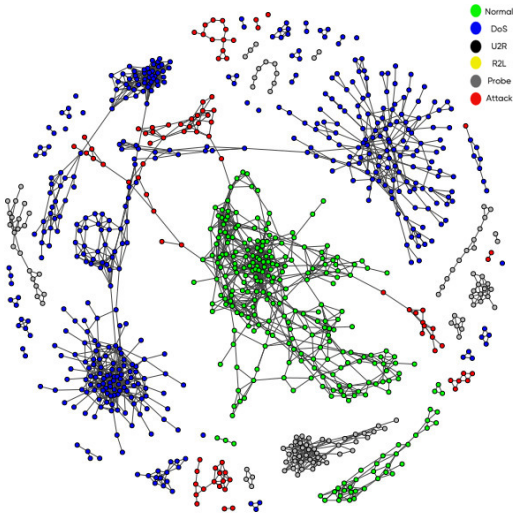

Figure 3: An example of the SF-SOINN network used by the ASFSOINN

2.3.1. Training mode

This phase must be performed before all others, as it allows the system to learn. The training of this system is particular, as it requires two different training sets:

- AE training set: composed only of good data (no attacks);
- SF-SOINN training set: composed by both attacks and good data.

The autoencoder only needs to be trained with good data as its operation is based on learning the normal behavior, then labeling anything that is different as a possibile "attack".

Algorithm 1 Training phase

Require: $sfsoinn, data\_ae, data\_sf, y\_sf$
  $ae \leftarrow build\_model$ ▷ Creation of AE
  $ae.fit(data\_ae)$ ▷ AE training
  for each $x, y \in data\_sf, y\_sf$ **do**
    $sfsoinn.input(x, y)$ ▷ Input of the data x with label y
  end for

2.3.2. Testing mode

The testing mode is used for classifying the data, according to the model obtained by training the system.

As we can see from Algorithm 2, comparisons between the two predictions are made to take the final decision. If the two subsystems agree on the label, that prediction is assigned to the input. If the two subsystems predict two different labels, priority is given to the subsystems that predicts an attack.

Algorithm 2 Test phase

Require: $ae, sfsoinn, data$
  for each $x \in data$ **do**
    $y\_ae \leftarrow ae.predict(x)$ ▷ AE prediction
    $y\_sf \leftarrow sfsoinn.input(x)$ ▷ SF-SOINN prediction
    if $y\_ae = \text{“}attack\text{”} \wedge y\_sf \neq \text{“}normal\text{”}$ **then**
      $y \leftarrow y\_sf$
    else if $y\_ae = \text{“}normal\text{”} \wedge y\_sf \neq \text{“}normal\text{”}$ **then**
      $y \leftarrow y\_sf$
    else if $y\_ae = \text{“}attack\text{”} \wedge y\_sf = \text{“}normal\text{”}$ **then**
      $y \leftarrow \text{“}attack\text{”}$
    else
      $y \leftarrow \text{“}normal\text{”}$
    end if
  end for

For the latter case, we distinguish two subcases: the first is the one in which the attack is predicted by the SF-SOINN and the second is the one in which the attack is predicted by the autoencoder. In the first subcase, the assigned label is the attack name (eg: "nmap", "smurf", "guess_passwd", etc..). In the second sub-case instead, since the predictions made by the AE are "normal" or "attack", the label "attack" is assigned. This generic label can later be replaced with a specific attack label by an expert domain operator, taking advantage of the live operating mode.

2.3.3. Live mode

This mode is very useful as it allows cybersecurity experts to transmit their knowledge to the IDS. Live mode allows an external operator to manually label data and use it to train the system.

Algorithm 3 Live phase

Require: $sfsoinn, data, y\_data$
  for each $x, y \in data, y\_data$ **do**
    $sfsoinn.input(x, y)$ ▷ Data x with label y
  end for

As we can see from the code, in this mode the AE is not involved, as the only thing to do is to update the SF-SOINN.

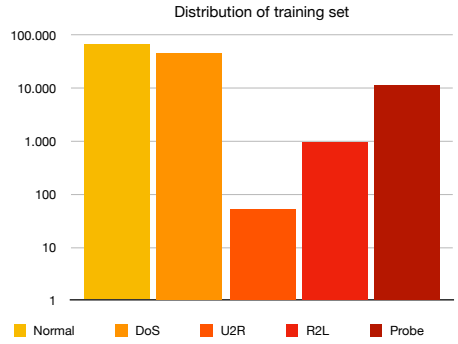

Figure 4: Training set distribution (log scale)

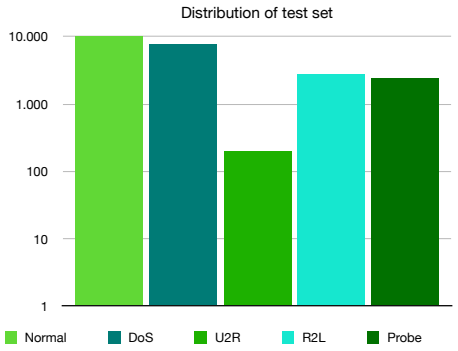

Figure 5: Test set distribution (log scale)

3. Experiments and Results

The dataset chosen to conduct the experiments with ASFSOINN is the NSL-KDD [11]. It contains five data classes, one normal and four attacks (DoS, U2R, R2L, Probe).

It is composed of 160367 elements, divided into a train set of 125973 elements, a test set of 22544 and a second test set, which we called test_n21, of 11850 elements. In addition to these two test sets, we created another one, which we called test_21, which contains only the test set items that have a difficulty of 21, i.e. 10694.

Fig. 4 shows the distributions of the training set, while Fig. 5 shows the distribution of the test set.

3.1. Metrics

This subsection lists the metrics used to evaluate the performance of ASFSOINN system in the various experiments conducted. The metrics used are:

Accuracy: this metric measures how many correct classifications were made out of the total number of predictions made. The formula is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Test Set	*Accuracy*	*Detection Rate*	*FP Rate*	*FN Rate*
test	90.40	96.67	17.89	3.33
test_n21	84.86	95.60	63.52	4.40
test_21	97.16	100.00	4.02	0

Table 1
ASFSOINN test result on the three test sets

where TP is the number of true positive predictions, TN is the number of true negative predictions, FP is the number of false negative predictions, and FP is the number of false positive predictions.

Detection Rate: this metric measures how many correct positive classifications were made out of the total number of positive cases. The formula is:

$$DR = \frac{TP}{TP + FN}$$

where TP and FN are the same as described above.

False Positive Rate: this metric measures the probability that a negative case will be classified as a positive. The formula is:

$$FPR = \frac{FP}{FP + TN}$$

where FP and TN are the same as described above.

False Negative Rate: this metric measures the likelihood that a positive case will go undetected. The formula is:

$$FNR = \frac{FN}{FN + TP}$$

where FN and TP are the same as described above.

3.2. Training with the NSL-KDD

In the first test performed, ASFSOINN is trained using the training set. Specifically, the AE is trained with the good data of the training set (67343 elements), while the SF-SOINN with the whole set (125973 elements). After training, ASFSOINN is tested using the test set, test_n21 and test_21, i.e. the one containing the items with difficulty equal to 21. Unlike the test set, the test_n21 contains many attacks that are not in the training set. For this reason, test_n21 is considered the hardest test set.

Below we show the table of the results obtained by the IDS in the testing phase (Table 1).

As we can see from the results shown in the Table 1, the system has an excellent ability to recognize attacks. Furthermore, it was able to obtain acceptable accuracy even in those more difficult test cases, as in the case of test_n21.

Test Set	*Accuracy*	*Detection Rate*	*FN Rate*
Train 1	90.80	98.07	1.93
Train 2	91.22	97.96	2.04
Train 3	91.06	97.94	2.06
Train 4	90.71	97.80	2.02
Train 5	90.95	97.56	2.44
Train 6	91.03	97.76	2.24
Train 7	91.51	98.36	1.64
Train 8	91.10	97.96	2.04
Train 9	91.32	98.02	1.98
Train 10	91.15	97.88	2.12
Train 11	91.47	98.08	1.92
Train 12	90.50	97.96	2.01
Train 13	91.50	97.82	2.18
Media	91.10	97.94	2.05

Table 2
ASFSOINN test result on split training set

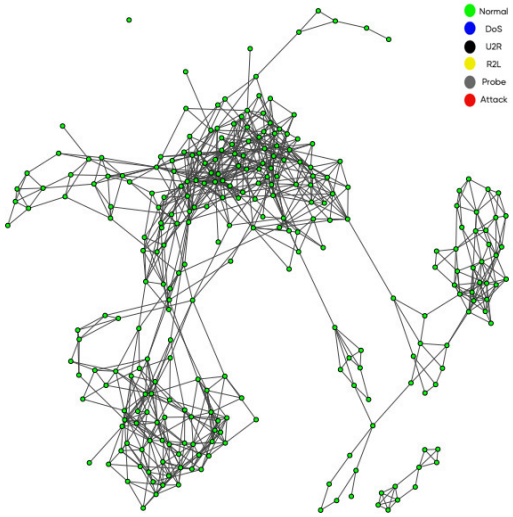

Figure 6: SF-SOINN after training with train_normal

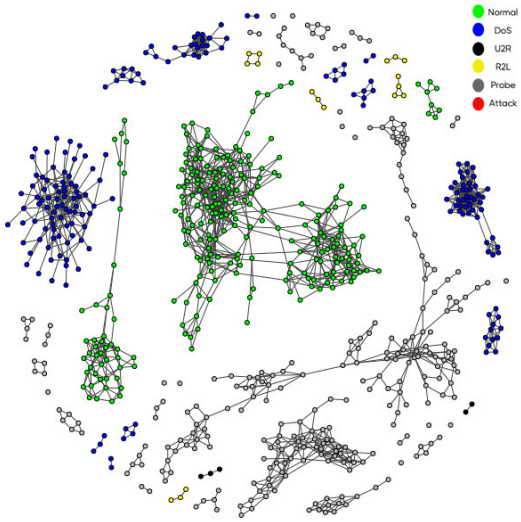

Figure 7: SF-SOINN after training with all train set

3.3. Training set vs Test set

The second test performed is a test proposed by Constantinides [12]. The peculiarity of this test consists in the fact that the system is trained with the *test set* and tested with the *train set*, dividing it into 12 sets of 10000 elements and a set with the remaining elements.

From the results shown in the Table 2 we can see how this system, despite the particularity of the experiment, still obtains excellent results, both in terms of Detection Rate and in terms of Accuracy.

3.4. Incremental Training

In this experiment, proposed by Li-Ye [13], the system is trained incrementally. The train set is divided into 5 sets containing all the data of that category (*train_normal, train_dos, train_r2l, train_u2r, train_probe*), which will be passed to the system one at a time.

After each incremental training phase, the system is tested on each class, using the *test set* divided into 5 classes. The original experiment starts by training the system with train_normal and train_dos, then dividing the experiment into four phases. In this work instead, the experiment is divided into five phases, training the system in the first phase only with the train_normal. This choice was made to show how the proposed system can detect attacks even without having seen one before.

After the first training phase, the SF-SOINN configuration is the one shown in Fig. 6. In Fig. 7 instead, we can observe the state of the model after training on all categories of attacks. As we can see from these two images, the configuration of the SF-SOINN model has evolved. This highlights the continuous learning of this model.

This experiment is the most important as it simulates the worst scenario that could happen, i.e. the one in which the IDS receives as input attacks it does not know. This type of test allows us to highlight the capabilities of this innovative IDS, which is able to recognize attacks never seen before, thanks to the anomaly detection work carried out by the AE. The results of this experiment is shown in Table 3. As we can see from the results, ASFSOINN recognizes a high number of attacks even after being trained with only good data (no attacks).

Round	*Training*	*Detection Rate Normal*	*Detection Rate DoS*	*Detection Rate R2L*	*Detection Rate U2R*	*Detection Rate Probe*
1	*normal*	83.02	94.11	99.22	83.58	99.36
2	*dos*	85.38	91.72	87.15	97.01	99.34
3	*r2l*	82.34	93.75	98.82	97.01	100.00
4	*u2r*	83.17	94.71	98.49	97.01	99.59
5	*probe*	86.70	92.46	85.31	88.06	99.67

Table 3
Incremental ASFSOINN test result on category test sets

4. Conclusions and Future works

In conclusion, the importance of cyber security research cannot be overstated in achieving collective security. With the rise of new types of attacks, it is crucial to have a reliable machine learning model that can detect anomalies.

The SF-SOINN model used in this study is a neural network designed for efficient continuous learning. It is capable of learning and adapting to new data, making it well-suited for detecting anomalies in network traffic. On the other hand, the Autoencoder is a neural network that has excellent ability to detect anomalies by compressing and reconstructing input data. By combining the two models, the proposed system in this study is able to detect new attacks and evolve its knowledge, making it a promising approach for intrusion detection.

The NSL-KDD dataset used in the experiments is a widely-used dataset for evaluating intrusion detection systems. It contains various types of network traffic, including both normal and attack traffic, and is designed to simulate real-world network traffic scenarios. By using this dataset, in this study we have been able to demonstrate the effectiveness of the proposed system in detecting and learning new types of attacks.

It is important to note that while the proposed system achieved high detection rates, it also had a slightly high False Positive Rate. This means that the system may identify normal network traffic as an attack. To address this issue, future work could focus on developing a solution that combines the results produced by both subsystems, improving the overall performance of the intrusion detection system.

Overall, the proposed system in this study shows great potential for detecting and learning new types of attacks in network traffic, making it a promising approach for improving cyber security.

Acknowledgments

This work was partially supported by the Department Strategic Project of the University of Udine within the InterDepartment Project on Artificial Intelligence (2020-25). We thank Matteo Paier for proof reading the early version of this paper.

Bibliography

[1] R. Bace, P. Mell, Intrusion detection system (2001).
[2] N. B. K., M. Niranjanamurthy, Network intrusion detection using decision tree and random forest (2022).
[3] E. Serkani, H. Gharaee, N. Mohammadzadeh, Anomaly detection using svm as classifier and decision tree for optimizing feature vectors, The ISC Int'l Journal of Information Security Volume 11 (2019).
[4] S. Naseer, Y. Saleem, S. Khalid, M. K. B. J. H. M. M. I. K. Han, Enhanced network anomaly detection based on deep neural networks (2018).
[5] M. Jianliang, S. Haikun, B. Ling, The application on intrusion detection based on k-means cluster algorithm, International Forum on Information Technology and Applications (2009).
[6] M. T. R. Laskar, J. X. Huang, V. Smetana, C. Stewart, K. Pouw, A. An, S. Chan, L. Liu, Extending isolation forest for anomaly detection in big data via k-means, ACM Transactions on Cyber-Physical Systems 5 (2021).
[7] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, F. Sabrina, Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset (2021).
[8] M. Rinaldo Martina, G. L. Foresti, A continuous learning approach for real-time network intrusion detection, Internation Journal of Neural Systems (2021) 22.
[9] J. An, S. Cho, Variational autoencoder based anomaly detection using reconstruction probability (2015).
[10] T. O. S. Furao, O. Hasegawa, An enhanced self-organizing incremental neural network for online unsupervised learning (2007).
[11] U. of New Brunswick, Nsl-kdd dataset, ????
[12] B. G. C. Constantinides, S. Shiaeles, N. Kolokotronis, A novel online incremental learn- ing intrusion prevention system, 10th IFIP Int. Conf. New Technologies, Mobility and Security (NTMS) (2019).
[13] L.-Y. Tian, W.-P. Liu, Incremental intrusion detecting method based on som/rbf, Int. Conf. Machine Learning and Cybernetics, ICMLC (2010).