INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING
DOI: 10.46300/9106.2022.16.81 Volume 16, 2022

Detection of IoT based DDoS Attacks by Network Traffic Analysis using Feedforward Neural Networks

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

Received: July 11, 2021. Revised: December 21, 2021. Accepted: January 14, 2022. Published: January 15, 2022.

Abstract: - **In this paper an optimized feedforward neural network model is proposed for detection of IoT based DDoS attacks by network traffic analysis aimed towards a specific target which could be constantly monitored by a tap. The proposed model is applicable for DoS and DDoS attacks which consist of TCP, UDP and HTTP flood and also against keylogging, data exfiltration, OS fingerprint and service scan activities. It simply differentiates such kind of network traffic from normal network flows. The neural network uses Adam optimization as a solver and the hyperbolic tangent activation function in all neurons from a single hidden layer. The number of hidden neurons could be varied, depending on targeted accuracy and processing speed. Testing over the Bot IoT dataset reveals that developed models are applicable using 8 or 10 features and achieved discrimination error of $4.91.10^{-3}$%.**

Key-Words: - **IoT, DDoS, keylogging, data exfiltration, botnet, network traffic analysis, feedforward neural network**

I. INTRODUCTION

Distributed Denial of Service (DDoS) attacks cause significant financial losses in various industries. A recent research [1] exposes that a single attack of this type could cost between USD 120 000 and USD 2 million to targeted businesses with an annual growth of the size of these activities close to 40%. IoT devices are more often incorporated in such attacks, after being intruded and remotely controlled, as intense generators of requests to online services, rendering them unusable [2]. Efficient measures to detect DDoS processes on a network level, monitoring packet-based traffic, are being sought for many years now [3]. Machine learning is widely exploited by researchers in order to find an efficient tools, such as Support Vector Machines (SVM), Neural Networks (NN), various clustering algorithms and others, through training for self-adaptation to specific properties and time-related changes of the overall profile of network traffic [4].

Chen et al. [5] propose detection system for DDoS attacks aimed at services, offered by cloud-based servers, in which multiple IoT devices, connected to smart poles, can play decisive role. It is a multi-layer discovery system, including off-line learning with iterative update of pre-trained models, where mutual incorporation of network and sensor data has contributed to accuracy levels, ranging from 97.30% to 99.98%. At the base of the classifier lie decision trees.

Edge-centric scheme for detecting and mitigating DDoS attacks, in which an active involvement of IoT devices occurs, is proposed in [6]. Network traffic variations influence the internal structure of short-term memories, as one type of a classifier used, and convolutional neural network (CNN) – as a second one involved in the scheme. Identification accuracy for the first is 98.9% and for the CNN – 99.9%. The proposed approach has low operational delays when implemented over edge servers, more powerful than a personal computer.

Entropy-based detector, relying on Software Define Networking (SDN), comes as a solution to the detection of IoT related DDoS attacks [7]. In that kind of detecting system, the states of SDN, estimated over a data space representation, provide overall certainty of spotting malicious activities between 68% and 99.7%.

Relation between application level DDoS attacks and network flows are studies by Bhardwaj et al. in [8]. They proposed edge computing techniques, which connect incoming traffic to a monitoring service through a fast channel for on-time reaction to stop damaging influence. Reduction of 82% of the unwanted traffic is reported. Intelligent functions through a ShadowNet support that process.

Protocol-agnostic approach proves to be efficient into limiting the effect of DDoS attacks, as described by Doshi et al. [9]. Comparison among the k-Nearest neighbors (k-NN), SVM with linear kernel, decision trees, random forest and a 4-layer neural network shows variation in classification accuracy between 0.91 and 0.99. Still, discriminating non-attack packets from those with malicious content, given their much more higher appearance rate, is challenging, especially with a classifier, which has lower complexity, that is simpler structure.

The benefits of introducing the Software Defined anything (SDx) paradigm along with the newly introduced framework by Yin et al. [10], based on software defined IoT, could strengthen the level of DDoS mitigation when serving multitude of heterogeneous devices. Cosine similarity is proved to be an efficient measure into discriminating incoming packets as malicious vs. non-malicious ones. Reduction of the bandwidth, related to wasted traffic flows as a result from the attack, achieved by the controller-switch during testing is around 6 times compared to the reduction of the DPCC algorithm.

Multi-objective optimization is another approach that proves useful when combined with a deep neural network, increasing the detection rate of DDoS attacks [11]. Jumping Gene NSGA-II algorithm reduces the dimensions of processed data by a CNN, combined with a Long Short-Term Memory (LSTM). Classification accuracy of the attacks is reported to be around 99.03%. Additionally, considerable reduction of the execution time with regards to the training process of about 5 times, compared to other similar methods, supports the applicability of the proposed approach.

The multi-agent approach in Intrusion Detection Systems (IDS), spread around monitored network against DDoS attacks, forms the foundations of another strategy to cope with this problem. In [12], Mehmood et al. use that approach together with Naïve Bayes classifier in order to sense irregular patterns of the network flows. Detection probability increases with around 0.2, compared to other generic IDS implementations, and the detection rate is above 97%.

Most of the presented techniques rely on either deep neural networks or combined classifiers with intricate structure, primarily aiming the discrimination of the ongoing attack, based on protocol and additional features in some cases, such as its purpose. In the presented within this paper study, the aim is to propose a simpler classifier, based on the well-established feedforward neural network with backpropagation, which could distinguish only malicious vs. normal traffic, generated by compromised IoT devices towards machines, offering legitimate services.

In Section 2 a description of the used test database, network traffic features, proposed classifier and algorithm for its optimization are described. Experimental results are contained in Section 3, followed by a discussion in Section 4 and a conclusion in Section 5.

II. PROPOSED CLASSIFIER

A. Experimental setup for obtaining the test database

The Bot-IoT dataset [13] is collected through a network setup, shown in Fig. 1. It is a local area network with internal IP address space. Four of the machines, namely Bot 1 to Bot 4, are running Kali Linux with simulated IoT services over the Message Queuing Telemetry Transport (MQTT) protocol. An Ubuntu Server offering web-based services, 2 workstations operated by Windows 7 and Metasploitable, as well as Ubuntu Mobile as a mobile station are being attacked by all bots. A separate machine, also working under Ubuntu, plays the role of a tap, monitoring and recording network traffic.

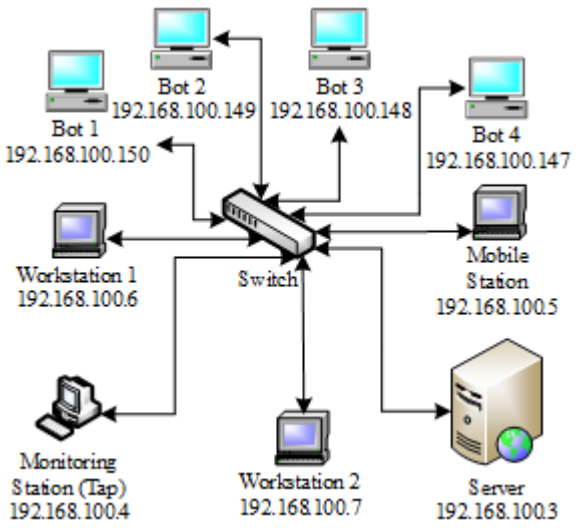


Fig 1: Network configuration for gathering Bot-IoT dataset

The Server maintains SSH, DNS, HTTP, FTP, e-mail services. It also simulates IoT related activities,

such as temperature, humidity and pressure sensing, passing information to virtual brokers by the MQTT with a repeated communication pattern every 5 seconds. Another 4 IoT services that are employed during testing are smart fridge, motionlights, automated garage door and autonomous thermostat [13]. All the traffic from these services corresponds to normal activities without any attack present.

The communication protocols and the proportion of initiated instances among them used by services during normal operation are TCP (18.3%), UDP (75.7%), ARP (4.9%), ICMP (0.09%), IPv6-ICMP (0.9%), IGMP (0.02%) and RARP (0.01%).

Simultaneously to normal traffic generation, several attack types are carried out by Bot 1 to 4. The first one is OS fingerprinting and the second one – service scanning by probing the whole range of ports from 0 to 65 535. Complete TCP connections and SYN requests are established while scanning. The third and the fourth type of attacks are Denial of Service (DoS) and Distributed Denial of Service (DDoS), depending on whether one or more of the bots are attacking the server. TCP (TCP SYN), UDP and HTTP (by POST method) floods occur in both DoS and DDoS scenarios. Information theft, comprising of data theft and keylogging as fifth and sixth major types of attack, is the last of malicious activities, realized during dataset gathering. Data theft is accomplished from Workstation 2 (Fig. 1) by the Metasploit platform to the Windows 7 workstation through SMB vulnerability and by brute force over the access credentials for the Ubuntu Server [13], further exfiltrating large volumes of data. For keylogging the same weaknesses along with an open SSH service with weak credentials in the Server allow access to the users' input.

B. Features description and statistics

Ten features are selected as most promising for attack classification in the Bot-IoT dataset by its authors [13]. The first one is *seq* – sequence number of record from the registering software (Argus), *stddev* – standard deviation of all inscriptions after compacting, *N_IN_Conn_P_SrcIP* – number of incoming connections associated with a source IP address, *min* – minimal period of time for compacted records, *state_number* – number, which describe the state of a feature, *mean* – average period of time for compacted records, *N_IN_Conn_P_DstIP* - number of incoming connections associated with a destination IP address, *drate* – packets rate from destination to source, *srate* – packets rate from source to destination, *max* – maximal period of time for a compacted records. Against each record there is a label, comprising of 2 fields (general type and sub-type) for the type of attack and in a separate field whether it is an attack or normal traffic. For the purposes of this study, only the second label which could be either 0 (normal traffic) or 1 (attack), putting it as a categorical variable and being the target value for the propose classifier.

The dataset is divided in 2 parts (2 separate files in comma-separated values format (csv)) – training set which has 2 943 817 samples, of which 370 non-attacks, and testing set with 733 705 samples, of which 107 non-attacks. In order to prioritize the features based on their significance for the classification, the following parameters are found over the training set:

- Information gain, [14]:

$$IG(TS,f) = H(TS) - \sum_{v\epsilon vals(f)} P_f(v)H(S_f(v)), \quad (1)$$

where *IG* is the information gain, *TS* – the set of training samples, *f* – particular feature, *H* – entropy, *v* – given value of a feature, $S_f(v)$ – sub-set of training samples for which selected feature *f* has a value of *v*, $P_f(v)$ – categorical probability distribution on the values *v* of a given feature *f*.

- Information gain ratio, [14]:

$$IGR(TS,f) = IG(TS,f)/IV(TS,f), \quad (2)$$

where *IGR* is the information gain ratio and *IV* is the intrinsic value, defined as:

$$IV(TS,f) = -\sum_{v\in val(f)} \frac{|\{t \in TS | val(t,f) = v\}|}{|TS|} . \log_2\left(\frac{|\{t\in TS|val(t,f)=v\}|}{|TS|}\right), \quad (3)$$

where *t* is specific example for attribute *f*.

- Gini coefficient, [14]:

$$G = \frac{2\sum_{i=1}^{n} it_i}{n\sum_{i=1}^{n} t_i} - \frac{n+1}{n}, \quad (4)$$

where *G* is the Gini coefficient over a population from any real distribution, consisting of values t_i ordered in ascending order for $i = 1 \div n$.

The resulting values for *IG*, *IV* and *G* for the complete training set are shown in Fig. 2. The most notable distinction of feature concerns *seq* and *N_IN_conn_P_SrcIP*, which are 2 orders of a magnitude lower than the rest 8 features. That is the main motivating factor in our experimentation to try

discriminating normal vs. attack traffic, not only using all 10 features, but also just 8 of them.

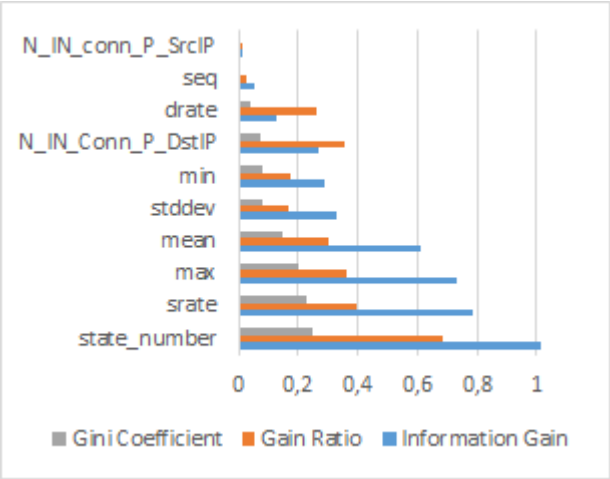


Fig 2: Features ranking

The minimal, maximal, and center values for all features is given in Table 1, along with their dispersion.

Table 1: Features' values distributions

Feature	Center	Dispersion	Min.	Max.
max	3.02	0.61	0	5
srate	3.13	250.79	0	1.10^6
drate	0.43	130.68	0	$5.9.10^4$
N_IN_Conn_P_DstIP	92.46	0.20	1	10^2
mean	2.23	0.68	0	4.98
state_number	3.13	0.38	1	11
min	1.02	1.46	0	4.98
N_IN_Conn_P_SrcIP	82.55	0.30	1	100
stddev	0.89	0.92	0	2.50
seq	$1.2.10^5$	0.62	1	$2.6.10^5$

During training and testing with the specifically designed classifiers all features are being normalized by the min-max approach [15]:

$$p_{in} = (p_i - p_{min})\frac{(\beta-\alpha)}{(p_{max}-p_{min})} + \alpha, \qquad (5)$$

where p_{in} is the normalized value for particular input value p_i of a feature, given $p_i \in [p_{min}, p_{max}]$ and $p_{in} \in [\alpha, \beta]$. In this study $\alpha = 0$ and $\beta = 1$.

The 10-bin histogram for each feature is presented in Fig. 3. It could be observed a particular similarity between the distributions of the *N_IN_Conn_P_SrcIP* and *N_IN_Conn_P_DstIP* features and as it is demonstrated below these features are also highly correlated.

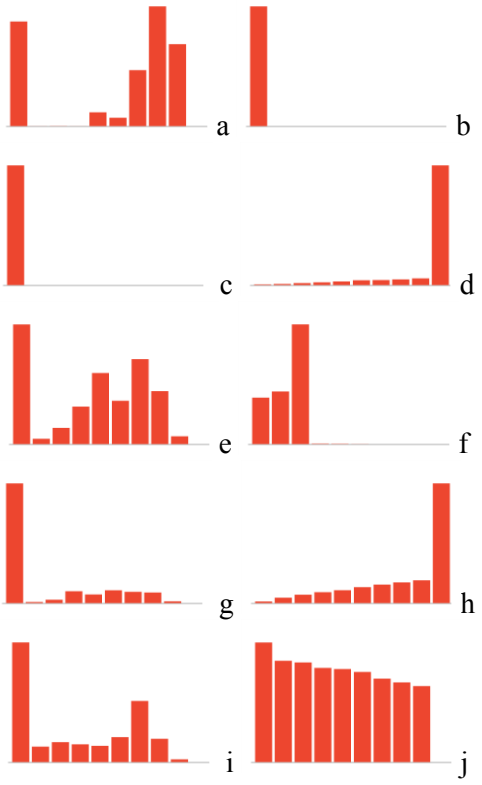


Fig 3: Features' histograms: a – max, b – srate, c- drate, d – N_IN_Conn_P_DstIP, e – mean, f – state_number, g – min, h – N_IN_Conn_P_SrcIP, i – stddev, j – seq

The correlation on a pair-wise basis for the first 20 most strongly connected features, using Pearson and Spearman coefficients, is shown in Fig.4. The Pearson coefficient is calculated according to [15]:

$$r_{fg} = \frac{\Sigma_{i=1}^{n}(f_i-\bar{f})(g_i-\bar{g})}{\sqrt{\Sigma_{i=1}^{n}(f_i-\bar{f})^2}\sqrt{\Sigma_{i=1}^{n}(g_i-\bar{g})^2}}, \qquad (6)$$

where f and g are two distinctive features from the training set with mean values $\bar{f}$ and $\bar{g}$, respectively, and n is the size of the training set. The Spearman coefficient r_s is found in a similar way to (6) but taking in to account the rank variables [15]:

$$r_s = \frac{cov(rk_f, rk_g)}{\sigma_{rk_f}\sigma_{rk_g}}, \qquad (7)$$

where rk_f and rk_g are the ranks of the features' raw values f and g, and σ denotes standard deviation of the respective parameter.

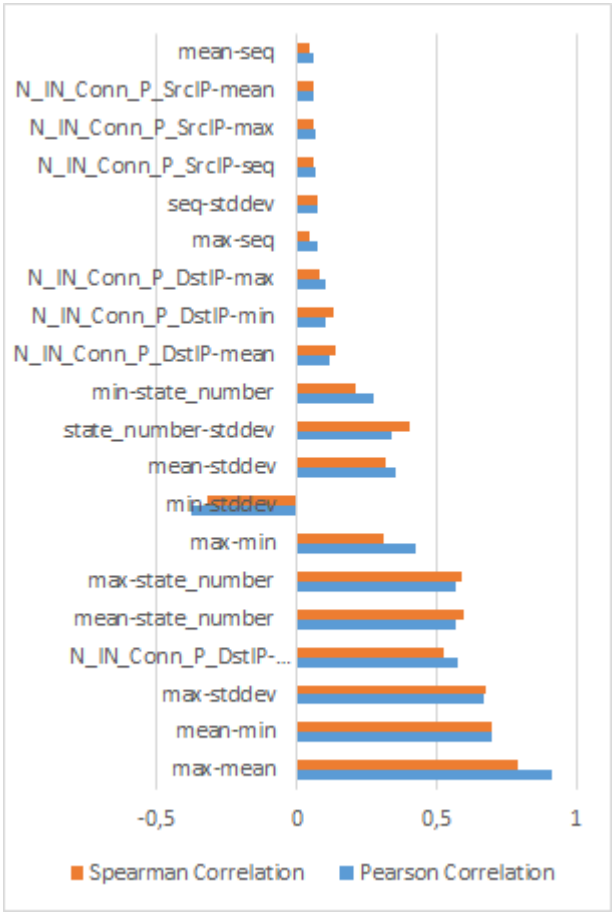


Fig 4: Pair-wise features correlation

The successive number of record *seq* is met 4 times in the first 20 correlation dependencies, which is also true for the *N_IN_Conn_P_SrcIP*. Both features form strong connections with the *N_IN_Conn_P_DstIP* feature. This is another reason to try testing only 8 features with the proposed classifier – without *seq* and *N_IN_Conn_P_SrcIP* - as a second comparing experiment to that of using all 10 features.

C. Proposed neural classifier general structure

One of the main goals of the presented study is to propose a simpler structure of a classifier to others, such as the Recurrent Neural Network, described in [13] for the same task of discriminating attacks from normal activities by traffic monitoring. This allows both the time execution during training and classification and the memory consumption to be lower. The general structure of the classifier, developed here, is presented in Fig. 5. It is feedforward neural network with backpropagation algorithm for training [16].

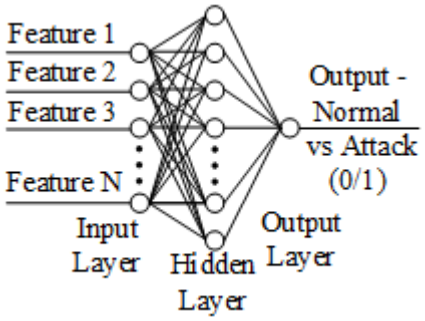


Fig 5: Proposed neural network for classification

All the features, employed for classification, form a vector $\vec{f} = \{f_1, f_2, \dots, f_{N_1}\}$, where $N_1 = 10$ for the complete set, described in Section 2.2. The weights of a neuron from layer l_k, where $k = 1 \div 3$, that is an input layer, a hidden one and an output layer, are $\overrightarrow{w_i^k} = \{w_{1i}^k, \dots, w_{N_k i}^k\}$, leading to outputs of all neurons from the same layer l_k, ordered as a vector in the form $\overrightarrow{o^k} = \{o_1^k, \dots, o_{N_k}^k\}$. One and the same type of activation function g_k, $k = 1 \div 3$, is used among all neurons from a particular layer, although activation functions from different layers are different, described in Section 3 within the experimental results. Before training the network is fully connected and it is initialized with the first set of features, passed from the training set $\overrightarrow{f_1}$. The output of any neuron can be expressed in the following generalized form [16]:

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

where h_i^k is the product of the input signals for a neuron i from layer k with its associated weights, b_i^k – the bias of the same neuron.

The training process involves the set of pairs $F = \{(\overrightarrow{f_1}, t_1), \dots, (\overrightarrow{f_M}, t_M)\}$, where for each feature vector $\overrightarrow{f_i}$ for $i = 1 \div M$, there is expected value, or target, as an output t_i. Cost function is the Mean Squared Error (*MSE*), [16]:

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

where E is the expectation operator, applied over all realizations during training, which explicit aim is minimizing *MSE*. It is achieved by iterative changes of both weights and biases, according to [16]:

$$\Delta w_{ij}^k = -\alpha \frac{\partial E(F)}{\partial w_{ij}^k}, \quad (10)$$

$$\Delta b_i^k = -\alpha \frac{\partial E(F)}{\partial b_i^k}, \quad (11)$$

where α is the learning rate.

D. Classifier optimization algorithm

There is no general approach that could indicate how to select the number of neurons in the hidden layer N_1 of the network, described in Section 2.3. That's why the following general scheme (Fig. 6) for their selection is applied during experimentation.

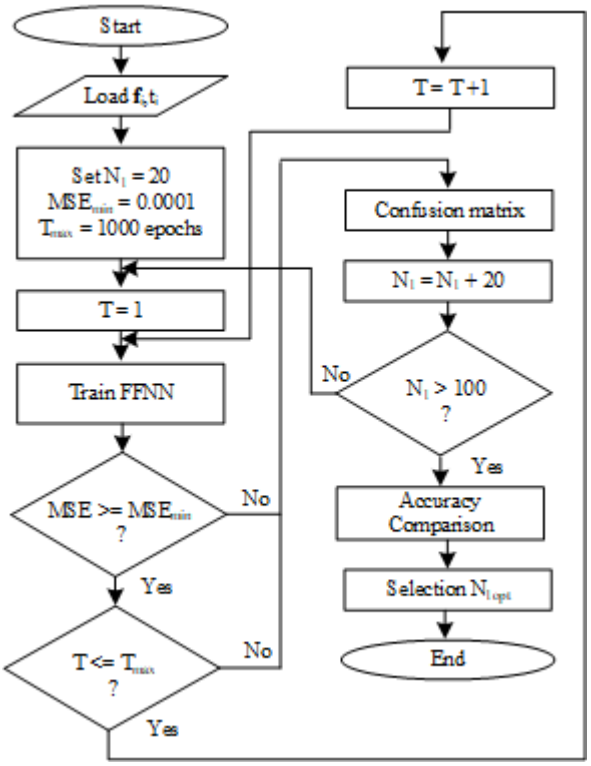


Fig 6: Neural network optimization

Five realizations of the network are tried with hidden neurons varying from 20 to 100 with a step of 20. Given target values for minimal MSE and maximum number of epochs, in this case 1000, training continues until one of the criteria is met. From all 5 tested networks accuracy is compared, based on confusion matrices over the results of classification and then the optimal N_1 is selected. This experiment is done for 4 different activation functions for the hidden neurons – Identity, Logistic, Tanh, and ReLU. Optimization is implemented with the full set of 10 features for not losing generalization. Also, 2 algorithms for backpropagation are tested – the Adam optimization and the Scaled Gradient Descent (SGD).

In order to evaluate the performance of the trained classifiers, the following measures are used [17]:

- True Positives – *TP* – the number of samples, that are correctly classified as either attack or non-attack to the corresponding class during testing;
- True Negatives – *TN* – the number of samples, that are correctly not associated with particular class;
- False Positives – *FP* – the number of samples, that are incorrectly assigned to given class but do not belong to it;
- False Negatives – *FN* – the number of samples, that actually belong to given class, but are wrongly associated to the other class;
- *Precision* = *TP* / (*TP* + *FP*);
- *Recall* = *TP* / (*TP* + *FN*);
- *Specificity* = *TN* / (*TN* + *FP*);
- *F1* = *TP* / (*TP* + (*FP* + *FN*)/2);
- *Log-loss* = -[*t*.ln(*p*) + (1-*t*).ln(1-*p*)], where *t* is the target value of a sample and *p* is the probability, found by the classifier that the very same sample belongs to the class, associated with the target. This parameter is found as average value over all samples;
- Classification accuracy – *CA* = (*TP* + *TN*) / (*TP* + *TN* + *FP* + *FN*);
- Area Under Curve – *AUC* – the probability that the classifier ranks arbitrary selected positive sample higher than an arbitrary selected negative sample.

III. EXPERIMENTAL RESULTS

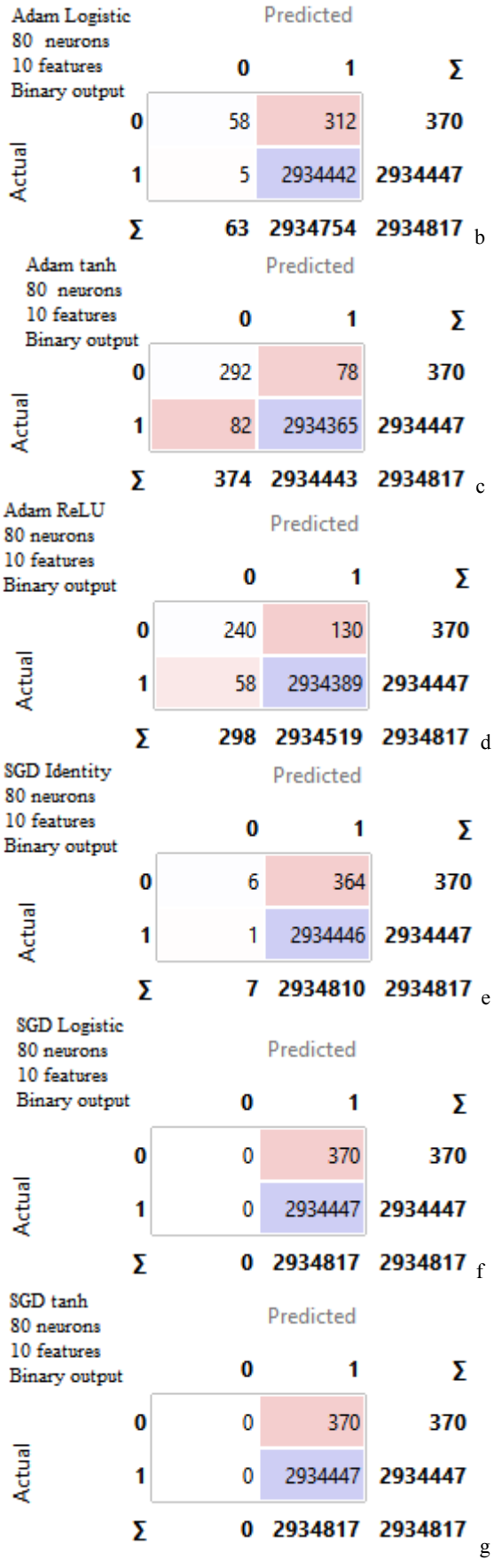
The hardware platform, used for testing, consists of Intel Xeon E5-1620 CPU with 4 cores, operating in hyper-threading mode at 3.50 GHz. The processor holds L1 cache of 256 kB, L2 – 1 MB and L3 – 10 MB. The size of RAM memory is 64 GB, and the size of the HDD is 2 TB. All components are under the control of MS Windows 10 Pro 20H2. Orange v. 3.28 is the simulation environment, in which all neural network models are trained and tested.

The results from finding the optimal network configuration are given in Fig. 7.

Adam Identity
80 neurons
10 features
Binary output

Actual \ Predicted	0	1	Σ
0	9	361	370
1	0	2934447	2934447
Σ	9	2934808	2934817

a



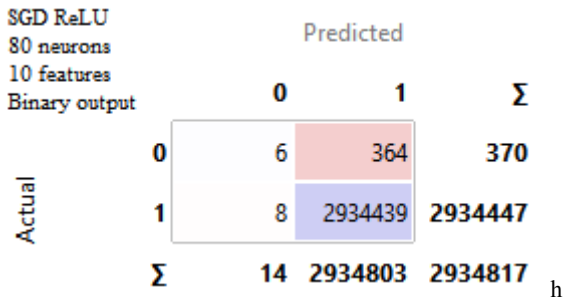


Fig 7: Optimizing the structure of the neural network by training algorithm and activation function

The results in Fig. 7 show confusion matrices only for the case of 80 neurons. All values are obtained after validation of the full training set. Similar are the result as ratios between actual and predicted numbers of attacks and normal network communication.

Execution times for both solvers and all activation functions are given in Table 2.

Table 2: Execution times achieved by SGD and Adam optimization using 80 neurons in the hidden layer

Algorithm	Activation Function	Training Time, sec	Testing Time, sec	
			Train set	Test set
SGD	Identity	138.79	2.59	0.95
	Logistic	159.50	7.86	1.97
	Tanh	243.18	7.82	2.19
	ReLU	228.59	5.93	1.59
Adam	Identity	145.88	3.01	0.95
	Logistic	179.65	8.05	9.17
	Tanh	212.47	11.53	3.36
	ReLU	189.19	4.61	1.89

Adam optimization turns out to be much more efficient than SGD in this binary classification problem (Fig. 7). The best accuracy is found for the Tanh activation function which is further tested (Tables 3 and 4).

Table 3: Classifier performance at 10 features, validating the training set

Cl.	N_1	AUC	CA	F1	Pre-cision	Recall	Log-loss	Specifi-city
0	20	0.9999	0.9999	0.8198	0.7929	0.8486	0.0001	0.9999
	40	0.9999	0.9999	0.7789	0.8498	0.7189	0.0001	0.9999
	60	0.9999	0.9999	0.8403	0.7559	0.9459	0.0001	0.9999
	80	0.9999	0.9999	0.7849	0.7807	0.7892	0.0001	0.9999
	100	0.9997	0.9999	0.8068	0.8503	0.7676	0.0001	0.9999
1	20	0.9999	0.9999	0.9999	0.9999	0.9999	0.0001	0.8486
	40	0.9999	0.9999	0.9999	0.9999	0.9999	0.0001	0.7189
	60	0.9999	0.9999	0.9999	0.9999	0.9999	0.0001	0.9459
	80	0.9999	0.9999	0.9999	0.9999	0.9999	0.0001	0.7892
	100	0.9997	0.9999	0.9999	0.9999	0.9999	0.0001	0.7676

Table 4: Classifier performance at 10 features, using the test set

Cl.	N_1	AUC	CA	F1	Pre-cision	Recall	Log-loss	Specifi-city
0	20	0.9999	0.9999	0.8148	0.8073	0.8224	0.0001	0.9999
	40	0.9999	0.9999	0.7539	0.8571	0.6729	0.0001	0.9999
	60	0.9999	0.9999	0.8448	0.7840	0.9159	0.0001	0.9999
	80	0.9999	0.9999	0.7714	0.7864	0.7570	0.0001	0.9999
	100	0.9998	0.9999	0.7960	0.8511	0.7477	0.0002	0.9999
1	20	0.9999	0.9999	0.9999	0.9999	0.9999	0.0001	0.8224
	40	0.9999	0.9999	0.9999	0.9999	0.9999	0.0001	0.6729
	60	0.9999	0.9999	0.9999	0.9999	0.9999	0.0001	0.9159
	80	0.9999	0.9999	0.9999	0.9999	0.9999	0.0001	0.7570
	100	0.9998	0.9999	0.9999	0.9999	0.9999	0.0001	0.7477

Results from training and testing the neural network with the Adam algorithm and Tanh when using 10 features and varying the number of neurons are given on a class (Cl.) basis.

The evaluation parameters from validation of the complete training set with the classifier when using 8 features, that is without *seq* and *N_IN_Conn_P_SrcIP*, are gathered in Table 5.

Table 5: Classifier performance at 8 features, using the training set

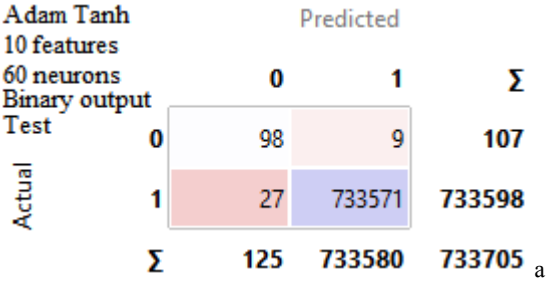
Cl.	N_1	AUC	CA	F1	Pre-cision	Recall	Log-loss	Specifi-city
0	20	0.9997	0.9999	0.6924	0.7356	0.6541	0.0002	0.9999
	40	0.9997	0.9999	0.6396	0.8008	0.5324	0.0002	0.9999
	60	0.9997	0.9998	0.4396	0.6818	0.3243	0.0002	0.9999
	80	0.9998	0.9999	0.7411	0.7212	0.7622	0.0002	0.9999
	100	0.9998	0.9999	0.7085	0.7689	0.6567	0.0002	0.9999
1	20	0.9997	0.9999	0.9999	0.9999	0.9999	0.0002	0.6541
	40	0.9998	0.9999	0.9999	0.9999	0.9999	0.0002	0.5324
	60	0.9997	0.9998	0.9999	0.9999	0.9999	0.0002	0.3243
	80	0.9998	0.9999	0.9999	0.9999	0.9999	0.0002	0.7622
	100	0.9998	0.9999	0.9999	0.9999	0.9999	0.0002	0.6567

Testing the performance of the classifier over the whole test set with the same 8 features leads to the parameters from Table 6.

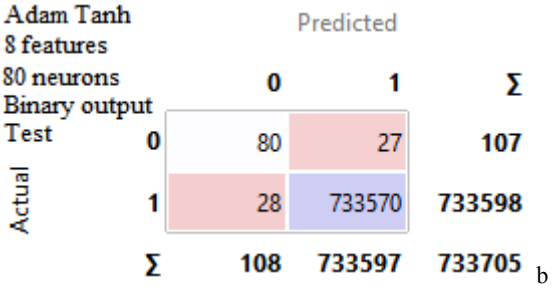
Table 6: Classifier performance at 8 features, using the test set

Cl.	N_1	AUC	CA	F1	Pre-cision	Recall	Log-loss	Specifi-city
0	20	0.9999	0.9999	0.6767	0.7362	0.6261	0.0002	0.9999
	40	0.9999	0.9999	0.6321	0.8208	0.5140	0.0002	0.9999
	60	0.9999	0.9998	0.4968	0.7407	0.3738	0.0002	0.9999
	80	0.9999	0.9999	0.7441	0.7407	0.7476	0.0002	0.9999
	100	0.9999	0.9999	0.7357	0.8255	0.6635	0.0002	0.9999
1	20	0.9999	0.9999	0.9999	0.9999	0.9999	0.0002	0.6261
	40	0.9999	0.9999	0.9999	0.9999	0.9999	0.0002	0.5140
	60	0.9999	0.9998	0.9999	0.9999	0.9999	0.0002	0.3738
	80	0.9999	0.9999	0.9999	0.9999	0.9999	0.0002	0.7476
	100	0.9999	0.9999	0.9999	0.9999	0.9999	0.0002	0.6635

Confusion matrices when processing the full test set with the optimal classifier, which turns to have N_{1opt} = 60 neurons for 8 features and N_{1opt} = 80 neurons for 10 features, are given in Fig. 8. These optimal configurations are obtained based on the minimal number of wrongly marked attacks as non-attacks, while the number of missed attacks is negligible in virtually all cases (Fig. 7 and 8, and Tables 4 and 6). Both classifiers use the Adam optimization with Tanh activation function for neurons in the hidden layer. They are trained either to reaching MSE_{min}, according to the algorithm from Fig. 6, or T = 1000 epochs. Learning rate in all cases is α = 0.0001.



a



b

Fig 8: Confusion matrices for the optimal classifier at 10 (a) and 8 (b) features

The time consumption during training with all the samples from the training set with 8 features and the time needed for classification with both the training and testing sets are given in Table 7.

Table 7. Execution times, using 8 features, in sec

N_1	Training	Testing	
		Train set	Test set
20	110.62	3.93	0.99
40	152.39	6.71	2.04
60	194.45	8.71	2.20
80	229.59	10.84	2.73
100	238.87	13.36	3.22

The respective execution times when using 10 features are presented in Table 8.

Table 8. Execution times, using 10 features, in sec

N_1	Training	Testing	
		Train set	Test set
20	121.24	4.23	1.06
40	131.88	6.96	1.83
60	200.79	9.51	2.45
80	212.47	11.53	3.36
100	243.13	14.15	3.42

IV. DISCUSSION

SGD algorithm leads to low classification accuracy of the non-attack samples – 1.62%, employing the Identity and ReLU activation functions for the hidden neurons and 0% - for the Logistic and Tanh (Fig. 7.e-h). In the same time the Adam algorithm allows to have 2.43% for the Identity activation function, 15.68% - for the Logistic, 78.92% - for the Tanh, and 64.86% - for the ReLU correctly discriminated non-attacks (Fig. 7.a-d). The level of correct classification of all kinds of attacks is high for all 8 initial tests, involving 80 neurons, for both SGD and Adam algorithms – 2.79.10-3% error for the worst case of Adam ReLU (Fig. 7.c). The execution times, registered during the training process, are smaller when the Adam optimization uses Tanh and ReLU activation functions, compared to SGD with the same functions – by a factor of 0.85 on average (Table 2).

SGD with a combination of Identity or Logistic activation function is faster than the Adam algorithm during training – around 1.09 times. The test phase takes almost the same time between each pair of classifiers, using one and the same activation function, no matter of the training algorithm used. The exception of 9.17 sec for the Logistic case is considered to be an outlier, coming as an exception from experimentation. Among all 4 activation functions the most time for testing occurs for the Tanh and the least – for the Identity function with a difference of around 2.5 times. Given the classification accuracy, the combination of Adam algorithm and Tanh activation function are selected for all further tests.

Varying the number of hidden neurons from 20 to 100 when using all 10 features leads to high average classification accuracy of 0.9999 for both attack and non-attack samples (Table 3 and 4). However, the F1 score reveals a particular drop for the non-attack class (Cl. = 0) with least value of 0.7539 for 40 neurons trying the test set (Table 4). The same measure is constantly bounding to 0.9999 in all cases for Cl. = 1, which is also true for the Precision and Recall measures, while Logloss is 0.0001. In the same time Precision and Recall drop to 0.7840 and 0.6729 for 60 and 40 hidden neurons, respectively for Cl. = 0. Specificity stays constantly high, equal to 0.9999, for the non-attacks but goes down to 0.6729 for 40 neurons for the attack samples. The optimal configuration of a binary classifier, working on 10 features, includes 60 hidden neurons, being trained with the Adam optimization algorithm and using Tanh activation function in the hidden layer. Only 8.4% from all non-attacks are misclassified in this case along with 3.68.10-3% missclassifications for the attack samples (Fig. 8.a).

Reducing the feature set to 8 parameters preserves the classification accuracy to 0.9999 in most of the cases (Tables 5 and 6) with the exception of 60 hidden neurons, where it is slightly lower – 0.9998. The area under the curve (AUC) is also high – dominantly 0.9999, but the F1 measure drops from around 0.2 to 0.4 with the change of the number of hidden neurons. Its lowest point is for 60 neurons – 0.4396 for Cl. = 0 and for Cl. = 1 – it is always 0.9999. Similar to the case of 10 features, classification of attacks is very accurate, which is also seen from the Precision and Recall, being equal to 0.9999. The Logloss is 0.0002 for both attack and non-attack samples, but only attack samples yields Specificity of 0.9999. Cl. = 1 testing reveals Specificity of 0.3243 for 60 hidden neurons when validating the classifier over the full train set and 0.3738 – over the test set as the minimum for this parameter. The best result is obtained for 80 neurons with 74.77% correctly classified non-attacks and 3.82.10-3% misclassified attacks (Fig. 8.b). Again, Adam optimization when using Tanh activation function for the hidden layer gives these optimal results. Obviously, this classifier with reduced feature set is more inaccurate than the 10-feature implementation with 16.83% less with regard to non-attacks, raising more false alarms. Given the execution times (Tables 7 and 8) at N1 = 60 for 10 features and N1 = 80 for 8 features, which are very close with an absolute difference of 0.28 sec (3.82.10-7 sec/sample), making the first one faster, it is natural to select it as the optimal version of a classifier and further use it in practical implementations.

Finally, a comparison is made of both the 8- and 10-feature classifiers, proposed in this study, with a Recurring Neural Network (RNN), described in [1], operating over the same training and testing set. As it can be seen from Table 9, the Recall, F1 measure and CA are higher for the two implementations of the feed forward neural network. Precision is lower by 0.0001 for the 8-feature variant, while the 10-feature produce the same result as the RNN. Given the more complex structure of the RNN, with 20, 60, 80 and 90 neurons in 4 hidden layers, and the overall achieved accuracy, the proposed optimal classifier from the current research would be preferable to use for detecting DoS, DDoS, theft and reconnaissance type of attacks.

Table 9. Classification efficiency comparison on average parameters for both classes

Parameter	RNN - 10 features, [1]	Proposed - 10 features	Proposed - 8 features
Precision	0.9999	0.9999	0.9998
Recall	0.9975	0.9999	0.9998
F1	0.7338	0.9999	0.9998
CA	0.9974	0.9999	0.9998

V. CONCLUSION

In this paper a new model of feedforward neural network with one hidden layer is proposed, capable of detecting DoS, DDoS, theft and reconnaissance activities over IP packet switching networks, aimed at particular victim machines. The optimal structure of the classifier consists of 10 inputs, 60 hidden neurons using hyperbolic tangent activation function and 1 single output for binary discrimination to attack vs. non-attack activity. Most accurate result from training is achieved by the Adam optimization procedure. Analogous neural network model, incorporating only 8 from the 10 features with 80

hidden neurons, behaves also accurately enough and together with the 10-feature implementation achieve better accuracy than a 4 hidden layers Recurring Neural Network over the same test base. The proposed classifier is considered promising for practical implementation as a component of an online monitoring system, mainly against DDoS attacks in packet switched networks.

REFERENCES

[1] Sharma, K., Mukhopadhyay, A., Cyber Risk Assessment and Mitigation Using Logit and Probit Models for DDoS attacks, *Americas Conference on Information systems (AMCIS 2020)*, August 2020, pp. 1-9.

[2] Vishwakarma, R., Jain, A. K., A Survey of DDoS Attacking Techniques and Defence Mechanisms in the IoT Network. *Telecommunication Systems*, Vol. 73, No. 1, 2020, pp. 3-25.

[3] Sallam, A. A., Kabir, M. N., Alginahi, Y. M., Jamal, A., Esmeel, T. K., IDS for Improving DDoS Attack Recognition Based on Attack Profiles and Network Traffic Features. *In 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA)*, IEEE, February 2020, pp. 255-260.

[4] Tahsien, S. M., Karimipour, H., Spachos, P., Machine Learning based Solutions for Security of Internet of Things (IoT): A Survey. *Journal of Network and Computer Applications*, Vol. 161, 2020, art. No. 102630.

[5] Chen, Y. W., Sheu, J. P., Kuo, Y. C., Van Cuong, N., Design and Implementation of IoT DDoS Attacks Detection System based on Machine Learning. *In 2020 European Conference on Networks and Communications (EuCNC)*, IEEE, June 2020, pp. 122-127.

[6] Jia, Y., Zhong, F., Alrawais, A., Gong, B., Cheng, X., Flowguard: An Intelligent Edge Defense Mechanism against IoT DDoS Attacks, *IEEE Internet of Things Journal*, Vol. 7, No. 10, pp. 9552-9562.

[7] Galeano-Brajones, J., Carmona-Murillo, J., Valenzuela-Valdés, J. F., Luna-Valero, F., Detection and Mitigation of DoS and DDoS Attacks in IoT-based Stateful SDN: An Experimental Approach, *Sensors*, Vol. 20, No. 3, 2020, Art. No. 816.

[8] Bhardwaj, K., Miranda, J. C., Gavrilovska, A., Towards IoT-DDoS Prevention using Edge Computing, *In {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 18)*, Boston, MA, USA, July 2018, pp. 1-7.

[9] Doshi, R., Apthorpe, N., Feamster, N., Machine Learning DDoS Detection for Consumer Internet of Things Devices, *In 2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, May 2018, pp. 29-35.

[10] Yin, D., Zhang, L., Yang, K., A DDoS Attack Detection and Mitigation with Software-Defined Internet of Things Framework, *IEEE Access*, Vol. 6, 2018, pp. 24694-24705.

[11] Roopak, M., Tian, G. Y., Chambers, J., An Intrusion Detection System against DDoS Attacks in IoT Networks, *In 2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, January 2020, pp. 0562-0567.

[12] Mehmood, A., Mukherjee, M., Ahmed, S. H., Song, H., Malik, K. M., NBC-MAIDS: Naïve Bayesian Classification Technique in Multi-Agent System-Enriched IDS for Securing IoT against DDoS Attacks, *The Journal of Supercomputing*, Vol. 74, No. 10, 2018, pp. 5156-5170.

[13] 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.

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

[15] Abu-Bader, S. H., *Using statistical Methods in Social Science Research: With a Complete SPSS Guide*, Oxford University Press, 2021.

[16] Elgendy, M., *Deep Learning for Vision Systems*, Manning Publications, 2020.

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