ACTA INFOLOGICA 2022;6(2):199-207
dergipark.org.tr/acin

DOI: 10.26650/acin.1142806

RESEARCH ARTICLE

Detection of Attacks in Network Traffic with the Autoencoder-Based Unsupervised Learning Method

Otokodlayıcı Tabanlı Denetimsiz Öğrenme Yöntemi ile Ağ Trafiğindeki Saldırıların Algılanması

Yalçın Özkan[1]

[1] (Assist. Prof), Istinye University, Faculty of Economics, Administrative and Social Sciences, Istanbul, Turkiye

ORCID: Y.Ö. 0000-0002-3551-7021

Corresponding author:
Yalçın ÖZKAN
Istinye University, Faculty of Economics, Administrative and Social Sciences, Istanbul, Turkiye
E-mail address: yalcin.ozkan@istinye.edu.tr

Submitted: 09.07.2022
Revision Requested: 13.10.2022
Last Revision Received: 14.10.2022
Accepted: 14.10.2022
Published Online: 18.11.2022

Citation: Ozkan, Y. (2022). Detection of attacks in network traffic with the autoencoder-based unsupervised learning method. *Acta Infologica, 6*(2), 199-207. https://doi.org/10.26650/acin.1142806

ABSTRACT

The effects of attacks on network systems and the extent of damages caused by them tend to increase every day. Solutions based on machine learning algorithms have started to be developed in order to develop appropriate defense systems by detecting attacks in a timely and effective manner. This study focuses on detecting abnormal traffic on networks through deep learning algorithms, and a deep autoencoder model architecture that can be used to detect attacks is recommended. To this end, an autoencoder model is first obtained by training the normal dataset without class labels in an unsupervised manner with an autoencoder, and a threshold value is obtained by running this model with small size test data with normal attack observations. The threshold value is calculated as a value that will optimize the model performance. It is observed that supervised learning methods lead to difficulties and cost increases in the detection of cyber-attacks and the labeling process. The threshold value is calculated using only small test data without resorting to labeling in order to overcome these costs and save time, and the incoming up-to-date network traffic information is classified based on this threshold value.

Keywords: Deep learning, Autoencoders, Unsupervised learning

ÖZ

Ağ sistemlerine yapılan saldırıların etkisi ve oluşturduğu hasarların boyutu gün geçtikçe artış eğilimi göstermektedir. Saldırıları zamanında ve etkin biçimde tespit ederek uygun savunma sistemleri geliştirmek üzere makine öğrenmesi algoritmalarına dayalı çözümler geliştirilmeye başlanmıştır. Bu çalışma, ağlara yönelik anormal trafiğin derin öğrenme algoritmaları yardımıyla belirlenmesi üzerine odaklanmakta ve saldırıların tespit edilmesinde kullanılabilecek bir derin otokodlayıcı model mimarisi önerilmektedir. Bu amaçla önce otokodlayıcı ile sınıf etiketleri olmayan normal veri kümesi denetimsiz biçimde eğitilerek bir otokodlayıcı model elde edilmekte, bu model normal saldırı gözlemlerine sahip küçük boyutlu bir test verisiyle birlikte çalıştırılarak bir eşik değer elde edilmektedir. Eşik değer, model performansını optimum kılacak bir değer olarak hesaplanmaktadır. Denetimli öğrenme yöntemlerinin, siber saldırıların tespit edilmesinde, etiketleme işleminin zorluklara ve maliyet artışlarına neden olduğu gözlemlenmektedir. Bu maliyetleri aşmak ve zaman kazanmak için etiketlendirme işlemine başvurmadan sadece küçük bir test verisini kullanarak eşik değer hesaplanmakta ve yeni gelen bir güncel ağ trafik bilgisi bu eşik değere göre sınıflandırılmaktadır.

Anahtar Kelimeler: Derin öğrenme, Otokodlayıcılar, Denetimsiz öğrenme

1. INTRODUCTION

Nowadays, many types of attacks that threaten existing network systems are known, and appropriate defense methods are suggested. It is considered an effective way to investigate whether there is an anomaly in the traffic flow in order to reveal whether there is an attack against the network. Such anomaly detections are based on the belief that malicious behavior differs from typical user behavior. The behaviors of abnormal users that are different from standard behaviors are also considered an intrusion (Khraisat & Gondal, 2019).

With regard to attacks on networks, it is possible to talk about two concepts such as "intrusion detection systems" and "intrusion prevention systems." The intrusion detection system passively monitors attacks and performs warning services. On the other hand, intrusion prevention systems try to stop the threat encountered and its effects. Since network attacks evolve continuously and in a way to become more dangerous, it is necessary to develop new defense models in order to prevent them effectively.

Unknown types of attacks are also referred to as zero-day attacks. In particular, it is important to detect zero-day attacks in a timely and accurate manner. It is observed that deep learning technologies are used to detect such attacks effectively (Gao & Ma, 2020). Deep learning technology has a very high ability to learn complex systems, patterns, details, and behaviors. Deep learning technologies are appropriate solutions that can be used to distinguish between normal traffic and network attacks detected as abnormal behaviors (Minsky & Doitshman, 2018).

Traditional attack detection algorithms do not perform well in detecting zero-day attacks. It is recommended to use autoencoders aimed at setting threshold values in order to detect such zero-day attacks with high accuracy (Aygun & Yavuz, 2017; Roshan. & Zafar). Deep autoencoders, among the deep learning technologies, are considered "unsupervised deep learning" algorithms. Since these algorithms can detect zero-day attacks instantly (Dutta & Pawlicki, 2022) and largely reduce the labeling process, they are considered an important tool for detecting attacks on the network (Song & Hyun, 2021). It has been observed that deep learning autoencoders can perform successfully in imbalanced datasets where the number of normal samples is much higher than the abnormal ones (He & Wang, 2021).

In this study, a deep autoencoder model architecture is recommended to detect network attacks in imbalanced datasets. It is aimed to determine the threshold value by applying this model, which is trained with unlabeled data, to another dataset and to use this threshold value as a classification tool in other datasets. In the designed system, no expert is needed to label the network traffic or to occasionally update the model, and there is a self-learning unsupervised model.

2. MATERIALS AND METHODS

Autoencoding methods, a sub-application of deep neural networks, can be used to detect, analyze, and interpret attacks on networks and develop solutions for them.

2.1. Deep neural networks

A multi-layer neural network, in other words, a **deep neural network,** consists of interconnected neurons that form a neural structure. As seen in Figure 1, it consists of many hidden layers along with input and output layers. The hidden layer consists of neurons, and these neurons are connected with each other and with input and output layers. Deep learning networks are constructed from as many hidden layers and neurons as needed (Chollet, 2019). The input layer is the first layer of artificial neural networks. Data inputs are made through this layer. The relevant datasets are read to the network through this layer in the form of rows. The attributes of the dataset are arranged in order as $xx_1, xx_2, \dots, xxn$. Each of the input elements at the first stage is connected with all the neurons in the first row of the hidden layers. Hidden layers process the data from the input layer and transfer it to the next hidden layer elements.

In artificial neural networks, each connection has a weight. These weights form the basis of artificial neural networks. These weights determine the relationships between the inputs and outputs of the network. The inputs and outputs of a neuron are

calculated consecutively. This calculation process is the "forward propagation" stage. The data to be transferred from the input layer to the output layer are calculated with the help of the "aggregation function." At the forward propagation stage, the net input value is calculated by adding the input value for each neuron after multiplying the network connection weights. As seen in Figure 2, a neuron consists of input and output connections. A bias input can also be added to the net input function.

The net input value is calculated with an activation function, and the output of the relevant neuron is obtained (Öztemel, 2020).

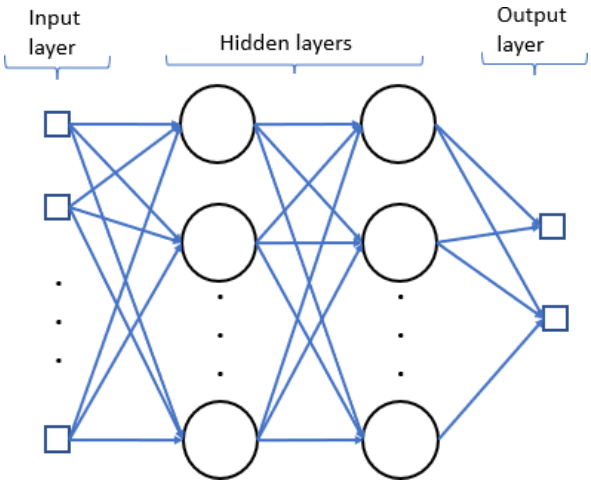

Figure 1. Deep learning model

An activation function in the neural network defines how the weighted sum of the input is converted into an output. The selection of the activation function significantly affects the capacity and performance of the neural network, and different activation functions can be used in different parts of the model. Activation functions should also typically be able to calculate the first-order derivative for a given input value. This property is required for the backpropagation stage in order to update the model's weights. The sigmoid function, one of the activation functions, takes any real value as input and converts it to values in the (0,1) range. If it is desired to convert the net input value to the (-1,1) range, the tanh activation function is preferred.

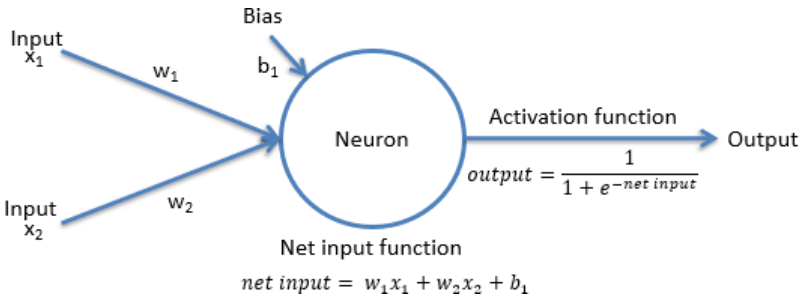

Figure 2. Behavior of a neuron in forward propagation

As seen in Figure 2, after all the output values are found, the forward propagation process is completed, and the backpropagation starts. The purpose of this process is to optimize the associated weights. The difference between the output obtained in the output layer and the actual outputs reveals the error obtained as a result of forward propagation, or in other words, the cost. After calculating the amount of error, the total error is distributed to all weights in the network. This process is called the "backpropagation process." In simple words, it tries to make the predicted value close to the actual value and reduce the error. To this end, to determine the effect of change in each w_k weight on the total error, its derivatives are calculated according to the relevant weight and multiplied by η, which is the learning rate, and thus the amount of change is calculated.

$$w_{\mathrm{k}}^{new} \leftarrow w_k - \eta \frac{\partial E_{total}}{\partial w_{\mathrm{k}}}$$

2.2. Autoencoders

There is a model of deep neural networks called an autoencoder, which is presented in figure 3. The most obvious feature of this model is that the outputs and inputs are the same. Another visible feature is that there is a narrowing in the middle of the hidden layers compared to the others. In the autoencoder deep neural network model, this hidden layer consisting of narrowed neurons is called the "bottleneck."

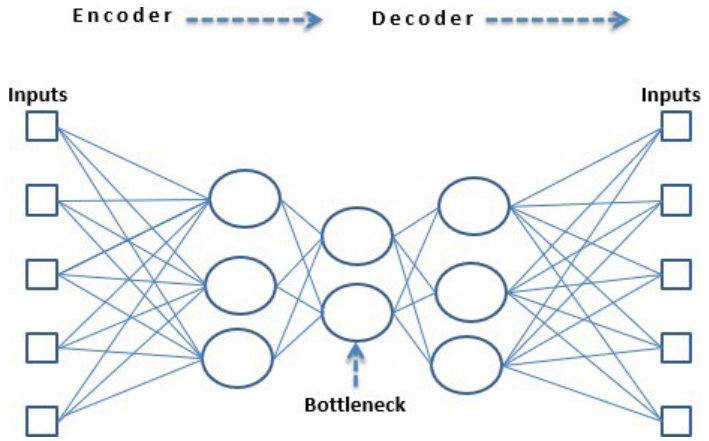

Figure 3. Autoencoder model architecture

An autoencoder model has two main parts. In the first part, there are components called encoders. The encoder process ensures that the data from the inputs are encoded up to the layer called the bottleneck. The bottleneck layer has fewer neurons compared to other hidden layers. In this way, it can also be interpreted as a *compression area* for the data coming to the bottleneck. The second part of the autoencoder model starts from the bottleneck layer and continues until it obtains outputs and is called the encoder. This part enables the decoding of the encoded hidden layer data. The forward propagation and backpropagation processes that are valid in deep learning models are also valid for autoencoders.

2.3. Anomaly detection with autoencoders

Autoencoders can be used to identify abnormal data, or in other words, outliers in data sets. The decoder part of autoencoders plays an important role in such analyses. As shown in Figure 4, the data in the bottleneck are resolved in other layers and reconstructed to match the output values. It is ensured that abnormal data appear by measuring the amount of reconstruction error obtained during the resolution process.

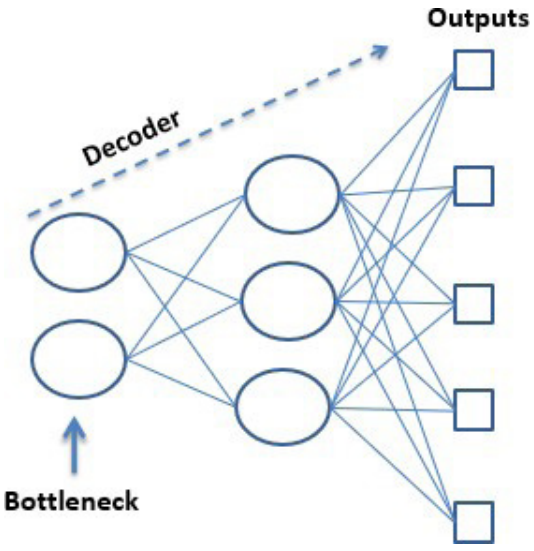

Figure 4. Decoder part of autoencoders

The way autoencoders work provides an effective solution that can be evaluated within the framework of unsupervised learning in the detection of anomalies. To this end, it is aimed to obtain a model without the target attribute. As indicated in Figure 3, a model is obtained by training the unlabeled data in an unsupervised manner with an autoencoder. When this model obtained from the data, which is considered normal, is run with different data, it detects different data as abnormal data (Özkan, 2021).

3. APPLICATION

In this study, the process indicated in figure 4 was applied. At the first stage, the dataset was divided into two parts to be used in the training and testing stages. Only normal traffic data were obtained by filtering the training dataset, and the test data were again divided into two parts. Thus, ultimately, a dataset containing normal traffic data, a validation dataset, and a test dataset were created.

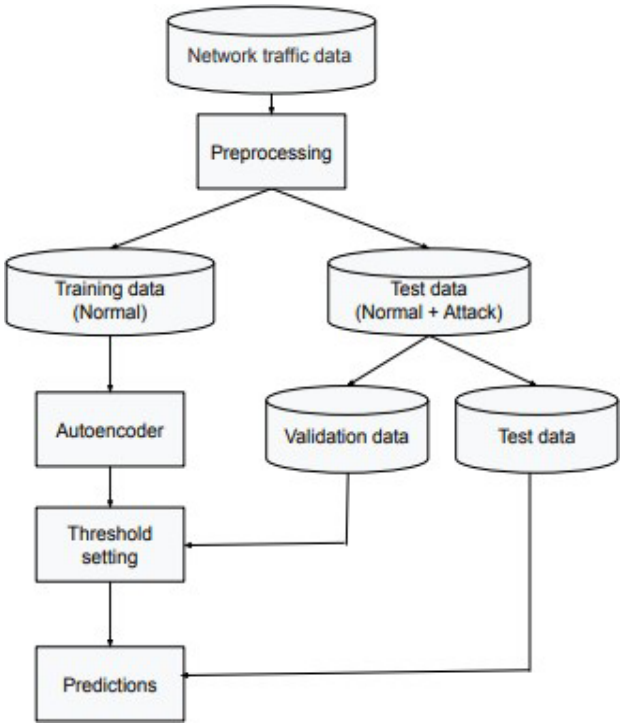

Figure 5. Process of detecting attacks on the network

For the raw dataset in the process in Figure 5, the Python programming language (Rossum & Drake, 1995) was used at the modeling and prediction stages, in addition to the stages of preprocessing and dividing data into three separate datasets. The software we developed included the Tensorflow (Abadi & Agarval, 2015), Keras (Chollet et al., 2015) and Matplotlib (Hunter, 2007) packages.

3.1. Dataset

To develop an autoencoder-based unsupervised learning model for cyber attacks, perform the classification process with this model, and obtain a threshold value for classification, the open-source dataset collection called CIC-IDS-2017 organized by the Canadian Cyber Security Center was downloaded (CICIDS2017, 2017), and appropriate processes were performed on it. This dataset includes normal traffic data and some common types of attacks. The dataset within the community includes timestamp, source and destination IPs, source and destination ports, protocols, and labeled class variables (Sharafaldin, 2018).

3.2. Data preprocessing

The dataset “Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv” included in the data collection called CIC-IDS-2017 is related to web attacks and was selected for our analysis. The preprocessing step was applied to this raw dataset, and some attributes that were found to be unnecessary were first removed from the raw dataset. The target attribute of this

dataset includes four class information. They include normal “Benign” labeled traffic records without network attack, ‘Brute Force’ labeled attack records and ‘XXL’ labeled attack records, and ‘Sql Injection’ labeled attack records. Normal traffic records were relabeled as “Normal,” and all others were relabeled as “Attack,” and the class attribute was converted to a two- class structure. After this process, the dataset was first divided into two parts as training and test datasets. Of the dataset for training, 60% and 40% of the dataset for testing were randomly selected. Only observations labeled “Normal” were selected by filtering the training data. Of the data reserved for the test process, 50% were randomly selected for validation, and 50% of the data were randomly selected for use in the test. Thus, three datasets were obtained to use them for different purposes. While the dataset reserved for training was created from 100821 observations with the class attribute, in other words, the class label only “normal,” the validation dataset was created from 34046 observations with the class attribute (normal + attack), and the test dataset was created from 34047 observations with the class attribute (normal + attack). The number of observations of all three data sets is presented in Table 1.

Since the autoencoder model does not have the class attribute, it is not actually a classification model. However, the model can be trained without the class attribute. Thus, if the dataset has a single class, then it can be trained using the data with a single class. The raw dataset used has four classes, as mentioned above. We aim to obtain a model using the dataset without test labels when it is given and perform classification on the test data in the form of normal+attack by employing this model. Thus, in order to pave the way for self-learning, the observations with the class attribute “Attack” were removed from the training dataset and turned into single-class data. Finally, in the data processing step, both normal and test data were normalized using the min-max algorithm.

Table 1
Datasets used in the study

Datasets	Number of observations
Training dataset	100821
Validation dataset	34046
Test dataset	34047

3.3. Autoencoder model

After data preprocessing was completed, the autoencoder model was defined. As seen in Figure 5, the model used has 77 input and output attributes and consists of 6 layers. As seen in Figure 6, the layer with 30 neurons in the model is the bottleneck layer of the autoencoder model.

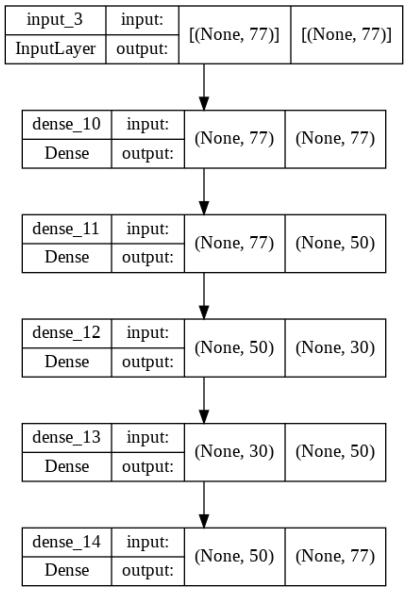

Figure 6. Autoencoder model architecture

3.4. Training of the model

The training inputs and outputs and the epoch parameters to be applied are defined before running the model. The epoch number of the training was applied as 20. The graph in figure 6 was created to monitor the model's performance during the training. During the model's training, losses are expected to decrease gradually and reach an acceptable level at each step. When the graphs in Figure 7 are examined, it is understood that there is a development in line with the purpose.

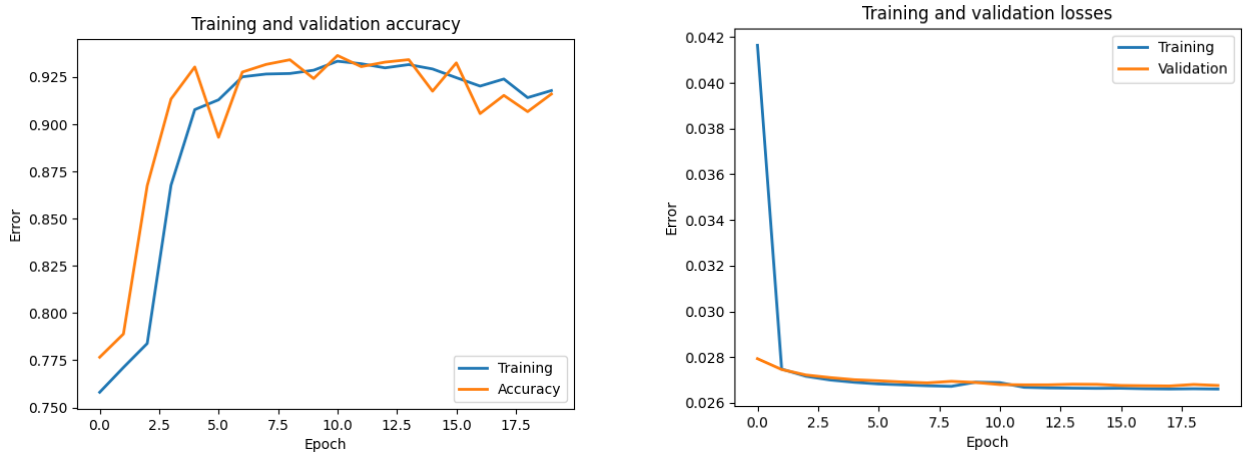

Figure 7. View of the model error

3.5. Prediction for the threshold value with validation data

A validation dataset was created to allow for threshold value setting in order to perform the classification process in the next steps. The trained autoencoder is run using the model validation dataset to obtain such a threshold value. The model learned only the observations with the class attribute label "normal." Since the validation dataset has two classes in the form of the class attribute (normal+attack), the model will accept the normal data in the validation data and evaluate the attack data as abnormal data. The MSE (Mean Squared Error) values that provide this distinction were obtained from the autoencoder. In this case, the sequence of MSE values was divided sequentially into two to select the most appropriate threshold value, each of the sections was considered a different class, and the predictions and AUC values were calculated. The split point with the largest AUC value among the results obtained was determined as the MSE=0.054347 threshold value.

3.6. Prediction with test data

The test dataset is classified according to the threshold value obtained using the validation data. The test data do not have to be labeled. If the model's performance is calculated at the end of the test process, labeled data should be used this time. While the observation values for MSE values greater than the threshold value in the test data were labeled as "attack," the smaller ones were labeled as "normal." Thus, separate predictions are obtained for each test observation.

Since the test dataset includes 30613 "normal" labeled data and 3434 "attack" labeled data, such a dataset is considered imbalanced. This is acceptable if the difference between the class distributions is small. However, in case of a great imbalance, the model accuracy becomes an unreliable measure of performance (Brownlee, 2021). The model accuracy was calculated as 90.67% in this state. ROC curves are used to calculate the actual performance of the model when there are imbalanced data. By drawing this curve, a value of 0.87 was obtained as the AUC value, as seen in figure 8, in order to reveal the model performance.

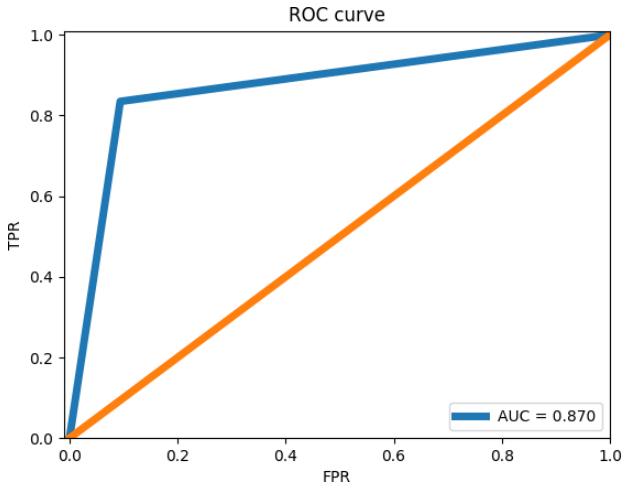

Figure 8. ROC curve

4. CONCLUSION

In this study, an autoencoder model was designed, and a threshold value calculation process was performed to achieve the purpose of classification. In this process, the raw network traffic data were divided into three parts to perform the training, validation, and testing phases. In the training phase, only the observations labeled as "normal" were addressed in accordance with unsupervised learning, and an autoencoder model was established. The reconstruction errors for each observation were compared by applying validation data consisting of normal + attack observations to this model, and consequently, an optimum threshold value capable of classifying between normal and attack observations was obtained. The model's performance was determined by applying the threshold value found on the test data. This study indicates that network traffic data can be completely classified without the need for labeled data. The presence of the class attribute, in other words, data labeling, increases the cost of modeling studies and slows down the development processes. It was revealed that in the real application of the proposed model, only validation data and some labeled data would be used at the stage of obtaining the threshold, and there was no need for labeled data during the training of the model and classification of new data.

Peer-review: Externally peer-reviewed.
Conflict of Interest: The author has no conflict of interest to declare.
Grant Support: The author declared that this study has received no financial support.

Hakem Değerlendirmesi: Dış bağımsız.
Çıkar Çatışması: Yazar çıkar çatışması beyan etmemiştir.
Finansal Destek: Yazar bu çalışma için finansal destek almadığını beyan etmiştir.

References/Kaynaklar

Abadi, M., Agarval, A., Barham, P., Brevdo., Chen, A., Citro, C. … Corrado, G.S. (2015), TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from tensorflow.org, DOI: 10.5281/zenodo.4724125

Aygun, R. C., & Yavuz, A. G. (2017, June). Network anomaly detection with stochastically improved autoencoder based models. In 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud) (pp. 193-198). IEEE.

Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Chollet, F., (2019). Python ile Derin Öğrenme [Deep Learning with Python]. (Aksoy, B.A. Trans.). İstanbul, Turkey: Buzdağı yayınevi.

CICIDS2017. (2017), *Intrusion Detection Systems Datasets*, Retrieved from https://www.unb.ca/cic/datasets/ids-2017.html

Dutta,V., Pawlicki,M., Kozik,R. & Choraś, M. (2022). Unsupervised network traffic anomaly detection with deep autoencoders, Logic Journal of the IGPL, jzac002.

Gao M, Ma L , Liu H, Zhang Z, Ning Z & Xu, J. (2020). Malicious Network Traffic Detection Based on Deep Neural
Networks and Association Analysis. Sensors.; 20(5):1452. https://doi.org/10.3390/s20051452

He, M., Wang, X., Zhou, J., Xi, Y., Jin, L., & Wang, X. (2021). Deep-Feature-Based Autoencoder Network for Few-Shot Malicious Traffic Detection. *Security and Communication Networks*, 2021. https://doi.org/10.1155/2021/6659022

Hunter, J. D. (2007), Matplotlib: A 2D graphics environment, Computing in Science \& Engineering, Volume 9, Number 3,
Pages 90-95.

Khraisat, A., Gondal, I., Vamplew, P. & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecur 2, 20 (2019). https://doi.org/10.1186/s42400-019-0038-7

Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A. & Kitsune. (2018). An Ensemble of Autoencoders for Online Network Intrusion Detection, Proceedings of the 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, CA, USA. 18–21 February 2018.

Özkan, Y., (2021). Uygulamalı Derin Öğrenme. Papatya Bilim Yayınevi.

Öztemel, E., (2020). Yapay Sinir Ağları. (4th ed.) [Neural networks], İstanbul, Turkey: Papatya Bilim yayınevi, ISBN: 978- 975-6797-39-6.

Roshan, K. & Zafar, A. (2021). An Optimized Auto-Encoder based Approach for Detecting Zero-Day Cyber-Attacks in Computer Network. 5th International Conference on Information Systems and Computer Networks (ISCON), 2021, pp. 1-6, doi: 10.1109/ISCON52037.2021.9702437.

Rossum, G., & Drake Jr, F. L. (1995). Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.

Sharafaldin,I., Habibi Lashkari, A.H., & Ghorbani, A.A., (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

Song, Y., Hyun, S., & Cheong, Y. G. (2021). Analysis of Autoencoders for Network Intrusion Detection. Sensors (Basel, Switzerland), 21(13), 4294, https://doi.org/10.3390/s21134294

Yang, L., Song, Y., Gao, S., Xiao, B., & Hu, A. (2020). Griffin: An Ensemble of AutoEncoders for Anomaly Traffic Detection in SDN, GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9322187.