Journal of Intelligent & Fuzzy Systems xx (20xx) x–xx
DOI:10.3233/JIFS-190159
IOS Press

Distributed denial of service attack detection using autoencoder and deep neural networks

Ferhat Ozgur Catak[a,*] and Ahmet Fatih Mustacoglu[b]
[a]*TUBITAK-BILGEM Cyber Security Institute, Kocaeli, Turkey*
[b]*Istanbul Sehir University, Cyber Security Engineering, Istanbul, Turkey*

Abstract. Today, many companies are faced with the huge network traffics mainly consisting of the various type of network attacks due to the increased usage of the botnet, fuzzier, shellcode or network related vulnerabilities. These types of attacks are having a negative impact on the organization because they block the day-to-day operations. By using the classification models, the attacks could be identified and separated earlier. The Distributed Denial of Service Attacks (DDoS) primarily focus on preventing or reducing the availability of a service to innocent users. In this research, we focused primarily on the classification of network traffics based on the deep learning methods and technologies for network flow models. In order to increase the classification performance of a model that is based on the deep neural networks has been used. The model used in this research for the classification of network traffics evaluated and the related metrics showing the classification performance have been depicted in the figures and tables. As the results indicate, the proposed model can perform well enough for detecting DDoS attacks through deep learning technologies.

Keywords: cyber security, ddos, deep learning, autoencoder

1. Introduction

The Internet providers and the companies are vulnerable to the various type of cyber-attacks such as botnet, DDoS, back door, exploits, shell code and fuzzier etc. [1, 2]. The network traffic exposed to cyber-attacks will cause negative effects on the networks preventing the legal users having the requested services [3, 4].

Firewalls and intrusion detection systems (IDS) are commonly used for preventing various types of network attacks. These tools are generally using signature-based security mechanisms and they are not interoperable with the new network packet models or malicious network traffic changes. It is crucial to detect malicious network traffics in real time [5–7]. In DDoS attacks, attackers use the compromised computers that are hacked earlier to send huge network traffics to the targeted server. Such abnormal changes could be detected using descriptive statistical methods. Feinstein et al. [8] use Chi-Square statistics to identify network flow anomalies. They proposed time window based entropy changes in network flow to detect malicious traffic.

Descriptive statistics based detection methods have relied on prior know data. A noticeable difficulty is that network flow anomalies are a timely changing target [9]. It is crucial to accurately characterize the set of malicious network traffic. A new type of malicious traffic will continue to appear over time. As a result, a malicious network classification model must avoid the over-fit to any predefined set of malicious traffic classes.

The contributions of this work are to detect network attacks using windowing based training samples by means of machine learning algorithms. We implemented Autoencoder based deep neural network algorithm to separate normal network traffic from

*Corresponding author. Ferhat Ozgur Catak, TUBITAK-BIL GEM Cyber Security Institute, 41400 Kocaeli, Turkey. E-mail: ozgur.catak@tubitak.gov.tr.

the attacks, including Analysis, Fuzzers, Generic, DoS, Backdoors, Exploits, Shellcode, Worms, and Reconnaissance. The main objective of the proposed model is to use a hybrid model approach for exact classification of malicious network flow from packets. The autoencoder layer of the model learns the representation of the network flows. The second layer (deep neural network model) tries to find out the exact malicious activity class. Our proposed model does not rely on signature-based detection or deep packet inspection methods. Both methods are easily beating by attackers using encryption or obfuscation methods.

In summary, the proposed DDoS detection model makes the following contributions:

- We present a novel hybrid malicious network flow detection technique based on Autoencoder and deep neural networks. Hence, our proposed model can avoid overfitting to pre-defined malicious patterns.
- We used different activation function and the number of hidden layer units to achieve best results in both time and model performance.
- The proposed model is very practical. It relies on sampled network flow data.
- We describe the implementation of the proposed detection system. We use large volumes of packet capture (PCAP) of network files that publicly available on the Internet for benchmarking of malicious network activity detection models.
- In our previous research [12], we applied only a deep neural network model to detect malicious activities. In this paper, we also applied the autoencoder algorithm to represent the input data structure.

The rest of the paper is organized as follows: Section 2 briefly introduces some of the earlier works related to our problem. Section 3 describes the dataset, basic concepts and performance metrics to evaluate the proposed detection model. Section 4 explains our proposed model. Section 5 evaluates the proposed the learning model. Section 6 concludes this paper.

2. Related work

In this section, we describe the general overview of the literature review.

2.1. Literature review overview

Machine learning algorithms based DDoS detection systems have been studied by many researchers. Many different statistical learning or predictive learning solutions over network flow data have been proposed recently. Some popular different learning based solutions to build a predictive model in the literature include the following.

Various types of malicious changes in network traffic could be identified by using statistical methods. Feinstein et al. [8] suggested the usage of the χ-square statistical method to define anomalies on network traffic. They offer to use the entropy changes based on the time frames on the network flow to detect malicious network activities. There exist some research works for detecting DDoS attacks by using various machine learning methods.

Bhatia [13] mentions that DDoS attacks should be evaluated based on the targeted network layer. SYN floods [14] attacks are invented to flood a network interface with malicious traffic in order to crush its resource and deny to answer to legitimate traffic. Furthermore, attacks on the application level can also aim to spent network bandwidth and CPU resources. The criteria that are used for identifying the attacks on the network layer will be different from the ones used for identifying attacks on the application level. So, different types of models have been used for detecting both types of attacks.

Another research work [15] experiences on using the different type of machine learning methods in order to detect SMURF and teardrop attacks for denial of service through the network. They used naive-Bayes, Bayesian networks, decision tree and decreasing fault deduction (REPTree) algorithms, and they also used the kddcup99 dataset. They build a classification model that is composed of running two or more algorithms as ensemble methods. The researchers in this work obtain the correct results in the ratio of %99.94117 by using kddcup99 dataset.

Osanaiye et al. [16] used the ensemble methods for selecting attributes. Information gain, gain ratio, χ-square and ReliefF methods have been tested on the NSL-KDD dataset that is an enhanced version of kddcup99 dataset. They suggested a model and used the model to select/collect 13 attributes. Then, they trained the selected attributes by using the decision tree algorithm and they measured the correctness of the classification model as %99.67.

Livadas et al. suggested a network flow based approach for detecting the command and control

traffic of the IRC based botnets [17]. In this work, they have first classified traffic flow into IRC Chat or IRC non-chat groups by using classification algorithms. Next, they have classified IRC information flows into malicious or normal activities. They have reached the %10–20 false-negative rate and %30–40 false-positive rate for the proposed detection model.

Zhao et al. suggested the botnet detection system based on the analysis of the traffic behavior and the flow time-frame [18]. They have used the Reduced Error Pruning algorithm (REPTree) in order to classify the network traffic as a malicious or a non-malicious activity.

Bojović et al. [11] proposed a hybrid method for the detection of DDoS attacks that combines feature-based and volume-based detection. In this work, they applied the exponential moving average algorithm for decision-making, applied to both entropy and packet number time series. Their proposed method uses the diversity of the source IP addresses that the packets come from.

Defining statistical based detection methods are based on the previous knowledge of network flow. One of the obvious difficulty is that the malicious network flows are becoming a changing target when the time passes [9]. However, it is a very difficult task to characterize the network traffic correctly. Since the new malicious network traffic will be moving forward in changing when the time passed. But, the classification model should be restricted in order to prevent the over fit from the predefined malicious network traffic [19].

2.2. *The differences between proposed model and literature review*

The main differences are:

- In [8], they applied statistical learning methods which are different from predictive modeling. Authors have proposed a method of detecting attacks using an entropy [20] change over the network flow. In the proposed model in our paper, we applied predictive modeling using deep neural networks and autoencoder.
- In [13], they detect only syn_flood [21] DDoS attack types. Syn_flood attack is layer 4 type attack that works at the transport protocol (TCP) layer. Our proposed model detects both layer 4 and layer 7 type attacks.
- In [15], they detect smurf and teardrop attack types [22]. The smurf attack exploits the Internet protocol (IP) broadcast addressing for a DDoS attack. Teardrop attack exploits the reassembly of IP packets using fragmented packets. Both smurf and teardrop attack occurs at network layer 4 that is different from our detection model.
- In [16], authors used ensemble methods for selecting attributes. Their model uses KDD-CUP'99 based at its training phase. Decision tree algorithm is applied to build the model. The KDDCUP'99 dataset does not contain up-to-date attacks because it is a rather old data set. Our data set has more network flows and it contains are up-to-date attacks.
- In [17], they build a detection model for IRC based botnets [23]. The botmasters are able to command and control the computers in their network by means of Internet relay chat (IRC) channels. IRC network traffic occurs at layer 7. Again, our proposed model detects both layer 4 and layer 7 type attacks.
- In [18], authors build a detection model based on very small time windows to identify existing and unknown botnet's activity.
- In [11], they detect different types of DDoS attack using Shannon entropy. But they have only one feature (source IP address) to create the detection model. Attackers can manipulate source IP address easily using some tools like scapy, hping etc. As a result, source IP address diversity feature is not a reliable source to detect DDoS attacks.

3. Methodology

3.1. *The dataset*

The most commonly used dataset for IDS systems is KDDCUP99. This data set, which has been used for many years, has been used for many benchmark operations and performance evaluation of classification algorithms. The KDDCUP99 database is prepared in a laboratory environment and includes several types of attacks. However, the attacks in this data set belong to 1999 and the data set does not include current attacks. Therefore, we did not want to use the KDD-CUP99 dataset in this study. Instead, we used a new dataset that included more recent attacks.

The dataset that has been used in this research has been obtained from Cyber Security Laboratory located at Australian Cyber Security Center [24]. IXIA PerfectStorm tool is used for aggregating data, also normal and malicious network traffic are saved

in PCAP format. The grouped data set contains 9 different types of tagged data including Normal, Analysis, Fuzzers, Backdoors, Exploits, DoS, Reconnaissance, Shellcode and Worm. The Argus tool is also used for extracting 49 different properties from the PCAP file, and these 49 properties also include nominal values such as IP address, protocol, etc. In this research, we have to use the continuous features due to the nature of the classification algorithm that we have used. Hence, 25 continuous features out of 49 have been selected in order to be used in this research. The selected features and their descriptions are given in Table 1.

Table 2 shows the distribution of the normal and malicious activities of the dataset.

Table 1
Feature list

Feature Name	Description
ackdat	TCP connection initialization time between SYN_ACK and ACK packets
ct_flw_http_mthd	Number of flows in http services such as Get and Post, etc.
ct_ftp_cmd	Number of flows in ftp session
dbytes	Size of the target operation
dinpkt	Arriving time between target layers (ms)
djit	Target jitter (ms)
dload	Targeted bit/sec
dloss	The number of target packets that are delegated or deleted
dmean	The mean of the raw packet delivered by the target size
dpkts	The number of target packet
dur	Total time
dwin	Target TCP frame value
is_ftp_login	FTP Session
response_body_len	The size of the response by the http server
sbytes	The size of the source operation
sinpkt	Arriving time between source layers (ms)
sjit	Source jitter (ms)
sload	Source bit/sec
sloss	The number of source packets that are delegated or deleted
smean	The mean of the raw packet delivered by the source size
spkts	The number of source packets
swin	Source TCP frame value
synack	TCP connection initialization time between SYN and SYN_ACK packets
tcprtt	The round trip time for TCP connection. Sum of the SYN_ACK and ACKDAT.
trans_depth	The length/level of the http request/response transaction

Table 2
Label distribution

Traffic	Total Records
Normal	37,000
Malicious/Attack	45,332

3.2. Basic concepts

The IP flow architecture is shown in Section 3.2.1. The basic idea of autoencoder technique is introduced in Section 3.2.2.

3.2.1. Network flow

The Network flow is a combined view of sequences of packets exchanged between a destination and source hosts. A network flow can be identified by a five-tuple: *srcIP*, *dstIP*, *srcPort*, *dstPort*, *Protocol*. We cannot determine whether an IP packet is malicious or not by just inspecting a single packet. It is better to group packets according to the five-tuple. Then, inspecting the grouped packets will give better results for their activities.

3.2.2. Auto encoder

Autoencoders are an unsupervised symmetrical neural network type that compresses multidimensional input data in hidden space and then reconstructs the data from the compressed hidden space. A typical simple autoencoder is presented in Fig. 1, but more complex architectures can be used. Fig. 1 shows a typical autoencoder with one input layer of n units, 3 hidden layers with different units, one reconstruction layer with n units, and one activation function f.

f is an activation function that can be any nonlinear function. In this work, our activation function f is a sigmoid function, which can be formulated as:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

The layer between the input layer and the hidden layer (Latent Space Representation) is called the encoding layer. The encoding layer allows multidimensional data to be reduced to a smaller size. The layer between the hidden space layer and the output layer is called as the decoding layer. Decoder tries to reconstruct the input by increasing the size of the compressed hidden space layer.

Let $\mathcal{X}$ denotes the set of input instances $\mathbf{x}$. The autoencoder tries to minimize data reconstruction error. The traditional squared error is often used as the loss function of the autoencoder.

$$L(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} ||\mathbf{x} - \hat{\mathbf{x}}||^2 \tag{2}$$

$\hat{\mathbf{x}}$ is the reconstruction of input instance $\mathbf{x}$ given by the autoencoder.

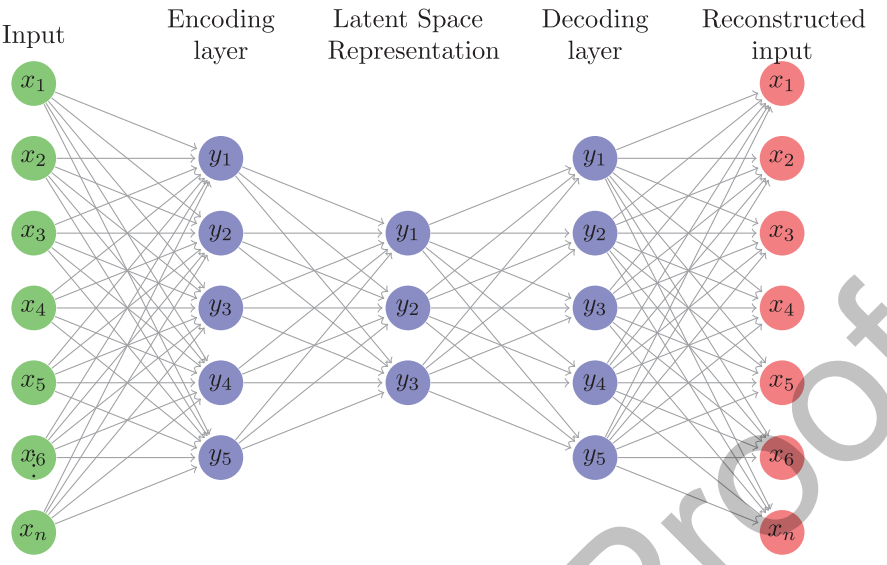


Fig. 1. Simple Autoencoder architecture - The input is compressed and then reconstructed.

3.3. *Performance metrics of the classification model*

The four different types of metrics have been used in order to evaluate our detection model and the metrics are given as follow:

- Precision
- Recall
- F1 measure
- Accuracy

It is not difficult to obtain a high accuracy rate by selecting the size of the sampling data carefully out of the dataset during the training of the classification algorithm. Since the input instances which are used in the simulations are extremely imbalanced, conventional accuracy based model classification performance evaluation is not adequate to find out an optimal hypothesis. In order to achieve this problem, four different metrics, the average precision, average recall [25], F_1, the overall prediction accuracy are used to evaluate the model accuracy which are common evaluation metrics in machine learning [26].

Precision is described as the ratio of retrieved relevant instances. Precision calculation is shown in Equation 3.

$$Precision = \frac{Correct}{Correct + False} \tag{3}$$

Recall is described as the ratio of relevant instances that is retrieved. Recall calculation is defined in Equation 4.

$$Recall = \frac{Correct}{Correct + Missed} \tag{4}$$

The proposed model is evaluated with the recall and precision for each class from prediction performance then we calculate their mean. The mean precision and recall calculations are shown in Equation 5-6.

$$Precision_{avg} = \frac{1}{n_{classes}} \sum_{i=0}^{n_{classes}-1} Prec_i \tag{5}$$

$$Recall_{avg} = \frac{1}{n_{classes}} \sum_{i=0}^{n_{classes}-1} Recall_i \tag{6}$$

F_1-measure is defined as the harmonic mean of the precision and the recall values. The evaluation model uses multi class modified version of F_1 that is shown in Eq.

$$F_1 = 2 \times \frac{Prec_{avg} \times Recall_{avg}}{Prec_{avg} + Recall_{avg}} \tag{7}$$

4. The proposed approach

In this section, we explain the details of our autoencoder based malicious traffic detection method. Section 4.1 provides the basic idea for our autoencoder model. The autoencoder based deep neural

network model implementation of the proposed method is introduced in Section 4.2.

4.1. Basic idea

The main motivation of this research is the idea that an autoencoder model behind a deep neural network model could build more accurate classifier model that is comparable to conventional neural network model to detect malicious computer network flow. Our main tasks are (i) to create CSV file from PCAP format, (ii) to use autoencoder methods to build data representation model and (iii) finally to create a malicious network flow detection model using a deep neural network. The basic idea in our autoencoder model is to learn an approximation to the identity function, $h_{W,b}(\mathbf{x}) \approx \mathbf{x}$. For our dataset, the inputs $\mathbf{x}$ are the network flow values created from PCAP file with 28 columns so input layer has 28 units, and there are 19, 9, and 19 units in hidden layers as shown in Fig. 2. Note that we also have $\mathbf{y} \in \mathbb{R}^{28}$. Since there are only 19, 9, 19 hidden units, so the network is forced to learn a *compressed* representation of the network flow.

4.2. The classification model

The proposed model consists of a combination of 2 different models in the form of an autoencoder and a deep artificial neural network. We designed our Autoencoder model as a total of 5 layers. The unit numbers in these layers are 28, 19, 9, 19 and 28, respectively. We used sigmoid as an activation function in each unit. Our classification model, which we designed based on deep artificial neural network, consists of 6 layers in total. The unit numbers in these layers are 28, 500, 800, 1000, 800, and 500, respectively. The model constructed for training purposes is depicted in Fig. 2.

Algorithm 1 shows the pseudo code of the autoencoder model building process. Algorithm 2 shows the overall process of the neural network model building with encoded input dataset that contains network flows. Algorithm 2 uses the autoencoder model build that is created in Algorithm 1. Algorithm 3 classifies the unknown labeled input instance $\mathbf{x}$ using *norm_model*, *auto_enc* and *ddos_model*.

Algorithm 1 Autoencoder based data transformation

input : Network flow based training data $\mathcal{X} \in \mathbb{R}^{m \times n}$
output: Autoencoder model *auto_enc*,
normalization model *norm_model*

```
/* Create normalization model with
   X                                   */
```
$norm\_model = (\mathcal{X})$
```
/* Normalize input
   data set                            */
```
$\mathcal{X} = norm\_model(\mathcal{X})$
```
/* create train and
   test dataset                        */
```
$\mathcal{X}_{train}, \mathcal{X}_{test} = train\_test\_split(\mathcal{X}, 0.2)$
```
   /* Create autoencoder model         */
```
$auto\_enc = train\_autoencoder(\mathcal{X}_{train})$
```
   /* Convert input dataset X
   using the model                     */
```
$\mathcal{X}_{encoded} = auto\_enc(\mathcal{X})$

Algorithm 2 Deep neural network based DDoS detection model building

input : Encoded input dataset $\mathcal{X}_{encoded}$,
label vector $\mathbf{y}$
output: neural network model *ddos_model*

```
/* create train and test dataset
   */
```
$\mathcal{X}_{end\_train}, \mathcal{X}_{end\_test}, \mathbf{y}_{train}, \mathbf{y}_{test} =$
$train\_test\_split(\mathcal{X}_encoded, \mathbf{y}, 0.33)$
```
   /* Create neural network model
   */
```
$ddos\_model = nn\_fit(\mathcal{X}_{enc\_train}, \mathbf{y}_{train})$

Algorithm 3 Autoencoder based deep neural network classification process

input : Instance vector with unknown label $\mathbf{x}$,
Autoencoder model *auto_enc*,
normalization model *norm_model*,
neural network model *ddos_model*
output: Input instance $\mathbf{x}$'s label y

```
/* Normalize input instance            */
```
$\mathbf{x} = norm\_model(\mathbf{x})$
```
/* Encode normalized
   input instance                      */
```
$\mathbf{x}_{encoded} = auto\_enc(\mathbf{x})$
```
   /* Find label of
   encoded input instance              */
```
$y = ddos\_model(\mathbf{x}_{encoded})$

4.3. The training steps of the classification model

The CPU and GPU platforms have been used in order to train the classification model. There are no

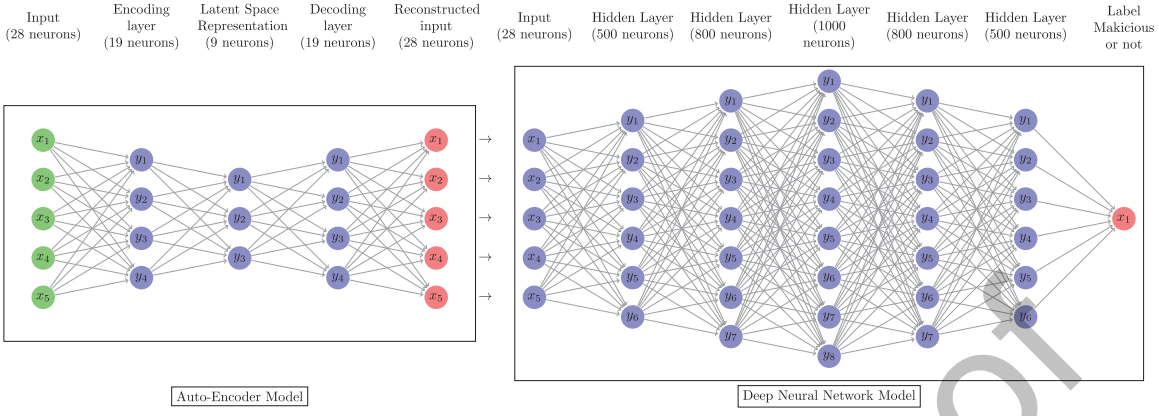


Fig. 2. The ddos detection model.

changes in the obtained result during the verification of the model. However, it is measured that the performance of the GPU is much better than the performance of the CPU when each iteration and the total time for training are considered. The proposed model has been tested with a CPU and a NVIDIA Quadro 1000M GPU.

5. Experiments

In this section, we explained our simulation environment and shared the obtained simulation results.

5.1. Experimental setup

In this section, we apply our detection model to an experimental data sets to verify its effectivity and efficiency. To demonstrate the effectiveness and performance of the proposed model, we apply it on a public classification data set.

Network traffic passing over the network components used today is quite high. As a result, the algorithms and parameters to be used must also be appropriate for this speed. In the deep learning and artificial neural networks, the most important parameter to be considered is the optimization method. Almost all of the classification and regression algorithms, such as artificial neural networks, linear regression, logistic regression, by default, use the stochastic gradient descent (SGD) optimization method. However, this optimization method requires the entire data set in each iteration to calculate the weights of the artificial neural network. Therefore, it is not suitable for training a massive data set such as network traffic. For this reason, we preferred mini-batch SGD optimization method. Batch-size is a hyperparameter that represents the number of samples used to update model parameters.

We developed our testing codes by using Python 3.5 with 64 bits. Keras, tensorflow and scikit-learn libraries of Python are used for machine learning and deep learning implementations in this work. We also used Windows 7 with 64 bits for our development environment.

5.2. Experimental results

A Receiver Operating Characteristic (ROC) curve is a way to compare diagnostic tests. It is defined as the ratio of the true positive rate against the false positive rate. ROC is commonly used in Signal Detection theory and also it is mainly used in machine learning in order to measure the performance. In our research work, we also used ROC to measure the performance of our autoencoder and deep neural network model.

Another important measurement tool is the precision-recall curve. This curve is usually used when the class distribution is imbalanced in the input data set. The Precision-recall curve shows trade-off values between precision and recall for different threshold values. In our research, we also used the Precision-recall curve to evaluate the performance of our autoencoder and deep neural network model.

We applied binary cross-entropy loss function to evaluate our model in each epoch. The binary cross-entropy loss function is used to measure the error at a softmax layer and is given by

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(y^{(i)}log(h_\theta(\mathbf{x}^{(i)}))+ \\ \text{ffl}(1-y^{(i)})log(1-h_\theta(\mathbf{x}^{(i)}))) \tag{8}$$

Our aim is to make our DDoS detection model's output distribution as close as possible to true label distribution. The detection model tries to maximize the probability of the input dataset is the best model.

Experimental results are given separately for autoencoder and deep neural network parts in the following sub-sections.

5.2.1. Autoencoder model results

We first explain the autoencoder performance results. The autoencoder is the first part of the proposed model. Hence, the performance result of the autoencoder model would affect the model that is the deep neural network model. We used autoencoder model in order to find a data model that represents normal and malicious network traffic. We set the epoch value to 500. The model after the 500 epoch converges to its steady-state position.

Fig. 3 shows the ROC, and the model loss values for the autoencoder model.This result indicates that our DDoS detection model can achieve quite high true positive rate and low false positive rate.

5.2.2. Deep neural network results

We expressed the performance results of our deep neural network model in this section. The evaluation results are depicted in Fig. 4. We set the epoch value to 300. Also, after the 300 there is not any meaningful changes.

Table 3
The evaluation results of the model with different activation functions

Activation	Accuracy	Precision	Recall	F_1
hard sigmoid	0,9471	0,8618	0,6913	0,7671
relu	0,9744	0,8924	0,9053	0,8985
sigmoid	0,9487	0,8802	0,6879	0,7722
softplus	0,9378	0,7236	0,9568	0,8113
tanh	0,9614	0,8079	0,9130	0,8569

Table 4
The evaluation results of the model

Platform	Training Time (sec)
Quadro 1000M	3516,21
Intel i7-600	10601,42

The overall of model evaluation results of the model is given in Table 3. We obtained the best F_1 results with *relu* activation function.

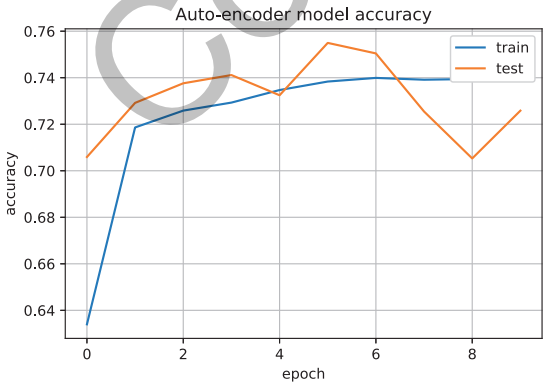
The overall training time for the proposed model is given in Table 4.

5.2.3. KDDCUP'99 results

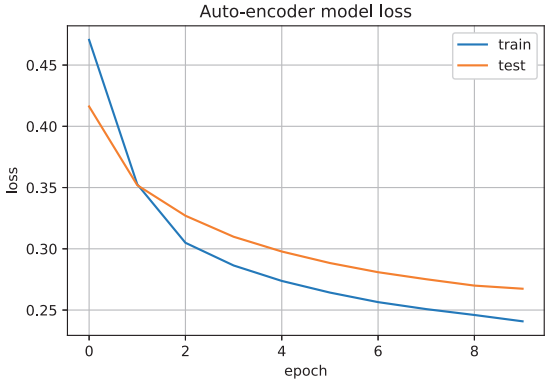
In this section, we have applied the DDoS detection system to the dataset KDDCUP'99, which is an old IDS dataset. We have not made any changes to the model created with the artificial neural network that we have proposed. The types of attacks in this data set are as follows: neptune, smurf, and teardrop.

A *teardrop* attack is a DDoS attack that involves sending fragmented network packets to a victim computer.

A *Neptune* attack generates a SYN Flood attack against a network host by sending session establishment packets using a forged source address.

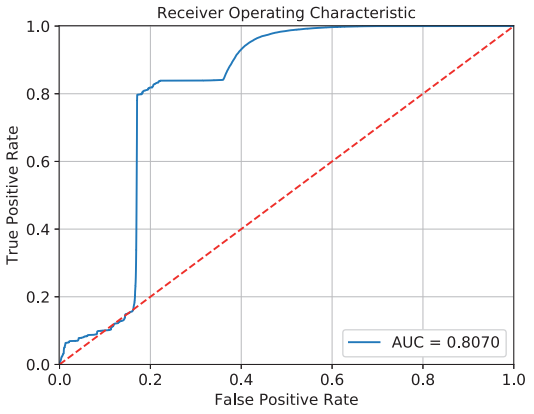


(a) Model accuracy results

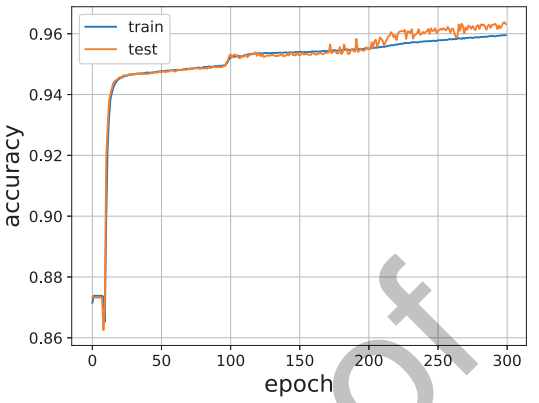


(b) Model loss results

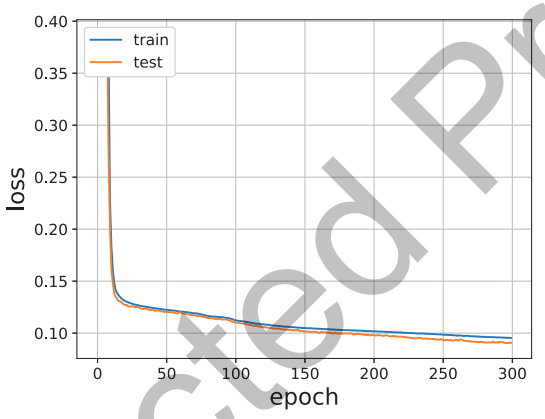
Fig. 3. Autoencoder model evaluation.



(a) ROC



(b) Model accuracy results



(c) Model loss results

Fig. 4. Deep neural network model evaluation.

Table 5
The evaluation results of the model with different activation functions for KDDCUP'99

Attack	Activation	Acc.	Prec.	Recall	F_1
Neptune	sigmoid	0.5269	0.5269	1.0	0.6902
	softplus	0.9996	0.9999	0.9993	0.9996
	softsign	0.9998	0.9999	0.9998	0.9998
	relu	0.9997	0.9997	0.9997	0.9997
	tanh	0.9998	0.9999	0.9997	0.9998
	hard sigm.	0.5269	0.5269	1.0	0.6902
Smurf	sigmoid	0.7450	0.7450	1.0	0.8538
	softplus	0.9998	0.9998	0.9999	0.9999
	softsign	0.9997	0.9998	0.9998	0.9998
	relu	0.9997	0.9997	0.9998	0.9998
	tanh	0.9998	0.9998	0.9998	0.9998
	hard sigmoid	0.74422	0.7442	1.0	0.8533
Teardrop	sigmoid	0.9911	0.0	0.0	0.0
	softplus	0.9979	0.9784	0.7816	0.8690
	softsign	0.9987	0.9807	0.8793	0.9272
	relu	0.9987	0.9745	0.8793	0.9244
	tanh	0.9987	0.9807	0.8793	0.9272
	hard sigmoid	0.9911	0.0	0.0	0.0

A *Smurf* attack is a DDoS amplification vector that boosts its damage potential by exploiting broadcast networks with large numbers of Internet Control Message Protocol (ICMP) packets.

Table 5 shows the detection performance results for KDDCUP'99 dataset. As can be seen when the table is examined, the performance results of the deep learning-based classifier model are also quite high for KDDCUP'99 dataset.

6. Conclusion and future work

In this work, we have proposed a new model in order to detect malicious activities on a network by using machine learning and deep learning technologies. In our work, we have experienced that the training of the proposed model required less time of period when it is run on a GPU. Furthermore, we

have concluded that the proposed approach can be applied to datasets that are used in several cybersecurity areas. The proposed approach ignores the misdirecting fields such as source IP whereas it considers the properties of arriving packets such as size, arriving time, size of a payload, etc. in order to detect the malicious network activities. So, the packets are not affected by the various attacks such as fraud IP or fraud MAC addresses.

Autoencoder algorithm is mainly used in deep learning methods. The Autoencoder model is used for cleaning the outlier on the data. Autoencoder is actually a generative model (ie the same as the input and output vector size). As a result, this model does not change the size of the data, but changes the structure of the data. In this work, we constructed a data representation from the data set that is built from the network traffic through the autoencoder algorithm. We aim to define each network behavior having different classification tag in the data set. Data passed through the autoencoder model is forwarded to deep neural network model in order to identify the correct classification tag. When we review the ROC curve, it is close to 0 around the top left corner of the X and Y. Hence, we can obtain that false-positive rate is close to 0 and the true-positive rate is close to 1 in our proposed model. As a result, our proposed approach reveals a robust model. Furthermore, when we look at the precision-recall graph, the area under the graph is big then it means that the recall and the precision value are also high.

In future work, we plan to use long-short term memory (LSTM) algorithm that is time varied method to use for training purposes. Malicious activities in network traffic are changes based on time, so we expect to obtain considerably correct results when we use the LSTM approach. Moreover, data sets for malicious network traffic and malicious software analysis are rapid changes based on time as well. Hence, we consider that the usage of LSTM approach will also be a good solution for other Cyber Security data sets.

References

[1] N. Ben-Asher and C. Gonzalez, Effects of cyber security knowledge on attack detection, *Computers in Human Behavior* **48** (2015), 51–61.

[2] B. Jasiul, M. Szpyrka and J. Sliwa, Detection and modeling of cyber attacks with petri nets, *Entropy* **16** (2014), 6602–6623.

[3] D. Jiang, Z. Xu, P. Zhang and T. Zhu, A transform domain-based anomaly detection approach to network-wide traffic, *Journal of Network and Computer Applications* **40** (2014), 292–306.

[4] B.C.M. Cappers and J.J. van Wijk, Snaps: Semantic network traffic analysis through projection and selection, *IEEE Symposium on Visualization for Cyber Security (VizSec)* (2015), 1–8.

[5] F. Han, L. Xu, X. Yu, Z. Tari, Y. Feng and J. Hu, Sliding-mode observers for real-time ddos detection, *IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, 2016, pp. 825–830.

[6] A. Abeshu and N. Chilamkurti, Deep learning: The frontier for distributed attack detection in fog-to-things computing, *IEEE Communications Magazine* **56** (2018), 169–175.

[7] A.A. Diro and N. Chilamkurti, Distributed attack detection scheme using deep learning approach for Internet of Things, *Future Generation Computer Systems* **82** (2018), 761–768.

[8] L. Feinstein, D. Schnackenberg, R. Balupari and D. Kindred, Statistical approaches to ddos attack detection and response, *Proceedings DARPA Information Survivability Conference and Exposition*, 2003, pp. 303–314.

[9] M.H. Bhuyan, D. Bhattacharyya and J. Kalita, An empirical evaluation of information metrics for low-rate and high-rate ddos attack detection, *Pattern Recognition Letters* **51** (2015).

[10] V. Srihari and R. Anitha, Ddos detection system using wavelet features and semi-supervised learning, *Security in Computing and Communications* (2014), 291–303.

[11] P.D. Bojović, I. Bašičević, S. Ocovaj and M. Popović, A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method, *Computers & Electrical Engineering* **73** (2019), 84–96.

[12] F.O. Catak and A.F. Mustacoglu, Derin Ögrenme teknolojileri kullanarak dagitik hizmet disi birakma saldirilarinin tespit edilmesi, *The 5th High Performance Computing Conference*, 2017, pp. 1–8.

[13] S. Bhatia, Ensemble-based model for ddos attack detection and flash event separation, *Future Technologies Conference (FTC)*, 2016, pp. 958–967.

[14] D. Kshirsagar, S. Sawant, A. Rathod and S. Wathore, Cpu load analysis & minimization for tcp syn flood detection, *Procedia Computer Science, International Conference on Computational Modelling and Security*, 2016, pp. 626–633.

[15] V.D. Katkar and S.V. Kulkarni, Experiments on detection of denial of service attacks using ensemble of classifiers, *International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, 2013, pp. 837–842.

[16] O. Osanaiye, H. Cai, K.-K.R. Choo, A. Dehghantanha, Z. Xu and M. Dlodlo, Ensemble-based multi-filter feature selection method for ddos detection in cloud computing, *EURASIP Journal on Wireless Communications and Networking* **2016** (2016), 130.

[17] C. Livadas, R. Walsh, D. Lapsley and W.T. Strayer, Usilng machine learning technliques to identify botnet traffic, *Proceedings 31st IEEE Conference on Local Computer Networks*, 2006, pp. 967–974.

[18] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani and D. Garant, Botnet detection based on traffic behavior analysis and flow intervals, *Computers & Security* **39** (2013), 2–16.

[19] V. Srihari and R. Anitha, Ddos detection system using wavelet features and semi-supervised learning, *Security in Computing and Communications* (2014), 291–303.

[20] A. Gaurav and A.K. Singh, Entropy-score: A method to detect ddos attack and flash crowd, *2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 2017, pp. 1427–1431.

[21] P. Deshpande, S.C. Sharma and P.S. Kumar, Security threats in cloud computing, *International Conference on Computing, Communication Automation*, 2015, pp. 632–636.

[22] F. Hategekimana, P. Nardin and C. Bobda, Hardware/software isolation and protection architecture for transparent security enforcement in networked devices, *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (2016), 140–145.

[23] L.F. Maimo, A.L.P. Gomez, F.J.G. Clemente, M.G. Perez and G.M. Perez, A self-adaptive deep learning-based system for anomaly detection in 5g networks, *IEEE Access* **6** (2018), 7700–7712.

[24] N. Moustafa and J. Slay, The evaluation of network anomaly detection systems: Statistical analysis of the unswnb15 data set and the comparison with the kdd99 data set, *Information Security Journal: A Global Perspective* **25** (2016), 18–31.

[25] B. Ma, T. Jiang, X. Zhou, F. Zhao and Y. Yang, A novel data integration framework based on unified concept model, *IEEE Access* **5** (2017), 5713–5722.

[26] Z. Jianqiang, G. Xiaolin and T. Feng, A new method of identifying influential users in the micro-blog networks, *IEEE Access* **5** (2017), 3008–3015.