INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING
DOI: 10.46300/9106.2022.16.84 Volume 16, 2022

Multiple IoT based Network Attacks Discrimination by Multilayer Feedforward Neural Networks

Vanya Ivanova
French Faculty of Electrical Engineering
Technical University of Sofia
8 Kliment Ohridski Blvd., 1756 Sofia
Bulgaria

Received: July 5, 2021. Revised: December 28, 2021. Accepted: January 15, 2022. Published: January 17, 2022.

Abstract: - **In this paper a new neural model for detection of multiple network IoT-based attacks, such as DDoS TCP, UDP, and HHTP flood, is presented. It consists of feedforward multilayer network with back propagation. A general algorithm for its optimization during training is proposed, leading to proper number of neurons in the hidden layers. The Scaled Gradient Descent algorithm and the Adam optimization are studied with better classification results, obtained by the developed classifiers, using the latter. Tangent hyperbolic function appears to be proper selection for the hidden neurons. Two sets of features, gathered from aggregated records of the network traffic, are tested, containing 8 and 10 components. While more accurate results are obtained for the 10-feature set, the 8-feature set offers twice lower training time and seems applicable for real-world applications. The detection rate for 7 of 10 different network attacks, primarily various types of floods, is higher than 90% and for 3 of them – mainly reconnaissance and keylogging activities with low intensity of the generated traffic, deviates between 57% and 68%. The classifier is considered applicable for industrial implementation.**

Key-Words: - **DDoS, TCP flood, UDP flood, HTTP flood, IoT, attack detection, neural network**

I. INTRODUCTION

Aggregated records of network traffic, exchanged among multiple front-end devices, many of which infected IoT units with various malware, part of botnets, and back-end machines, assuring web, storage and specific services to multiple clients, contain traces of undergoing Distributed Denial of Service (DDoS) attacks [1]. Once detected by these traces, such attacks could be blocked or at least mitigated to an extent that would render regular services still operable. Machine learning is a field that provides techniques for this kind of analysis.

Ur Rehman et al. [7] propose Gated Recurrent Unit (GRU) as alternative to Recurrent Neural Networks (RNN) and Naïve Bayes (NB) to detect DDoS reflection and exploitation attacks. They achieve accuracy of 99.69% for the first type and 99.94% – for the second. Nazih et al. [8] also incorporate GRU in their study, but combining them with RNN (with F1-score varying between 68.53% and 87.45%, using character-based features and going up to 100% in some cases for token-based features) and Long Short-Term Memories (LSTM, with even higher F1-scores between 99.36% and 100%) as a counter-mechanism to DDoS activities in Voice over IP (VoIP) networks. They also tried combinations with Support Vector Machines (SVM), achieving 72.46% as lowest value for the F1-score, and as high as 98.23% for a particular case. Obviously, the accuracy of classifiers is affected by both the feature set and the variety of attacks to be detected.

Chaudahary and Gupta [9] evaluate the detection rate of TCP, UDP and HTTP flood in a network with multiple IoT devices, applying independently SVM, Random Forest (RF), Logistic Regression (LR) and Decision Tree (DT). They achieve accuracy of 98.06%, 99.17%, 97.50% and 98.34%, respectively. Given a relatively isolated part of a packet switched network with a single or at least limited in number attacks, it becomes possible to detect their presence with simpler classifiers. Software Defined Networks (SDN) give additional freedom into running multiple scenarios of attacks as Ye et al. [10] show. Using 6-component descriptors the authors manage to obtain 95.24% accuracy of DDoS attacks detection with a SVM, while collecting limited volume of the network traffic. Sahoo et al. [11] extend the use of SVM with evolutionary algorithms for the same purpose. They add Kernel Principal Component Analysis (KPCA) and Genetic algorithm (GA), which allows dimensionality reduction of feature space and in the same time finding the optimal parameters of the SVM. The generalizing capability of the classifier has been improved with comparison to traditional SVM. Another reason for that is the introduction of a new kernel function, based on Radial Basis Functions (RBF), called N-RBF. Accuracy of 98.55% is achieved when a ratio of 70:30 is used between the training and the test set.

Based on already developed implementations, Alguliyev et al. [12] propose improved models for LSTM and Convolutional Neural Network (CNN) to detect DDoS attacks, particularly aimed at social media services. The F-measure after testing with the CNN is 0.8683 (0.3630 higher than traditional CNN), and for the LSTM – 0.8138 (0.1374 higher than other, commonly used LSTM). Li and Lu [13] combine LSTM with Bayes Approach (BA) to overcome the limitations of LSTM alone, related to difficulties of the construction process, insufficient accuracy or low generalization. Detection accuracy is reported to be 98.15% with this combined classifier. Lu and Tian [14], in their more recent study, tried to enhance the performance of classifiers, applied to Advanced Metering Infrastructure (AMI), dealing successfully with data imbalance and the large number of dimensions, typical for the features used. LSTM has been combined with dimensionality reduction through stacked autoencoder (SAE) and thus the capability of spotting abnormal traffic and to get efficiently the bidirectional structuring descriptors in the same time. Accuracy, achieved over the NSL-KDD dataset, is 0.9943, while the F1-score is 0.9940 with False Alarm Rate (FAR) as low as 0.0036. Shurman et al. [15] incorporate LSTM into Intrusion Detection System (IDS) for detecting DDoS attacks, building 3 different models that achieve 91.54%, 96.74% and 99.19% accuracy over the test set, respectively. In comparison, two of the previously implemented Random Forest models achieve 99.0% and 73.9% accuracy. The LSTM approach is considered effective not only on the general type of DDoS activities, but also on the reflection type of DDoS, where other approaches does not seem to be effective enough.

Deep learning techniques prove to be especially efficient into discovering DDoS attacks with the help of Fog Computing (FC) [16]. It is innovative paradigm that makes use of locally implemented analysis over data portions, generated by front-end devices, associated with the edge of a complete system for attack prevention. Using SDN controllers, suspicious activity on transport and network levels could be investigated and makes it possible to filter malicious packets, while forwarding packets from the normal traffic. LSTM are at the base of this approach, connecting them to the user, for and cloud system into complete framework. It is shown that 2 hidden layers, comprising of 128 neurons, are enough in order to reach 95.89% validation accuracy , which rises to 97.21% for 3 hidden layers [16]. Another example of a complete system for DDoS detection is LUCID [17], reaching accuracy of 99.67%. It uses CNN and is being optimized to extend that lead to 40 times faster execution to previous systems. Ujjan et al. [18] propose sampling with adaptive polling, together with sFlow, as a measure to enhance the deep learning efficiency into discovering DDoS in SDN, embedding IoT devices. Disadvantages, such as low precision, higher memory demand and computational overhead, could be avoided. True Positive Rate reaches 95% with only 4% of the False Positive Rate (FPR). Deep CNNs are also adapted to the detection of DDoS activities in Cyber-Physical Systems (CPSs), especially those using the benefits of 5G networks [19]. Testing over real network data with present botnets, which are sources for silent calls, unwanted signaling, spam (including SMS) and other disrupting actions, discrimination of 91% between normal and attacked cell becomes possible. DDoS attacks also threaten highly distributed types of services, such as the Bitcoin ecosystem and other blockchain systems. Back et al. [20] propose multilayer neural network with preliminary reduction of the number of dimensions of features, using Principal Component analysis (PCA), as a possible solution to this problem. They grouped the extracted data to Block and Transaction level with further estimation of whether the transactions are input or output. Resulting accuracy over the test set reaches 55.12% for DDoS related samples and 72.36% - for normal samples. Stacked Auto-Encoder Multilayer Perceptron (SAE-MLP) is the base of the DLSDN system, proposed by Ahuja et al. [21] for detecting DDoS attacks in SDN. Accuracy score from tests reaches 99.75% when working with a dataset, containing TCP-SYN, UDP flood and ICMP flood samples, along with normal traffic samples.

Comparison with CNN, LSTM, CNN-LSTM and SVC-SOM (Linear Support Vector Classifier with Self-Organizing Map) shows differences towards lower accuracy with 1.01%, 4.15%, 0.27%, and 0.3%, respectively. DeepDefense [22] is another example of a complete system, in this case using RNN as foundation, for detecting DDoS, where enhanced training reduces the detection error from 7.52% to 2.10%.

The main goal of this study is to find an optimal configuration of multilayer feedforward neural network for detection of multiple DoS, DDoS and reconnaissance attacks, carried out by IoT botnets by analyzing the network traffic to various machines, offering legitimate services. Such a neural model would allow simpler and faster implementation of a detection tool with a comparison to others more complicated solutions, such as recurrent neural networks and others.

In Section 2 the test database is described, together with the mutual distribution of samples over the features for classification, the general structure of the classifier and algorithm for its optimization. Section 3 contains the experimental results, which are commented in Section 4. Conclusion is made in Section 5.

II. METHODOLOGY AND APPROACH

A. *Test database*

The test database is developed by a group of researchers from the University of New South Wales in Canberra, Australia [1]. It contains records from sessions among 4 simulated IoT bots, using Kali Linux, infected by malware, that carries out both DoS (Denial of Service) and DDoS (Distributed Denial of Service) by TCP (Transmission Control Protocol), UDP (User Datagram Protocol) and HTTP (Hypertext Transfer Protocol), as well as Keylogging, Data theft activities, OS Fingerprinting and Sniffing. These bots generate malicious traffic to an Ubuntu Server and Ubuntu Mobile, Windows 7, and Metasploitable workstations from internal network. Normal activities from the infected IoT devices with their related normal network traffic are also being captured by an Ubuntu Tap, corresponding to simulated weather station, smart fridge, controlled lights, automated garage door and an intelligent thermostat, using the MQTT (Message Queuing Telemetry Transport) protocol.

Ten features from the network traffic records are extracted as most promising for spotting the undergoing attacks – *seq* – sequence number of the capturing software, *min* – minimal duration of a record, *mean* – average duration of a record, *max* – maximum duration of a record, *state_number* – identifier for a feature state, *N_IN_Conn_P_SrcIP* – number of incoming connections for a source IP address, *N_IN_Conn_P_DstIP* – number of incoming connections for a destination IP address, *srate* – number of packets in a second between source and destination node, *drate* – rate of packets between destination and source node, and *stddev* – standard deviation of the registered records.

The part of training samples in the current study is 2934817, and the test samples are 733705. Class 0 corresponds to normal traffic, 1, 2, and 3 – to DoS TCP, UDP, HTTP, respectively, 4, 5, and 6 – to DDoS TCP, UDP, HTTP, respectively, 7 – Keylogging, 8 – Data Exfiltration, 9 - OS Fingerprinting, and 10 – Service Scan. Distribution of training samples by class is given in Fig. 1, with a similar proportion for the test set.

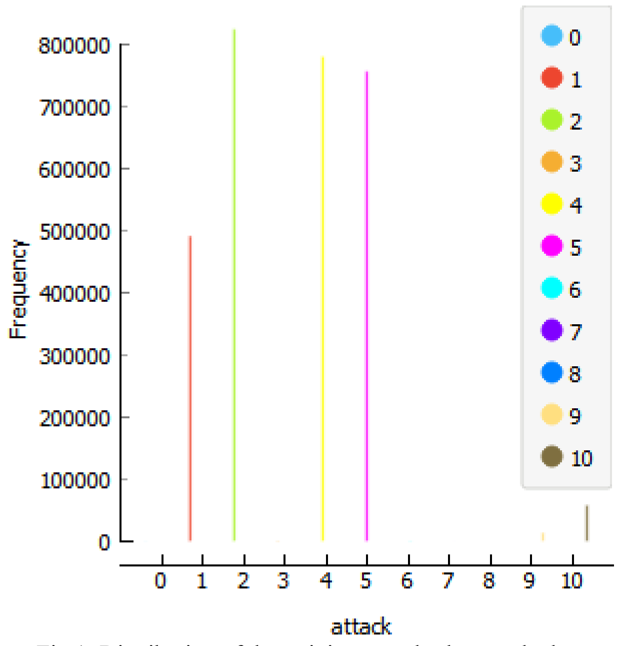


Fig 1: Distribution of the training samples by attack class

B. *Distribution of features*

The full set of 10 features need to be investigated as a distribution over all samples from the test database in order to determine the most influencing components over the separation of the classes. One of the simplified methods to accomplish this task is the FreeViz [6]. It is multivariate visualization approach, which makes use of simple physical interpretation of charge interaction in space, based on generated forces of attraction and repelling and the resulting movement of the charges until stable configuration with minimal potential energy is met [6]. Charges in this instance are samples – those from different classes are going apart from each other, and those from one and the same class – towards a mutual center. The final redistribution is presented in two-dimensional space (2D projection), from where it could be judged which components have the greatest influence on the class discrimination. The computational steps are based on the following simple considerations - the force of interaction between charge q_1 and q_2 is $\boldsymbol{F}_{q1\text{->}q2}$. A charge typically has the following components $\boldsymbol{q} = [q^1, q^2, \ldots, q^n]$. The corresponding n-dimensional point could be projected in 2D by the matrix $\boldsymbol{A}$ with i-th row $A^i = [A_x^i, A_y^i]$, which will transform the i-th base component to its x and y components in 2D [6]. So, the sample will be transformed as $\boldsymbol{q'} = \boldsymbol{qA}$, where $q_x = \text{sum}_i(e^i A_x^i)$ and $q_y = \text{sum}_i(e^i A_y^i)$ [6]. If E is the potential energy of the system and A – the work associated with a movement of a charge by particular force, then the following equations hold true [6]:

$$dE = A = -\boldsymbol{F}_{q_1 \to q_2} d\boldsymbol{q}', \tag{1}$$

$$\boldsymbol{F}_q = \textstyle\sum_{q_1 \neq q_2} \boldsymbol{F}_{q_1 \to q_2}, \tag{2}$$

$$dE = -\textstyle\sum_q \boldsymbol{F}_q d\boldsymbol{q}', \tag{3}$$

$$dE = -\textstyle\sum_q \boldsymbol{F}_q (\boldsymbol{q} d\boldsymbol{A}), \tag{4}$$

$$\frac{dE}{dA_x^i} = -\textstyle\sum_q \boldsymbol{F}_{q,x} q^i. \tag{5}$$

Analogous is the process to project data samples and find the location of related feature axes over the verticals, that is y in 2D. Then optimization is performed using the Gradient Descent algorithm to rotate all projected vectors, corresponding to selected features, that they relate the most to the clusters of samples in the same space of projection.

Using the FreeViz tool from the Orange data mining software and passing all test samples, the initial distribution in 2D with equal spread of features along a circular pattern is given in Fig. 2.a.

After optimization by the FreeViz in 2D (Fig.2.b) it is observed that the *seq* and *N_IN_Conn_P_SrcIP* axes cover the least of changes over all samples. The second feature is connected primarily to vectors, belonging to class 1 attack (shown in red in the lower right part of the figure), but with a small variance, thought to be less informative than *drate*, *srate* and *N_IN_Conn_P_DstIP*. This is the reason to select in a second set of experiments to train and test a classifier with only 8 features, omitting *seq* and *N_IN_Conn_P_SrcIP*.

C. Proposed multilayer feedforward neural network architecture

Taking into account the work of Koroniotis et al. [1], where a Recurrent Neural Network with 7 hidden layers is proposed for detecting multiple IoT based network attacks, predominantly DDoS ones, an alternative neural network structure is proposed within this study. It is a feedforward neural network with similar degree of complexity in number of neurons, again with 7 hidden layers, but without the recurrent connections in it (Fig. 3).

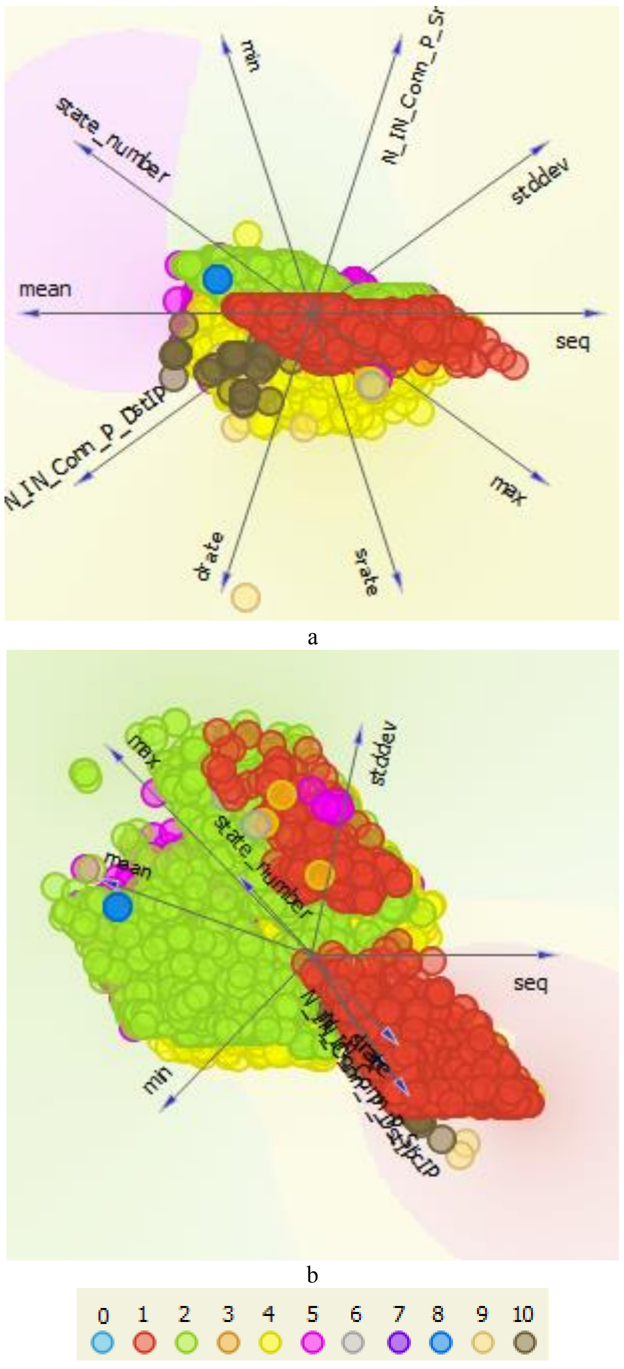


Fig 2: FreeViz representation of test data in 2D: a) initial location of samples, b) after optimization (below is the color legend for persistent classes)

The input layer (IL) consists of 10 connections for the complete set of 10 features (f1, …, f10), which in one of the tested variations of the network is reduced to 8 features. Then, follow the hidden layers – HL1, HL2, …, HL7, and finally the output layer (OL) consists of 11 connections a0 – indicating normal traffic, and a1-a10 – for the 10 types of attacks present in the network.

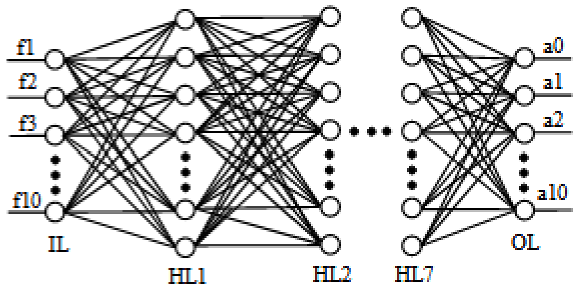


Fig 3: Constructed multilayer neural network for detection of IoT based network attacks

Feature components could be arranged as a vector $\vec{f} = \{f_1, f_2, \dots, f_{N_1}\}$, where N_1 is either 8 or 10. For all the layers k = 1, …, 9, including the IL and the OL, the weights of neurons also could be grouped in a vector $\overrightarrow{w_i^k} = \{w_{1i}^k, \dots, w_{N_k i}^k\}$. Following one of our previous studies on discriminating network traffic to malicious and normal one, using feedforward neural networks [2], all neurons from the hidden layers are preferred to have tangent hyperbolic (Tanh) activation function, that is the function g_k, $k = 2, \dots, 8$. Then, the resulting values from the output of each neuron could also be represented as a vector $\overrightarrow{o^k} = \{o_1^k, \dots, o_{N_k}^k\}$. It is expected during the training some of the connections between neurons from adjacent layers to be lost, and other – harden, but initially the network is fully connected. The level of the resulting signal from each neuron i in layer k could be found according to the equation below [3]:

$$o_i^k = g_k\left(h_i^k\right) = g_k\left(\overrightarrow{w_i^k}\,.\,\overrightarrow{o^{k-1}} + b_i^k\right) = b_i^k + \sum_{j=1}^{N_k} w_{ji}^k\, o_j^{k-1}. \quad (8)$$

In (8) h_i^k is the weighted input for the same i-th neuron from k-th layer, which could be also biased by the component b_i^k, associated alone with the same neuron.

Each input vector $\vec{f_i}$ has associated target value (or label) t_i, for $i = 1, \dots, M$. Comparing all output values from the last layer (OL) o_i^9 with designated target t_i, it is possible to evaluate the accuracy of classification. Both the mean square (*MSE*) and the root mean square (*RMSE*) errors could be employed for integral evaluation in that process [3]:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(o_i - t_i)^2, \quad (9)$$

$$RMSE = \frac{\sum_{i=1}^{M}(t_i - o_i)^2}{\sum_{i=1}^{M}(t_i - \bar{o})^2}, \quad (10)$$

where $\bar{o}$ is the average output for the particular class (attack).

The update of the weight for neuron j from layer k, connected to neuron i, is done according to [4]:

$$\Delta w_{ij}^k = -\alpha \frac{\partial E(f)}{\partial w_{ij}^k} = -\alpha \delta_j o_i, \quad (11)$$

$$\delta_j = g'_k\left(h_i^k\right)\sum_k w_{kj}\delta_k, \quad (12)$$

where the sum in (12) covers all neurons, taking as input the resulting value from the output of the j-th neuron. In (11) f denotes current feature sample and α is the learning rate.

D. *Finding the optimal structure of the classifier*

No precise rule exists for choosing the number of neurons in the hidden layers of a feedforward neural network with back propagation, in this instance N_{HL2}, N_{HL3}, N_{HL4}, N_{HL5}, N_{HL6}, N_{HL7}, N_{HL8}. In the current study, an iterative approach is selected with step-wise increase of these numbers, according to Fig. 4.

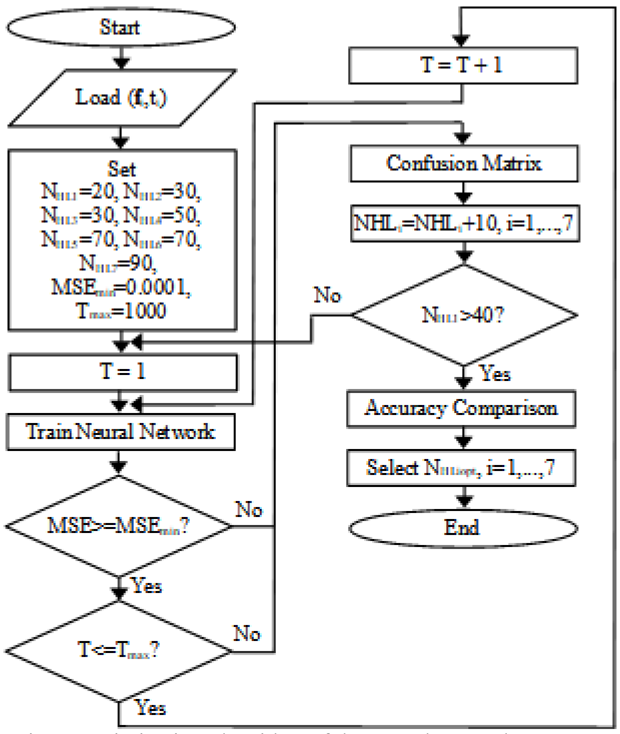


Fig 4: Optimization algorithm of the neural network structure

Twelve implementations of the neural network of the studied type are tested, having respectively 20-30-30-50-70-70-90, 30-40-40-60-80-80-100 and 40-50-50-70-90-90-110 neurons in the hidden layers. Each variant is trained and tested once with 10 and separately with 8 features – one time using the Scaled Gradient Descent (SGD) algorithm and second time – using the Adam optimization.

According to algorithm from Fig. 4, end of training comes when either the *MSE* falls down to 0.0001 or the number of epochs reaches 1000. Confusion matrices of the complete set of tested neural networks are then compared - as per the average proportion of the actual detected attacks, the ability to discriminate attack vs. non-attack input and also training and testing times are considered in order to derive the optimal combination of number of neurons from the hidden layers, plus the optimal set of input features – either 8, or 10.

The single valued parameters, that could be derived from confusion matrices are the True Positives (*TP*), True Negatives (*TN*), False Positives (*FP*), False Negatives (*FN*), Precision, Recall, Specificity, *F1*-measure, Log-loss, and Classification accuracy (*CA*) [5].

III. EXPERIMENTAL RESULTS

The following components of an IBM compatible PC, used for experimentation, supported the numerical simulations – a 4-core CPU Intel Xeon E5-1620, working at 3.50 GHz, having 256 kB L1 cache, 1 MB – L2 and 10 MB – L3; 64 GB RAM; 2 TB HDD. The operating system is Microsoft Windows 10 Professional, version 20H2. Visual programming of the neural networks, together with the training, validation and testing is done with the Orange v. 3.28 application.

The parameters of the neural networks that are trained and tested within the current study are given in Table 1. Regularization parameter for all modifications is $\alpha = 0.0001$ and the maximum number of epochs for training is set to 1000.

Table 1: Neural networks parameters

Name	Number of features	Training algorithm	Number neurons by layers from input to output
NN_1	10	SGD	10-20-30-30-50-70-70-90-11
NN_2	10	SGD	10-30-40-40-60-80-80-100-11
NN_3	10	SGD	10-40-50-50-70-90-90-110-11
NN_4	10	Adam	10-20-30-30-50-70-70-90-11
NN_5	10	Adam	10-30-40-40-60-80-80-100-11
NN_6	10	Adam	10-40-50-50-70-90-90-110-11
NN_7	8	SGD	10-20-30-30-50-70-70-90-11
NN_8	8	SGD	10-30-40-40-60-80-80-100-11
NN_9	8	SGD	10-40-50-50-70-90-90-110-11
NN_{10}	8	Adam	10-20-30-30-50-70-70-90-11
NN_{11}	8	Adam	10-30-40-40-60-80-80-100-11
NN_{12}	8	Adam	10-40-50-50-70-90-90-110-11

The ratio of discovered attacks during training as a proportion of the actual ones, using the full set of 10 features for both the Adam and SGD algorithms for all 3 tested neural networks, is given in Table 2.

Table 2: Proportion discovered attacks from the actual instances from full training set validation at 10 features, in %

Attack	NN_1	NN_2	NN_3	NN_4	NN_5	NN_6
0	93.8	93.5	89.2	88.9	93.0	81.1
1	95.8	97.0	98.5	95.7	95.2	98.1
2	100.0	100.0	100.0	100.0	100.0	100.0
3	86.1	78.8	82.3	81.5	94.1	84.1
4	96.1	95.6	93.5	95.1	95.6	94.9
5	100.0	99.9	99.9	99.9	99.9	100.0
6	89.6	92.5	73.3	91.1	72.4	96.9
7	0.0	52.5	66.1	86.4	66.1	88.1
8	0.0	0.0	0.0	0.0	16.7	50.0
9	56.4	54.4	56.7	55.2	57.3	39.4
10	96.3	96.1	96.1	93.8	95.2	98.9
All	74.0	78.2	77.8	80.7	80.5	84.7

In the same time, the resulting proportions over the training set, when using only 8 of the 10 features are given in Table 3.

Table 3: Proportion discovered attacks from the actual instances from full training set validation at 8 features, in %

Attack	NN_7	NN_8	NN_9	NN_{10}	NN_{11}	NN_{12}
0	73.8	87.0	79.5	57.8	88.6	53.2
1	76.9	76.7	78.3	73.6	71.8	73.3
2	99.7	99.8	99.7	99.6	99.9	99.5
3	35.3	37.1	38.9	80.6	27.6	30.4
4	97.5	94.6	95.9	98.2	98.2	96.2
5	99.9	99.7	99.9	99.9	99.4	99.4
6	34.4	31.8	42.4	37.8	64.0	33.5

7	0.0	16.9	1.7	0.0	10.2	18.6
8	0.0	0.0	0.0	0.0	0.0	0.0
9	1.7	5.1	5.0	4.9	4.2	5.0
10	99.0	99.0	98.9	98.3	96.5	97.2
All	56.2	58.9	58.2	59.2	60.0	55.1

Testing of the trained models over the complete test set leads to proportions of correctly discovered attacks to all by types, shown in Table 4, when using 10 features.

Table 4: Proportion discovered attacks from the actual instances from full testing set validation at 10 features, in %

Attack	NN_1	NN_2	NN_3	NN_4	NN_5	NN_6
0	86.0	88.8	88.8	85.0	91.6	80.4
1	95.3	97.0	96.3	95.7	95.2	98.0
2	100.0	100.0	100.0	100.0	100.0	99.9
3	65.8	79.1	90.7	84.1	90.4	83.4
4	96.0	95.6	96.2	95.2	95.6	94.9
5	99.9	99.9	100.0	99.9	99.9	100.0
6	50.7	88.7	87.2	92.1	68.5	95.6
7	0.0	50.0	57.1	92.9	57.1	92.9
8	0.0	0.0	0.0	0.0	0.0	0.0
9	42.1	54.9	56.1	55.6	57.6	40.2
10	98.1	96.2	96.4	93.9	95.1	98.9
All	73.4	85.0	86.9	89.4	85.1	88.4

The results of discriminating attacks from the records in the test set, employing 8 features, are given in Table 5.

Processing times of training and testing on both the train and test set are presented in Table 6.

Classification accuracy (CA), F1-measure, Precision, Recall, LogLoss and Specificity, obtained from working with the test set could be seen in Table 7.

As described in the discussion bellow, the optimal configuration for 10 features is selected from NN_5, and for 8 features – for NN_{11}. The confusion matrices from classification over the test set are shown in Fig. 5 and Fig. 6, respectively.

Table 5: Proportion discovered attacks from the actual instances from full testing set validation at 8 features, in %

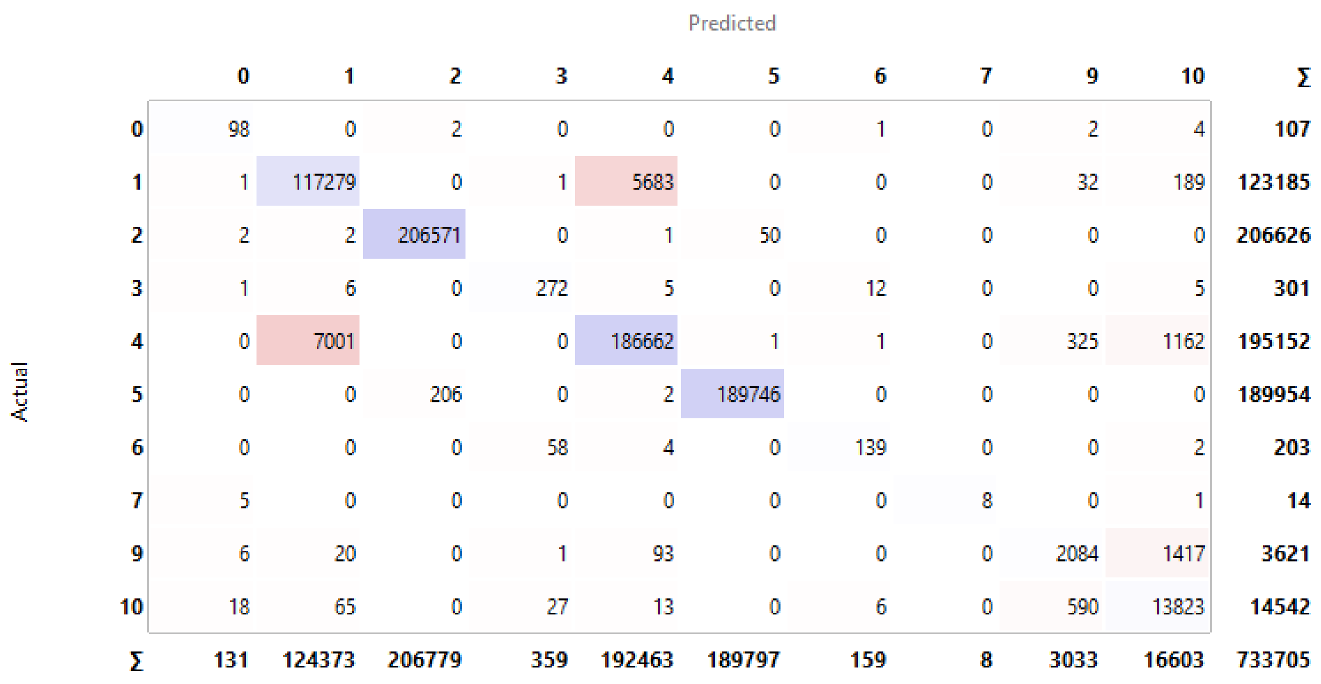
Attack	NN_7	NN_8	NN_9	NN_{10}	NN_{11}	NN_{12}
0	73.8	87.0	79.5	57.8	88.6	53.2
1	76.9	76.7	78.3	73.6	71.8	73.3
2	99.7	99.8	99.7	99.6	99.9	99.5
3	35.3	37.1	38.9	80.6	27.6	30.4
4	97.5	94.6	95.9	98.2	98.2	96.2
5	99.9	99.7	99.9	99.9	99.4	99.4
6	34.4	31.8	42.4	37.8	64.0	33.5
7	0.0	16.9	1.7	0.0	10.2	18.6
8	0.0	0.0	0.0	0.0	0.0	0.0
9	1.7	5.1	5.0	4.9	4.2	5.0
10	99.0	99.0	98.9	98.3	96.5	97.2
All	56.2	58.9	58.2	59.2	60.0	55.1

Table 6: Neural networks processing times

Neural Network	Training time, sec	Testing over train set, sec	Testing over test set, sec
NN_1	60750.79	152.92	30.69
NN_2	52097.14	223.74	40.53
NN_3	63164.43	143.01	36.65
NN_4	20379.68	130.45	40.61
NN_5	36476.55	155.19	40.72
NN_6	44199.82	155.40	49.26
NN_7	42569.97	179.38	29.90
NN_8	28490.63	214.32	49.12
NN_9	32927.37	196.71	44.46
NN_{10}	13113.01	191.26	35.71
NN_{11}	159480.7	167.50	51.29
NN_{12}	15747.10	233.59	62.84

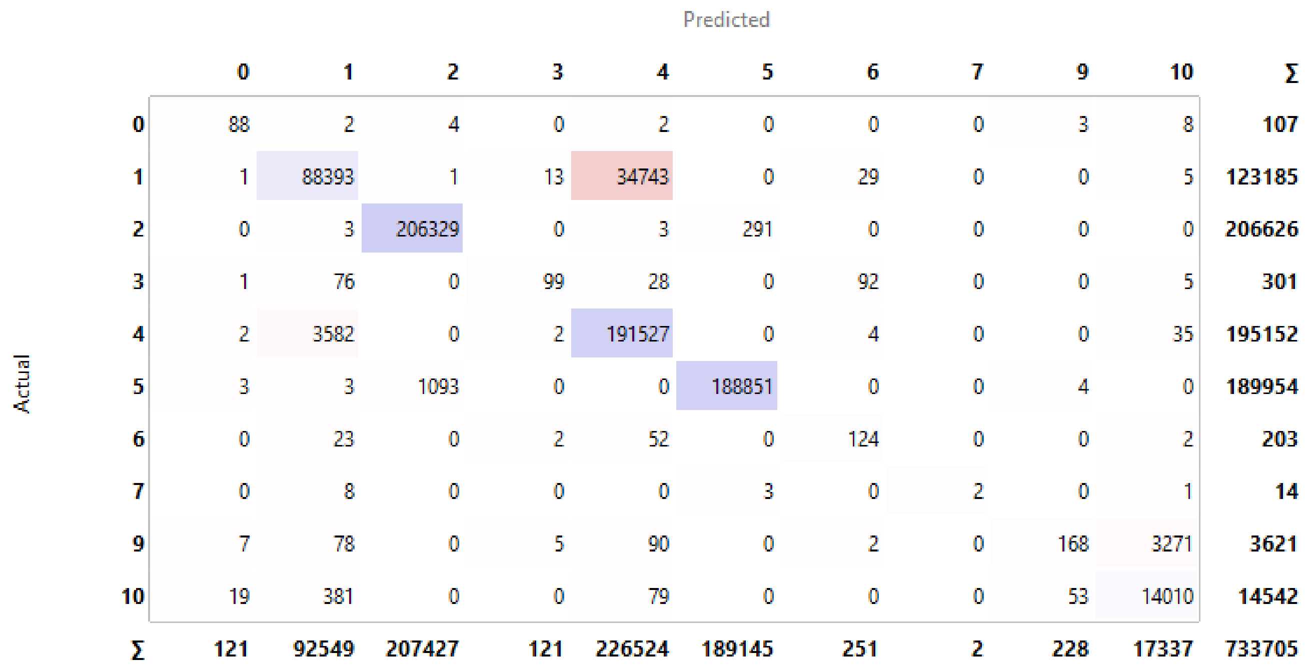
Table 7: Neural networks efficiency

Neural Network	CA	F1	Precision	Recall	LogLoss	Specificity
NN_1	0.9795	0.9793	0.9795	0.9795	0.0413	0.9954
NN_2	0.9797	0.9796	0.9799	0.9797	0.0411	0.9956
NN_3	0.9772	0.9771	0.9783	0.9772	0.0485	0.9956
NN_4	0.9760	0.9761	0.9764	0.9760	0.0511	0.9951
NN_5	0.9768	0.9768	0.9770	0.9768	0.0502	0.9950
NN_6	0.9797	0.9792	0.9805	0.9797	0.0404	0.9959
NN_7	0.9482	0.9450	0.9508	0.9482	0.1050	0.9842
NN_8	0.9399	0.9371	0.9408	0.9399	0.1318	0.9823
NN_9	0.9465	0.9438	0.9479	0.9465	0.1066	0.9841
NN_{10}	0.9443	0.9410	0.9476	0.9443	0.1046	0.9825
NN_{11}	0.9399	0.9361	0.9441	0.9399	0.1273	0.9808
NN_{12}	0.9362	0.9329	0.9387	0.9362	0.1332	0.9805


Predicted
Actual

	0	1	2	3	4	5	6	7	9	10	Σ
0	98	0	2	0	0	0	1	0	2	4	107
1	1	117279	0	1	5683	0	0	0	32	189	123185
2	2	2	206571	0	1	50	0	0	0	0	206626
3	1	6	0	272	5	0	12	0	0	5	301
4	0	7001	0	0	186662	1	1	0	325	1162	195152
5	0	0	206	0	2	189746	0	0	0	0	189954
6	0	0	0	58	4	0	139	0	0	2	203
7	5	0	0	0	0	0	0	8	0	1	14
9	6	20	0	1	93	0	0	0	2084	1417	3621
10	18	65	0	27	13	0	6	0	590	13823	14542
Σ	131	124373	206779	359	192463	189797	159	8	3033	16603	733705

Fig 5: Confusion matrix for NN_5 from classification of the test set


Predicted
Actual

	0	1	2	3	4	5	6	7	9	10	Σ
0	88	2	4	0	2	0	0	0	3	8	107
1	1	88393	1	13	34743	0	29	0	0	5	123185
2	0	3	206329	0	3	291	0	0	0	0	206626
3	1	76	0	99	28	0	92	0	0	5	301
4	2	3582	0	2	191527	0	4	0	0	35	195152
5	3	3	1093	0	0	188851	0	0	4	0	189954
6	0	23	0	2	52	0	124	0	0	2	203
7	0	8	0	0	0	3	0	2	0	1	14
9	7	78	0	5	90	0	2	0	168	3271	3621
10	19	381	0	0	79	0	0	0	53	14010	14542
Σ	121	92549	207427	121	226524	189145	251	2	228	17337	733705

Fig. 6: Confusion matrix for NN_{11} from classification of the test set

IV. DISCUSSION

The first observation, comparing the training efficiency of SGD and Adam optimization at 10 features (Table 2), is that the classifiers, employing SGD, achieve from 2.3% to 6.9% less correctly predicted number of attacks, than those, using Adam. When using 8 features, the difference is from 1.1% to 3%, after applying validation over the full training set (Table 3). During testing with 10 features, Adam leads to increase of correct prediction from 0.1% to 16% (Table 4), and for 8 features – from 1.1% to 3% (Table 5). These results indicate that Adam could be preferred in front of SGD for the particular task.

Training times are smaller for the Adam algorithm – in the order between 2 and 3 times (Table 6). Yet, this is another factor, that gives flavor towards the selection of Adam optimization for future implementations of classifiers for multiple network attacks discrimination. The period, taken for full test set classification, is comparable between Adam and SGD, which means that the structures of the classifiers are similar, with almost equal number of preserved connections between neurons and thus almost equal number of computations to accomplish. Training by one or the other algorithm would not significantly affect the processing speed of a real-world module, embedded in a monitoring position within the network.

Although the classification accuracy (*CA*) is higher for smaller number of neurons in the hidden layers for some of the realizations (Table 7), it is worth noting that the number of correctly discovered normal instances of network connections is highest for the networks with middle number of neurons (seen from confusion matrices in Fig. 5 and Fig. 6), that is the configuration - 10-30-40-40-60-80-80-100-11. This is true for both variants, employing 8 and 10 features. The maximal variation of *CA* for Adam optimization is 0.0029 among the three structures at 10 features and 0.0044 – at 8 features. Thus, it is found that the optimal configurations for 10 and 8 features are NN_5 and NN_{11}, respectively. Using 10 features leads to 0.0369 higher *CA*, or 3.93%. Similar are the proportions for the rest of the accuracy parameters, presented in Table 7. It turns out that is more difficult to spot normal traffic with 8 features – the correct rate of discovery of these instances is 11.36% less (Fig. 5 and 6). On the other hand, the training time of NN_{11} is almost twice less than that of NN_5, which means that in practical scenarios with high critical demands for adaptation speed (re-training) of the classifier, it would be preferable to use the 8-feature implementation.

Looking at the most precise configuration, NN_5, the least discovered attacks are – 7 – with 57.1% of the actual instances, 9 – with 57.6%, 6 – with 68.5% and all the rest are above 90% with a modest rate of 90.4% for the 3rd type of attack and 100% - for the 2nd (Fig. 5). One of the main reasons for this result is the lower number of instances, corresponding to these types of attacks. The complete network connections and associated traffic ratio among various attacks is however close to what is being observed during ral DoS and DDoS activities, so these discovery rates could be acknowledged as satisfactory.

Comparison between the Recurrent Neural network (RNN), developed in [1], and NN_5 with regard to the DDoS attacks, implemented in the test network environment, is presented in Table 9. The number of layers and the number of hidden neurons in the RNN are the same as those in NN_5. Although the *CA* is smaller with 0.0194, for all these attacks NN_5 performs better as per the F1-measure.

Table 9. Performance comparison between RNN, [1], and the proposed classifier for DDoS HTTP, TCP and UDP

Parameter	*CA*	*Precision*	*Recall*	*F1*
DDoS HTTP, RNN [1]	0.9932	0.9930	0.9970	0.0147
DDoS HTTP, Proposed	0.9998	0.8742	0.6847	0.7679
DDoS TCP, RNN [1]	0.9999	0.9999	0.9999	0.0105
DDoS TCP, Proposed	0.9805	0.9698	0.9565	0.9631
DDoS UDP, RNN [1]	0.9999	0.9999	0.9999	0.0063
DDoS UDP, Proposed	0.9996	0.9997	0.9989	0.9993

In Table 10, it could be seen similar results for the DoS attacks – NN_5 has smaller CA with 0.0177, compared to RNN, for the TCP flood, but again F1 is much higher for all considered instances.

Table 10. Performance comparison between RNN, [1], and the proposed classifier for DoS HTTP, TCP and UDP

Parameter	*CA*	*Precision*	*Recall*	*F1*
DoS HTTP, RNN [1]	0.9806	0.9898	0.9845	0.0314
DoS HTTP, Proposed	0.9998	0.7576	0.9036	0.8242
DoS TCP, RNN [1]	0.9999	0.9999	0.9999	0.0713
DoS TCP, Proposed	0.9822	0.9429	0.9521	0.9475
DoS UDP, RNN [1]	0.9999	0.9999	0.9999	0.0126
DoS UDP, Proposed	0.9996	0.9990	0.9997	0.9993

Table 11 reveals resulting accuracy in discriminating all remaining types of attacks by the proposed multilayer feedforward neural network, which tends to be higher than that of the RNN, with the Service Scan case, where both classifier perform almost equally.

Given all resulting parameters for accuracy from Tables 9-11 and the fact that the feedforward neural network has a simpler structure than the RNN, where recurrent connections are introduced for some of the neurons, that is taking more time for training and to perform the calculations at the classification stage, the proposed here NN_5 could be considered more efficient classifier.

Table 11. Performance comparison between RNN, [1], and the proposed classifier for OS Fingerprinting, Service Scan, Data Exfiltration and Keylogging

Parameter	*CA*	*Precision*	*Recall*	*F1*
OS Finger, RNN [1]	0.9917	0.9984	0.9931	0.0587
OS Finger, Proposed	0.9952	0.8325	0.9505	0.8876
Service Scan, RNN [1]	0.9957	0.9986	0.9971	0.2159
Service Scan, Proposed	0.9952	0.8342	0.9515	0.8890
Data Exfilt., RNN [1]	0.9875	0.0000	0.0000	0.0000
Data Exfilt., Proposed	0.9966	0.6871	0.5755	0.6264

Keylogging, RNN [1]	0.9873	0.9853	0.9178	0.0021
Keylogging, Proposed	0.9999	1.0000	0.5714	0.7272

V. CONCLUSION

In this paper a new implementation of a feedforward multilayer neural network with backpropagation is proposed for classification of IoT-based DoS, DDoS and other malicious network activities. Wide variety of tests prove the Adam optimization procedure as more efficient than the Scaled Gradient Descend algorithm. Tangent hyperbolic activation function tends to be suitable for all neurons from the hidden layers. Both the training and classification times suggest the applicability of the neural model in real-word applications. Using 8, less informative features, of the complete set of 10 features, seems a balanced solution with regard to classification accuracy, which could speed up the training process at least twice. Some of the attacks, such as Keylogging and OS Fingerprinting, given the significantly lower intensity of generated traffic, are more difficult to discover with around 57% detection rate. Additional modifications of the classifier are needed, possibly, incorporating it in a stacked realization with other classifiers in order to catch similar activities, equally well with the more intense flood attacks.

REFERENCES

[1] Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B., Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT dataset. *Future Generation Computer Systems*, Vol. 100, November 2019, pp. 779-796.

[2] ur Rehman, S., Khaliq, M., Imtiaz, S. I., Rasool, A., Shafiq, M., Javed, A. R., ... & Bashir, A. K., DIDDOS: An approach for detection and identification of Distributed Denial of Service (DDoS) cyberattacks using Gated Recurrent Units (GRU). *Future Generation Computer Systems*, vol. 118, pp. 453-466, 2021.

[3] Nazih, W., Hifny, Y., Elkilani, W. S., Dhahri, H., Abdelkader, T., Countering DDoS Attacks in SIP Based VoIP Networks Using Recurrent Neural Networks. *Sensors*, vol. 20, no. 20, 5875, 2020.

[4] Chaudhary, P., Gupta, B. B., DDoS detection framework in resource constrained Internet of Things domain. *In 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE),* pp. 675-67, October 2019.

[5] Ye, J., Cheng, X., Zhu, J., Feng, L., Song, L., A DDoS attack detection method based on SVM in software defined network. *Security and Communication Networks*, vol. 2018, 9804061, 2018.

[6] Sahoo, K. S., Tripathy, B. K., Naik, K., Ramasubbareddy, S., Balusamy, B., Khari, M., Burgos, D., An evolutionary SVM model for DDOS attack detection in software defined networks. *IEEE Access*, vol. 8, pp. 132502-132513, 2020.

[7] Alguliyev, R. M., Aliguliyev, R. M., Abdullayeva, F. J., The improved LSTM and CNN Models for DDoS attacks prediction in social media. *International Journal of Cyber Warfare and Terrorism (IJCWT)*, vol. 9, no. 1, pp. 1-18, 2019.

[8] Li, Y., Lu, Y., LSTM-BA: DDoS detection approach combining LSTM and Bayes. *In 2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, pp. 180-185, 2019.

[9] Lu, G., Tian, X., An Efficient Communication Intrusion Detection Scheme in AMI Combining Feature Dimensionality Reduction and Improved LSTM. *Security and Communication Networks*, vol. 2021, 66310752021, 2021.

[10] Shurman, M., Khrais, R., Yateem, A., DoS and DDoS Attack Detection Using Deep Learning and IDS. *International Arab Journal of Information Technology*, vol. 17, no. 4 A, pp. 655-661, 2020.

[11] Priyadarshini, R., Barik, R. K., A deep learning based intelligent framework to mitigate DDoS attack in fog environment. *Journal of King Saud University-Computer and Information Sciences*, 2019

[12] Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martinez-del-Rincon, J., Siracusa, D., LUCID: A practical, lightweight deep learning solution for DDoS attack detection. *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp.876-889, 2020.

[13] Ujjan, R. M. A., Pervez, Z., Dahal, K., Bashir, A. K., Mumtaz, R., González, J., Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN. *Future Generation Computer Systems*, vol. 111, pp. 763-779, 2020.

[14] Hussain, B., Du, Q., Sun, B., Han, Z., Deep Learning-Based DDoS-Attack Detection for Cyber–Physical System Over 5G Network. *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 860-870, 2020.

[15] Baek, U. J., Ji, S. H., Park, J. T., Lee, M. S., Park, J. S., Kim, M. S., DDoS attack detection

on bitcoin ecosystem using deep-learning. *In 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1-4, September 2019.

[16] Ahuja, N., Singal, G., Mukhopadhyay, D., DLSDN: Deep learning for DDOS attack detection in software defined networking. *In 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 683-688, January 2021.

[17] Yuan, X., Li, C., Li, X. DeepDefense: identifying DDoS attack via deep learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1-8, May 2017.

[18] Demšar, J., Leban, G., Zupan, B. FreeViz — An intelligent multivariate visualization approach to explorative analysis of biomedical data. *Journal of biomedical informatics*, vol. 40, no. 6, pp. 661-671, 2007.

[19] Ivanova, V., Tashev, T., Draganov, I., Detection of IoT based DDoS Attacks by Network Traffic Analysis using Feedforward Neural Networks. *WSEAS Transactions*, 2021 (under review).

[20] Rhys, H., *Machine Learning with R, Tidyverse, and MLR*, Manning Publications, 2020.

[21] Abe, S., *Pattern Classification: Neuro-Fuzzy Methods and their Comparison*, Springer-Verlag, 2001.

[22] Kolo, B., *Binary and Multiclass Classification*, Weatherford Press, 2011.