

# A Two-Step Network Intrusion Detection System for Multi-Class Classification

(Discussion Paper)

Giuseppina Andresini<sup>1</sup>, Annalisa Appice<sup>1,2</sup> and Donato Malerba<sup>1,2</sup>

<sup>1</sup>Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro via Orabona, 4 - 70126 Bari - Italy

<sup>2</sup>Consorzio Interuniversitario Nazionale per l'Informatica - CINI

## Abstract

A network intrusion detection system aims to discover any unauthorised access to computer networks by analysing the network traffic for signs of malicious activity. In this paper, we present a two-step system for network intrusion detection. The first step comprises a Triplet network that processes the flow-based characteristics of the historical network traffic data to learn an embedding space, where distances between samples labelled with opposite classes are greater than distances between samples labelled with the same class. We take advantage of this embedding space to separate the normal samples from the malicious ones. The second step uses a multi-class eXtreme Gradient Boosting classifier to recognize the attack family of the detected malicious flows. The experiments prove the effectiveness of the proposed system as it leads to higher accuracy when compared to several, recent competitors.

## Keywords

Network Intrusion Detection, Deep metric learning, Triplet network, Autoencoder

## 1. Introduction

Network intrusion detection is a crucial cyber-security problem, where deep learning is recognised as a relevant approach to learn signs of malicious activity [1, 2, 3, 4, 5]. In this study, we illustrate a two-step network intrusion detection system that first recognises signs of malicious activities in the network traffic and then classifies the malicious type of these activities.

In the first step, we resort to deep learning to analyse the flow-based characteristics of the network traffic data. In particular, we resort to an intrusion detection model that is trained through a Triplet network [6]. This is an emerging deep learning network that combines deep learning and metric learning. A Triplet network is commonly trained taking triplet samples as input. Every triplet is commonly composed of a training sample selected as an anchor, a training sample labelled in the same class as the anchor and a training sample labelled in the opposite class. Then, the Triplet network learns an embedding space, where distances between samples labelled with opposite classes are greater than distances between samples labelled with

---

SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy

✉ giuseppina.andresini@uniba.it (G. Andresini); annalisa.appice@uniba.it (A. Appice); donato.malerba@uniba.it (D. Malerba)

🆔 0000-0002-5272-644X (G. Andresini); 0000-0001-9840-844X (A. Appice); 0000-0001-8432-4608 (D. Malerba)

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

the same class. Although, the Triplet network is emerging as a relevant approach to separate samples belonging to opposite classes, existing approaches based on Triplet networks commonly suffer from poor convergence. In particular, the Triplet network convergence problem is mainly caused by a random selection of samples for the triplet construction in the training set [7]. In fact, the traditional Triplet network implementations randomly select a single training sample labelled in the same class as the anchor and a single training sample labelled in the opposite class.

To overcome the convergence affecting traditional Triplet networks, we resort to an innovative triplet construction strategy. The original contribution of this work is reported in [8]. This strategy uses autoencoders [9] instead of traditional sampling to derive both the positive and negative information for the triplet construction. Specifically, we train two separate autoencoders on historical normal network flows and attacks, respectively. We construct every triplet by considering the positive and negative pseudo-samples that are the unique reconstructions of the considered anchor restored through the two autoencoders. We note that, basing the triplet construction, and consequently the triplet-learned embedding, on the pseudo-samples reconstructed through these two autoencoders, we are able to exploit possible patterns existing among the normal and attack classes, given the class of the anchor sample of the triplet.

As the learned embedding moves each flow close to its reconstruction, restored with the autoencoder associated with the same class as the flow, and away from its reconstruction, restored with the autoencoder of the opposite class, we are able to perform a predictive stage assigning each new flow to the binary class (normal or attack) associated with the autoencoder that restores the closest reconstruction of the flow in the embedding space.

In the second step, we use a multi-class classifier to perform the task of attack type classification, in addition to attack detection. To this aim, we use a multi-class eXtreme Gradient Boosting classifier trained on attack samples only, in order to recognize the attack family of the detected malicious flows.

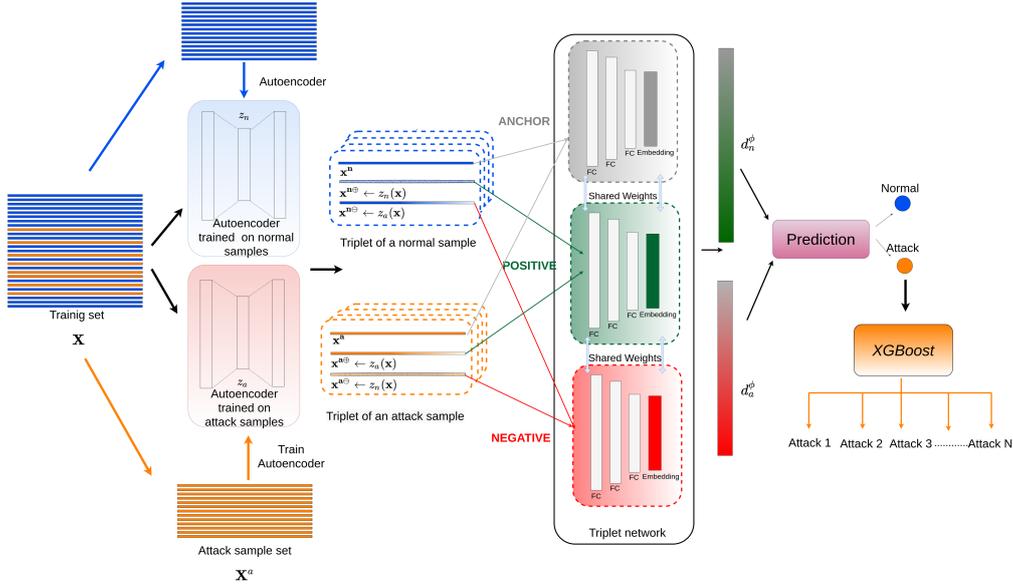
This paper is organised as follows. The formulated two-step network intrusion detection system is described in Section 2, while the implementation details are illustrated in Section 3. The NSL-KDD dataset considered for the empirical validation and the findings in the evaluation of the proposed network intrusion detection system are discussed in Section 4. Finally, Section 5 refocuses on the purpose of the research and draws conclusions.

## 2. The two-step system

The detailed description of the training and predictive stage are reported in Sections 2.1 and 2.2, respectively. The block diagram of the proposed system is reported in Figure 1

### 2.1. Training stage

Let us consider  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$  as a set of  $N$  training samples, where each  $\mathbf{x}_i \in \mathbb{R}^D$  is a row vector corresponding to an input network flow defined over  $D$  features, and  $y_i$  is the corresponding binary label denoting a *normal* or an *attack* sample.



**Figure 1:** Architecture of the two-step system (Triplet network + XGBoost).

In the first step, we train a Triplet network using autoencoders to built triplets. Let  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T \in \mathbb{R}^{N \times D}$  denote the data matrix of  $N$   $D$ -dimensional random variables  $\mathbf{x} \in \mathbb{R}^D$ , so that  $\mathbf{X}^n = \mathbf{X}_{|y_i=normal}$ , resp.  $\mathbf{X}^a = \mathbf{X}_{|y_i=attack}$ , represent the subset of samples in  $\mathbf{X}$ , whose label is normal, resp. attack. We process  $\mathbf{X}^n$  and  $\mathbf{X}^a$  separately to learn two independent autoencoders, denoted as normal autoencoder  $z_n$  and attack autoencoder  $z_a$ , respectively. We use both autoencoders to construct the two triplet counterparts of each anchor.

In principle, the normal autoencoder  $z_n$  should aid in recovering a denoised representation of  $\mathbf{X}$ , which highlights attacks as anomalies. Based upon this consideration, normal flows can be considered as positive samples, while attacks can be seen as negative samples from the normal autoencoder  $z_n$ . Vice-versa, the attack autoencoder  $z_a$  aids in recovering a denoised representation of  $\mathbf{X}$ , which highlights normal flows as anomalies. Therefore, the autoencoder  $z_a$  can see normal flows as negative samples and attacks as positive samples from its point of view. This conjecture inspires the idea of using the representations of  $\mathbf{X}$  reconstructed through both  $z_n$  and  $z_a$  to derive the unique positive and negative component of each triplet. Following this idea, let us consider a training sample  $\mathbf{x}$  assigned to the label  $y = normal$  as the anchor. We build  $\mathbf{x}^{\oplus} = z_n(\mathbf{x})$  as the positive triplet counterpart of  $\mathbf{x}$ , and  $\mathbf{x}^{\ominus} = z_a(\mathbf{x})$  as the negative triplet counterpart of  $\mathbf{x}$ , respectively. Vice-versa, let us consider a training sample  $\mathbf{x}$  assigned to the label  $y = attack$  as the anchor. We build  $\mathbf{x}^{\oplus} = z_a(\mathbf{x})$  as the positive triplet counterpart of  $\mathbf{x}$ , and  $\mathbf{x}^{\ominus} = z_n(\mathbf{x})$  as the negative triplet counterpart of  $\mathbf{x}$ , respectively.

By leveraging the collection of sample triplets constructed from  $\mathbf{X}$ , we train a Triplet network that embeds every triplet component into a  $d$ -dimensional Euclidean space  $\phi: \mathbb{R}^D \mapsto \mathbb{R}^d$  learned to better quantify the similarity between sample components within each triplet. This

Triplet network processes  $\mathbf{x}$ ,  $\mathbf{x}^\oplus$  and  $\mathbf{x}^\ominus$  across three base feed-forward networks with shared parameters. The network learns the embedding  $\phi: \mathbb{R}^D \mapsto \mathbb{R}^d$  by optimising a triplet loss function. This loss function minimises the distance between the embedding vectors of both the anchor  $\mathbf{x}$  and its positive triplet counterpart  $\mathbf{x}^\oplus$ , while it maximises the distance between the anchor  $\mathbf{x}$  and its negative triplet counterpart  $\mathbf{x}^\ominus$ . We compute the soft-margin triplet loss proposed in [10].

In the second step, we train a multi-class eXtreme Gradient Boosting (XGBoost) classifier to classify the attacks into specific categories. This type of disentanglement into different attack types is important for the network administrator who can take appropriate responsive steps depending on the type of the intrusion detected. Based upon the analysis conducted in [11], we select XGBoost as an appropriate multi-class classifier for this scope. It is trained on the feature space  $\mathbf{X}$  on the attack samples labelled with the specific attack category.

## 2.2. Testing stage

Let us consider a query sample  $\mathbf{x} \in \mathbb{R}^D$ . First the sample reconstructions  $z_n(\mathbf{x})$  and  $z_a(\mathbf{x})$  are restored through both the normal autoencoder  $z_n$  and the attack autoencoder  $z_a$ , respectively. Then the Euclidean distance is computed to compare  $\mathbf{x}$  to both  $z_n(\mathbf{x})$  and  $z_a(\mathbf{x})$ , respectively. The Euclidean distance is computed within the embedding space  $\phi$ , so that:

$$d_n^\phi(\mathbf{x}) = \|\phi(\mathbf{x}) - \phi(z_n(\mathbf{x}))\|^2, \quad (1)$$

$$d_a^\phi(\mathbf{x}) = \|\phi(\mathbf{x}) - \phi(z_a(\mathbf{x}))\|^2. \quad (2)$$

If  $d_a^\phi(\mathbf{x}) < d_n^\phi(\mathbf{x})$ , then  $\mathbf{x}$  is classified as an attack, while the attack type is predicted with the classification model trained with XGBoost. Otherwise  $\mathbf{x}$  is classified as a normal network flow. Finally, if  $\mathbf{x}$  is classified as an attack.

## 3. Implementation details

The triplet network is implemented in Python 2.7. The source code is available online.<sup>1</sup> The deep neural network architectures are developed in Keras 2.3<sup>2</sup> – a high-level neural network API with TensorFlow<sup>3</sup> as the back-end. The pre-processing step includes the operation to scale the input numeric features, using the Min-Max scale (as implemented in the Scikit-learn 0.22.2 library<sup>4</sup>). In addition, the pre-processing includes the implementation of the one-hot-encoder mapping of the categoric attributes. We conduct an automatic hyper-parameter optimization using the tree-structured Parzen estimator algorithm, as implemented in the Hyperopt library [12]. This hyper-parameter optimization is performed by using 20% of the entire training as a validation set. In particular, we randomly select the validation set with the stratified sampling procedure. We automatically choose the configuration of the parameters, which achieves the

---

<sup>1</sup><https://github.com/gsndr/RENOIR>

<sup>2</sup><https://keras.io/>

<sup>3</sup><https://www.tensorflow.org/>

<sup>4</sup><https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>

**Table 1**  
Hyper-parameter search space.

	Autoencoders	Triplet network
batch size	$\{2^5, 2^6, 2^7, 2^8, 2^9\}$	$\{2^5, 2^6, 2^7, 2^8, 2^9\}$
learning rate	[0.0001, 0.01]	[0.0001, 0.01]
dropout	[0,1]	[0,1]
#neurons for hidden layer	-	$\{2^7, 2^8, 2^9\}$

best validation loss. The values of the hyper-parameters, automatically explored with the tree-structured Parzen estimator, are reported in Table 1.

Each autoencoder architecture comprises 3 fully-connected (FC) layers of  $32 \times 16 \times 32$  neurons and one dropout layer, in order to prevent the overfitting phenomenon. The mean squared error (*mse*) is used as the loss function. The classical rectified linear unit (*ReLU*) is selected as the activation function for each hidden layer, while for the last layer the *Linear* activation function is used. The Triplet network is implemented with three base feed-forward networks with shared weights. Each base network is a deep neural network with three intermediate layers (with the number of neurons chosen with the hyper-parameter optimization), an embedding layer with 512 neurons and two dropout layers. The (*ReLU*) function is selected as the activation function for each hidden layer, while for the embedding layer the *Sigmoid* activation function is used. The *Sigmoid* activation function is commonly used for the embedding layer [13] instead of a *Linear*, in order to guarantee that each dimension will be between 0 and 1. This architecture assigns samples to the binary normal or attack classes by returning the Euclidean distance returned by the embedding layers.

The multi-class classifier XGBoost is integrated from `xgboost.XGBoostClassifier`,<sup>5</sup> by adopting the default parameter set-up reported in the documentation.<sup>6</sup>

## 4. Empirical evaluation

We analyse the effectiveness of the proposed methodology also in a multi-class scenario of network intrusion detection. To this purpose, we consider the multi-class version of the dataset NSL-KDD.<sup>7</sup> This dataset is introduced in [14] as a revised version of KDDCUP99, which is obtained by removing the duplicate samples from KDDCUP99. The multi-class NSL-KDD comprises normal flows and four categories of attack: Denial of Service Attack (DoS), User to Root Attack (U2R), Remote to Local Attack (R2L) and Probing Attack. For this experiment we adopt the original data setting described in [14], that includes KDDTrain<sup>+</sup> as the training set and KDDTest<sup>+</sup> as the testing set. The number of samples collected in both the training and testing sets for each category is reported in Table 2. We note that both U2R and R2L are rare attacks. We select this dataset as it include zero-days attacks in the testing set, and despite it is an old dataset, it has been recently used in the evaluation of various multi-class intrusion detection methods. Table 3 reports Precision, Recall and F-score achieved by the proposed method on each class.

<sup>5</sup><https://xgboost.readthedocs.io/en/latest/index.html>

<sup>6</sup><https://xgboost.readthedocs.io/en/latest/parameter.html>

<sup>7</sup><https://www.unb.ca/cic/datasets/nsl.html>

**Table 2**

Number of samples per category in both KDDTrain<sup>+</sup> and KDDTest<sup>+</sup>.

Dataset	Total	DoS	Probe	R2L	U2R	Normal
<b>KDDTrain<sup>+</sup></b>	125973	45927 (37%)	11656 (9.11%)	995 (0.85%)	52 (0.04%)	6734 (53%)
<b>KDDTest<sup>+</sup></b>	22544	7458 (33%)	2421 (11%)	2754 (12.1%)	200 (0.9%)	9711 (43%)

**Table 3**

Triplet network + XGBoost: Precision, Recall and F-score per class in NLS-KDD

Class	Precision	Recall	F-score
<b>DoS</b>	96.1	82.5	88.8
<b>Probe</b>	76.4	85.8	80.8
<b>R2L</b>	95.2	48.0	63.8
<b>U2R</b>	62.5	7.5	13.4
<b>Normal</b>	77.8	96.3	86.1

For the comparative study, we consider the performance of several recent competitors that handle the same multi-class problem addressed here, by integrating various techniques to deal with multi-class attacks. In particular, we consider: (1) Deep learning multi-class competitors with Siamese networks [15]<sup>8</sup> and XGBoost: SIAM-IDS[16][11] and I-SiamIDS [11]. (2) Deep learning multi-class competitors with rare data augmentation: DSSTE+AlexNet[17], IGAN-IDS[18] and ID-CAVE[19]. (3) Deep Reinforcement Learning competitors: AESMOTE[20] and AE-RL[21]. The results of Macro-Average F1, Micro-Average F1 and Weighted F1 of both the multi-class configurations of Triplet network + XGBoost and its competitors are reported in Table 4. No values of the Macro-Average F1 are reported in the reference papers for DSSTE+AlexNet, IGAN-IDS, AESMOTE and AE-RL.

We note that SIAM-IDS and I-SiamIDS are the nearest-related competitors to our method. This is an expected outcome, as both algorithms use a deep learning methodology based on a deep metric architecture for the binary classification and resort to a multi-class XGBoost-based re-classification of the attacks. So, this experiment contributes to assess the effectiveness of our Triplet network-based method in separating normal samples from attacks. In addition, it highlights that resorting to XGBoost for the attack classification can achieve good performance. On the other hand, our method also outperforms the remaining competitors, that have been defined in the recent literature to deal with the multi-class problem in the imbalanced scenario of NSL-KDD. The only exception is IGAN-IDS, that uses a deep Generative Adversarial Network process to generate new samples for the minority classes. Therefore, this improvement is achieved at the cost of a complex process of augmentation of the training set. In any case, this result suggests that new milestones may be reached in the future by injecting Generative Adversarial Learning mechanisms into multi-class network intrusion detection.

<sup>8</sup>Siamese networks learn a contrastive loss to quantify similarity between sample pairs.

**Table 4**

Results of the Macro-average F1, the Micro-Average (OA) and the weighted F1 obtained by our method compared with several competitors that perform multi-class classification in NSL-KDD. The best results are in bold. The results of the competitors are collected from the reference papers. "-" denotes that no value is reported in the reference paper.

Algorithm	Macro-Average F1	Micro-Average (OA)	Weighted F1
Triplet+XGBoost	<b>66.58</b>	83.91	83.05
SIAM-IDS[16][11]	56.14	76.96	75.28
I-SiamIDS[11]	66.54	79.90	78.49
DSSTE+AlexNet[17]	-	82.84	81.66
IGAN-IDS[18]	-	<b>84.45</b>	<b>84.17</b>
AESMOTE[20]	-	82.09	82.43
AE-RL[21]	-	80.16	79.40
ID-CAVE[19]	59.27	80.10	79.08

## 5. Conclusion

In this paper we present an innovative Triplet network approach that takes advantage of autoencoder information for effective network intrusion detection. In addition, the attack samples identified with the first step through the Triplet network are input to the second step composed of the multiclass eXtreme Gradient Boosting for the classification of the detected malicious activity into attack categories. In general, this study provides the evidence that the Triplet network, originally formulated for a binary task, may also be combined with multi-class attack classifiers to have potential in the multi-class scenario. In any case, this conclusion paves the way for a systematic investigation of the topic in the future.

## 6. Acknowledgments

We acknowledge the support of the MIUR-Ministero dell'Istruzione dell'Università e della Ricerca through the project "TALIsMan -Tecnologie di Assistenza personalizzata per il Miglioramento della qualità della vita" (Grant ID: ARS01\_01116), PON RI 2014-2020 funding scheme, as well as the project "Modelli e tecniche di data science per la analisi di dati strutturati", funded by the University of Bari "Aldo Moro".

## References

- [1] D. S. Berman, A. L. Buczak, J. S. Chavis, C. L. Corbett, A survey of deep learning methods for cyber security, *Information* 10 (2019) 1–35.
- [2] G. Andresini, A. Appice, N. D. Mauro, C. Loglisci, D. Malerba, Multi-channel deep feature learning for intrusion detection, *IEEE Access* 8 (2020) 53346–53359.
- [3] G. Andresini, A. Appice, F. Caforio, D. Malerba, Improving cyber-threat detection by moving the boundary around the normal samples, in: Y. Maleh, Y. Baddi, M. Shojaafar, M. Alaza

- (Eds.), *Machine Intelligence and Big Data Analytics For Cybersecurity Applications*, Studies in Computational Intelligence, 2021, pp. 105–127.
- [4] G. Andresini, A. Appice, D. Malerba, Nearest cluster-based intrusion detection through convolutional neural networks, *Knowledge-Based Systems* 216 (2021) 106798.
  - [5] Gan augmentation to deal with imbalance in imaging-based intrusion detection, *Future Generation Computer Systems* 123 (2021) 108–127.
  - [6] E. Hoffer, N. Ailon, Deep metric learning using triplet network, in: A. Feragen, M. Pelillo, M. Loog (Eds.), *Similarity-Based Pattern Recognition*, Springer International Publishing, Cham, 2015, pp. 84–92.
  - [7] B. Yu, T. Liu, M. Gong, C. Ding, D. Tao, Correcting the triplet selection bias for triplet loss, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), *Computer Vision – ECCV 2018*, Springer International Publishing, Cham, 2018, pp. 71–86.
  - [8] G. Andresini, A. Appice, D. Malerba, Autoencoder-based deep metric learning for network intrusion detection, *Information Sciences* (2021).
  - [9] C. C. Aggarwal, *Neural Networks and Deep Learning - A Textbook*, 2018.
  - [10] A. Hermans, L. Beyer, B. Leibe, In Defense of the Triplet Loss for Person Re-Identification, *arXiv preprint arXiv:1703.07737* (2017) 1–17.
  - [11] P. Bedi, N. Gupta, V. Jindal, I-siamids: an improved siam-ids for handling class imbalance in network-based intrusion detection systems, *Applied Intelligence* 51 (2021) 1133–1151.
  - [12] J. Bergstra, D. Yamins, D. D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: *ICML*, 2013, pp. 115–123.
  - [13] C. Deng, Z. Chen, X. Liu, X. Gao, D. Tao, Triplet-based deep hashing network for cross-modal retrieval, *IEEE Transactions on Image Processing* PP (2018) 1–1.
  - [14] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: *CISDA*, 2009, pp. 1–6.
  - [15] H. O. Song, Y. Xiang, S. Jegelka, S. Savarese, Deep metric learning via lifted structured feature embedding, in: *Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4004–4012.
  - [16] P. Bedi, N. Gupta, V. Jindal, Siam-ids: Handling class imbalance problem in intrusion detection systems using siamese neural network, *Procedia Computer Science* 171 (2020) 780 – 789. *Third International Conference on Computing and Network Communications (CoCoNet'19)*.
  - [17] L. Liu, P. Wang, J. Lin, L. Liu, Intrusion detection of imbalanced network traffic based on machine learning and deep learning, *IEEE Access* 9 (2021) 7550–7563.
  - [18] S. Huang, K. Lei, Igan-ids: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks, *Ad Hoc Networks* 105 (2020) 1–11.
  - [19] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot, *Sensors* 17 (2017) 1–17.
  - [20] X. Ma, W. Shi, Aesmote: Adversarial reinforcement learning with smote for anomaly detection, *IEEE Transactions on Network Science and Engineering* (2020) 1–1.
  - [21] G. Caminero, M. Lopez-Martin, B. Carro, Adversarial environment reinforcement learning algorithm for intrusion detection, *Computer Networks* 159 (2019) 96 – 109.