GU J Sci, Part C,9(1): 071-083 (2021)

Gazi Üniversitesi

Fen Bilimleri Dergisi

PART C: TASARIM VE TEKNOLOJİ

http://dergipark.gov.tr/gujsc

Traffic Classification and Comparative Analysis with Machine Learning Algorithms in Software Defined Networks

Özgür TONKAL[1,*], Hüseyin POLAT[2]

[1]Samsun Üniversitesi, Bilgi İşlem Daire Başkanlığı, 55080, Canik, SAMSUN

[2]Gazi Üniversitesi, Teknoloji Fakültesi, Bilgisayar Mühendisliği Bölümü, 06560, Yenimahalle, ANKARA

Article Info

Research article
Received: 27/01/2021
Accepted: 05/02/2021

Makale Bilgisi

Araştırma makalesi
Başvuru: 27/01/2021
Kabul: 05/02/2021

Keywords

Software-Defined Network
Machine Learning
Traffic Classification

Anahtar Kelimeler

Yazılım Tanımlı Ağ
Makine Öğrenmesi
Trafik Sınıflandırma

Abstract

In computer networks, diverse applications generate network traffic with different characteristics. Network traffic classification is significant to manage networks better, improve service quality and ensure security. Software-Defined Networks (SDN) provides flexible and adaptable techniques for traffic classification with its programmable structure. SDN flows naturally exhibit particular characteristics of network applications and protocols. Therefore, it can be said that SDN can present significant opportunities in traffic classification using machine learning. This study proposes a traffic classification approach using machine learning models in SDN. In this study, DNS, Telnet, Ping and Voice traffic flows were created on the SDN using the Distributed Internet Traffic Generator (D-ITG) tool. Twelve-features representing these traffic flows (the number of packets transmitted, average transmission time, the number of instantly transmitted packets, etc.) were determined, and over the SDN controller in the physical network, a real-time dataset was created by collecting data depending on the features. Later, the performance of k Nearest Neighbor (k-NN), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), Decision Tree (DT) and Naive Bayes (NB) machine learning models were tested for traffic classification on this dataset. When the k-NN model was tested on this real-time dataset, its classification accuracy was obtained as the maximum with 99.4%. Therefore, this model has been determined as a machine learning giving the highest classification performance with the lowest cost flow features in traffic classification in SDN.

Yazılım Tanımlı Ağlarda Makine Öğrenme Algoritmaları ile Trafik Sınıflandırma ve Karşılaştırmalı Analiz

Öz

Bilgisayar ağlarında, farklı uygulamalar farklı özelliklere sahip ağ trafiği üretirler. Ağları daha iyi yönetmek, hizmet kalitesini artırmak ve güvenliği sağlamak için ağ trafiğinin sınıflandırılması önemlidir. Yazılım Tanımlı Ağlar (YTA) programlanabilir yapısı ile trafik sınıflandırması için esnek ve uyarlanabilir teknikler sağlar. YTA akışları doğal olarak ağ uygulamaları ve protokollerinin belirli özelliklerini sergiler. Dolaysıyla, YTA' nın makine öğrenmesi kullanarak trafik sınıflandırmada önemli fırsatlar sunduğu söylenebilir. Bu çalışmada, YTA' da makine öğrenme modellerini kullanarak bir trafik sınıflandırma yaklaşımı öneriyoruz. Dağıtık İnternet Trafik Oluşturucu (D-ITG) aracı kullanılarak YTA üzerinde DNS, Telnet, Ping ve Ses trafik akışları oluşturulmuştur. Bu trafik akışlarını temsil eden on iki öznitelik (iletilen paket sayısı, ortalalama iletim süresi, anlık iletilen paket sayısı vb.) belirlendi ve fiziksel ağdaki YTA kontrolcüsü üzerinden gerçek zamanlı olarak özniteliklere ait veriler toplanarak bir veri seti oluşturuldu. Daha sonra da bu veri seti üzerinde trafik sınıflandırması için k En Yakın Komşu (k-EYK), Destek Vektör Makinesi (DVM), Çok Katmanlı Algılayıcı (ÇKA), Karar Ağacı (KA) ve Naive Bayes (NB) makine öğrenme modellerinin başarımı test edildi. Gerçek zamanlı olarak oluşturulan bu veri seti üzerinde k En Yakın Komşu modeli kullanıldığında %99.4 doğruluk oranı ile en yüksek sınıflandırma doğruluğu elde edilmiştir. Dolayısıyla, YTA'da trafik sınıflandırmasında, en düşük maliyetli akış öznitelikleri ile en yüksek sınıflandırma performansı veren makine öğrenme modeli olduğu tespit edilmiştir.

*Corresponding author, e-mail: ozgur.tonkal@samsun.edu.tr

DOI: 10.29109/gujsc.869418

1. INTRODUCTION

The increases in internet data traffic and diversity have made network traffic classification a critical issue for computer science. Classification methods are utilized to increase network service quality, to use network resources efficiently, and to detect attacks and anomalies on the network using traffic analysis [1].

In traditional networks, various methods, such as rule-based, load-based, and correlation-based are used for network traffic classification. However, these methods also have different problems of their own [2].

Rule-based methods are widely used in network traffic classification. Predefined rules are used to classify packets in the network. Network packet header information and port information are the properties on which the classification is based. Although this method provides very high performance for the known applications, it has not been successful for applications using dynamic port numbers. Therefore, network operators wanted to use different classification solutions [3].

The classification performance decrease due to the dynamic ports' use in rule-based methods has led researchers to classify using the load-information carried by the packets flowing over the network. This method, also known as the deep packet analysis, removed connection number and IP address information dependency. Even though the protocol header information in use changes, the classification performance will be high because of the matching process of characteristic signatures or patterns in the packet loads. However, the high costs of hardware used for deep packet analysis and the problem of not detecting the encrypted packet contents, which have been increasing recently, have created the limitations of these methods [4].

To eliminate the limitations and difficulties encountered in rule-based and load-based classification methods, researchers have used correlation-based network classification methods. The statistical properties of the flows making up the network traffic are used, such as packet size, arrival rates and flow time. Classification is performed by including different machine learning techniques such as DT, SVM, k-NN into the classification process. Classification accuracy is relatively higher for packs containing encrypted traffic since the packet content is not handled specially. However, since the correlation analysis carried out in each flow requires additional calculations, it creates an extra consumption when creating classified dataset. Also, traditional networks consist of many routers and switches managed by many different protocols. Because of this distributed structure, the application of machine learning methods on traditional networks poses a significant challenge [5].

Unlike traditional network architecture, SDN separates the control plane from the data plane, allowing the network to be programmed directly over a central controller [6]. The control plane is transported to a high-performance server, and the management of the network is performed through central controller software. The data plane is left on routers or switches supported by OpenFlow protocol and is only responsible for the packs' transmission. Although many protocols can be used in SDN architecture, the OpenFlow protocol is the most successful [7]. SDN provides data flow through the OpenFlow protocol. The controller collects network flow information through the OpenFlow protocol and can direct it to switches by writing dedicated rules. SDN allows network administrators to analyze traffic in software without the need for any physical hardware.

Machine learning-based methods use statistical properties learned from data for traffic classification. Since package contents are not utilized for classification, they can be classified as encrypted data. Machine learning methods reach the necessary information for the traffic classification via OpenFlow protocol in SDN, and the classification can be performed with a low calculation cost.

For traffic classification using machine learning, various approaches have been carried out. Wang et al. [8] proposed a Quality of Service (QoS) conscious traffic classification system, using a semi-controlled learning algorithm and Deep Packet Inspection (DPI). Traffic flows of both known and unknown applications are divided into different QoS classes with the method proposed. D. Rossi and S. Valenti [9] focused on the classification of applications running over the User Datagram Protocol (UDP) in their work. The UDP network traffic obtained with the Netflow network monitoring tool is classified with the SVM algorithm. He et al. [10] proposed a software-defined virtual traffic classification model called vTC, which dynamically determines the most suitable flow properties and the most effective machine learning classifier. They identified basic and advanced flow properties and used six different classification algorithms, k-NN, SVM, MLP, DT, NB and AdaBoost. Amaral et al. [11] proposed an SDN-based traffic classification architecture in their study. In their study, 500 samples from each of Youtube, Vimeo, Facebook, Linkedin, Skype, BitTorrent, Web Browsing (HTTP) and Dropbox applications were collected, and then these traffic samples were classified with the specified machine learning algorithms (Random Forest (RF), Stochastic Gradient Boosting (SGB) and Extreme Gradient Boosting (EGB)). Wang et al. [12] have proposed the Software-Defined Network Home Gateway model (SDN-HGW) to manage better smart home networks. In the study, an encrypted data packet classifier was developed using three deep learning-based approaches: MLP, Stacked Autoencoder (SAE) and Convolutional Neural Network (CNN). For the developments of these classifiers called DataNets, a dataset including more than 20,000 encrypted packages from 15 applications was used. The results obtained have shown that the developed DataNets can be applied to the SDN-HGW model for the real-time processing in the smart home network with accurate packet classification and high computation efficiency. Lim et al. [13] proposed an SDN architecture that could classify traffic using deep learning methods. In their study, Deep learning models, including the Multi-Layer Long Short Term Memory (LSTM) model and the combination of CNN and single-layer LSTM models, were used to classify dataset consisting of Remote Desktop Connection (RDP), Skype, Secure Shell (SSH), BitTorrent and Hyper-Text Transfer Protocol (HTTP) traffics. Meenaxi M Raikar et al. [14] have stated in their study that traditional traffic classification approaches have limitations due to heavy data traffic. To overcome these limitations, they proposed a model in which SDN architecture and machine learning methods are used together. HTTP, E-mail and Video-Audio (streaming) traffic data are classified with SVM, NB and the Nearest Centroid Classifier (NCC).

When the studies are examined, it is seen that machine learning methods give outstanding results in traffic classification. The datasets and the traffic flow characteristics selected have been determinant in the accuracy rates obtained in traffic classification using machine learning. The approach proposed is aimed to contribute to the studies in this field.

This study proposes an architectural approach classifying the traffic flow in SDN using machine learning methods. The traffic flows of DNS, Telnet, Ping and Voice were created in the network environment created using SDN architecture. These traffic flows were collected in real-time and classified using machine learning methods. The number of features to use for machine learning methods was determined based on maintaining compatibility with the application (SDN controller), and the study refrained from excessive and complicated calculations. Although the network traffic class is predetermined, it can be adapted to suit the destination. The system presented offers network operators a solution to classify the network traffic to increase network efficiency and service quality.

2. MATERIAL AND METHOD

This study proposes an architectural approach classifying the traffic flow in SDN using machine learning methods. The proposed approach consists of two stages. In the first stage, real-time traffic flows of DNS, Telnet, Ping and Voice were formed in the SDN simulation environment, and these traffic data were cumulated with the Collector module on the controller. The features of the traffic flows to be used in classification were determined and the model was trained by using k-NN, SVM, MLP, DT and NB machine learning models. In the second stage, again using trained models via the Classifier module on the controller, a real-time traffic classification was carried out. Figure 1 shows the architectural structure of the proposed approach.

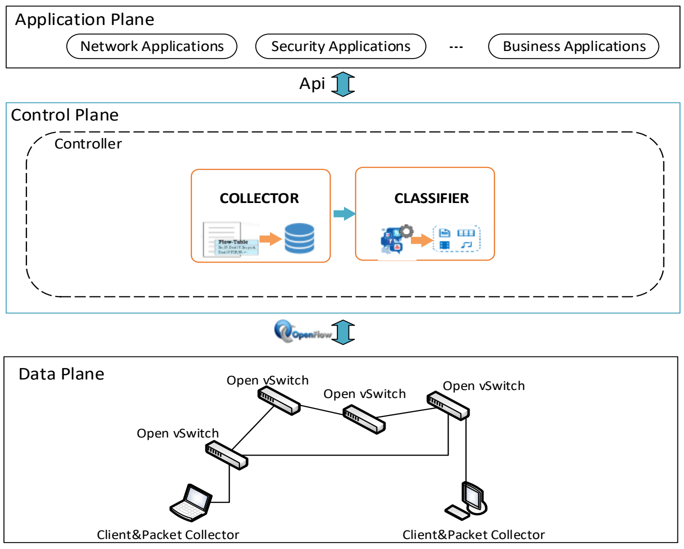


Figure 1. *The Proposed System Architecture*

2.1. Software-Defined Network Environment and Dataset Generation

As shown in Figure 2, a basic network topology has been created by using a virtualization environment (VirtualBox-VM). The topology has consisted of one controller, one Layer-2 switch, and three client computers (their system features are shown in Table 1) installed on five virtual machines. Nodes in the network were modeled as VM, and therefore some delay has been experienced in network traffic. Another option was to use the Mininet simulation environment commonly used in researches. However, a single VM simulation environment was not preferred to create a simulation close to the real environment.

Table 1. *System Features*

Operating System	*Ubuntu 14.04_64 bit Core i7 16 GB RAM*
SDN Controller	*RYU(4.30)_ 2 GB RAM*
Switch	*Open vSwitch (2.0.2) 2 GB RAM*
Hosts	*Ubuntu 14_04_64_bit 2 GB RAM*
Network Simulation Environment	*VirtualBox-VM*

An overlay network was created to allow the traffic, which had been generated by the client computers, to pass through the switch virtual machine to establish communication instead of using the internal switching mechanism of the virtual machines making up the SDN environment. Here, the study aims to create a network environment that is as realistic as possible and prevent delays. For client computers in the network, two interfaces were identified. The first interface was connected to the internal network, and the second interface to the switch (OVS) using the VXLAN tunnel. The switch was connected to both the client computers via the VXLAN interface and directly to the controller by the internal network. The RYU controller was placed in the Controller VM.

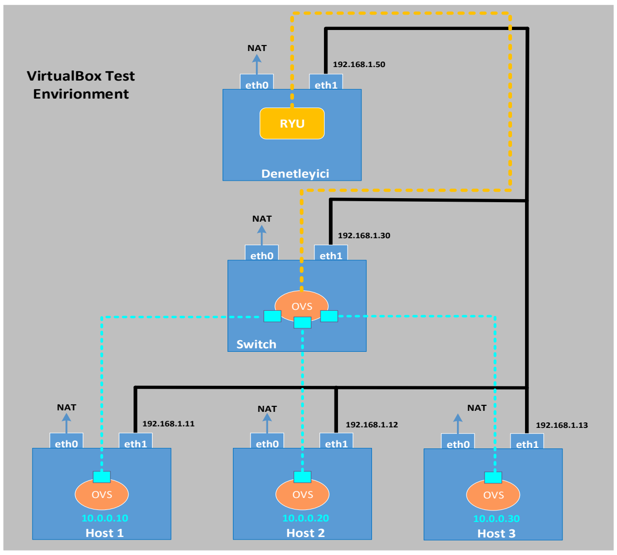


Figure 2. *Simulation Network Topology*

Once the SDN is configured, the controller needs to be aware of the packets flowing through the switch between client computers. For this, the controller can view the flow information shown in Table 2 at every second, using the Python script coded.

Table 2. *Flow Data Features-1*

Flow	***Description***
time	*UTC value at the time of flow information*
datapath	*Key ID in RYU*
in-port	*incoming traffic port*
eth_src	*source MAC address of the flow*
eth_dst	*destination MAC address of the flow*
out-port	*outbound traffic port*
total_packets	*total flow packets*
total_bytes	*the total size of flow packets (in Bytes)*

The flow data shown in Table 2 were used as input data to a script named "traffic_classifier.py" The script created a Flow object with the features shown in Table 3, using the flow data shown in Table 2. By using the features in the flow data, the number of distinguishing features to be used in traffic classification was increased.

Table 3. *Flow Data Features-2*

Feature	***Description***
Forward_delta_packets	*The number of packets seen since the last forward stream detection*
Forward_delta_bytes	*Bytes seen since the last forward stream detection*
Forward Instantaneous Packets per Second	*Instantaneous packets per second in the forward direction (src-> dst)*
Forward Average Packets per second	*The average number of packets per second in the forward direction (src-> dst)*
Forward Instantaneous Bytes per Second	*Instantaneous number of bytes per second in the forward direction (src-> dst))*
Forward Average Bytes per second	*Instantaneous number of bytes per second in the forward direction (src-> dst)*
Reverse _delta_packets	*The number of packets seen since the last reverse flow detection*
Reverse _delta_bytes	*Bytes seen since the last reverse flow detection*

DeltaReverse Instantaneous Packets per Second	*instantaneous packets per second in reverse direction (dst-> src)*
Reverse _avg_pps	*The average number of packets per second in the reverse direction (dst-> src)*
Reverse _avg_bps	*average bytes per second (dst-> src)*
Reverse Instantaneous Bytes per Second	*instantaneous bytes per second in the reverse direction (dst-> src)*
Traffic Type	*traffic type (ping-telnet-voice-DNS)*

The Traffic_classifier.py script can perform the following tasks:

Collection of the Training Data: For the traffics of DNS, Telnet, Ping, and Voice, training data are collected. The traffic must be flowing between the two client computers before the script file is operated.

Classification Using Machine Learning: Classifying the type of traffic flow between client computers using classification algorithms.

2.2. Dataset Generation

The D-ITG application was utilized to generate traffic flow data for training Machine Learning models. D-ITG is defined as a platform that can generate IPv4 and IPv6 traffic by accurately replicating the workload of existing Internet applications [15]. In this study, Ping, Telnet, DNS, Voice (G.711) traffics were created through the D-ITG application.

First, a specific traffic flow between a particular pair of client computers was simulated using D-ITG or other tools. Second, the traffic_classifier.py script file was started with the appropriate options to train the traffic type. The script initiates the RYU controller and the simple_monitor_AK.py (a modified version of simple_monitor_13.py). Data generated from the simple monitor script is collected and transformed to update Flow object features. The Flow object features are then periodically exported to the Comma Separated Variables file (CSV). Once CSV files are generated for each traffic type, they are combined into a complete Pandas Dataframe object used for model training and testing.

2.3. Data Preprocessing

The collected data were subjected to several preprocessing stages before being used for machine learning model training. First, the rows containing missing data were identified and removed. The mean values of relevant features were replaced in place of the removed ones to avoid possible information loss. Repeated or linearly increasing data may cause false predictions in the process of training machine learning models. Therefore, the features of "the number of transmitted packets", "the number of returned packets", "the amount of transmitted data" and "the amount of returned data" were removed from the dataset. Table 3 shows the features to use in the learning process. However, the high correlation level between the features determined during machine learning model training does not contribute to model training. The highly correlated features should be removed to increase the machine learning algorithms' efficiency. For this, the Principal Component Analysis (PCA) algorithm was used to determine the main components in the dataset and the non-parser features were removed from the dataset.

At the end of the data collection and data preprocessing, a dataset consisting of 10145 rows and 13 columns (12 features and 1 class feature) was obtained. This dataset consists of 2692 Ping, 2438 Telnet, 2554 Voice and 2461 DNS packets evenly.

3. TRAFFIC CLASSIFICATION WITH MACHINE LEARNING ALGORITHMS AND A COMPARATIVE ANALYSIS

Different classification algorithms are used to solve classification problems. Each classification algorithm has a unique mathematical model that it uses. Therefore, the results obtained show variations. It is only possible to determine which model is more successful by trying different classification models. This study tested the classification algorithms widely used today and compared their success rates.

A python-based scikit-learn library was used to test the performance of classification algorithms. This library supports many machine learning models [16]. This study focused on k-NN, SVM, MLP, DT and NB machine learning classification algorithms frequently used in classification problems. The features of these classification algorithms can be summarized as follows. k-NN is an algorithm used in both classification and regression problems. In this algorithm, the distance of the new data to join in the dataset is calculated according to the existing data, and its closest neighbors in the k number are checked. [17]. Euclidean, Manhattan, and Minkowski distance functions are used for distance calculations generally. The SVM is used to separate data belonging to two classes in the most appropriate way [18]. For this, decision boundaries, or in other words, the hyperplanes, are determined. The Linear Support Vector Machine (L-SVM) and the Radial Kernel Support Vector Machine (R-SVM) can be selected as the classifiers. DT is a decision support classifier that enables predictions using tree-like structures. Each node represents a stream tag, and root-to-leaf paths represent classification rules. Entropy is used for categorical variables, and the Least Squares method is used for Gini continuous variables. MLP is a type of feed-forward neural network working according to the supervised learning method. By giving the inputs and expected outputs to the network, the nodes' weights optimization is aimed. While feed-forward calculates the network output, back-propagation ensures the errors detected to be reduced by updating the weights in the net. This process repeats until faults in the network are minimized, or training is terminated. NB is a classifier based on Bayes decision theory using probability calculations. It aims to select the decision with the highest probability. It has different algorithms such as Gauss (G-NB), Multinomial (M-GB), and Bernoulli (BNB) distribution. Training and testing of machine learning models were carried out using the python_3 core structure in the Jupyter Notebook experiment environment with 16-GB RAM, 64-bit operating system, and 7-core 2.60-GHz processor.

Hyperparameter-adjustments were made on the models used to obtain optimum efficiency. Table 4 shows the algorithms and hyperparameter values used. During the models' training and testing processes, deviations (bias) and errors can occur due to the data distribution. The k-fold Cross-Validation Method (k=10) was used to prevent this situation. The k-Fold Cross Validation method divides the dataset into equal parts according to a specified number of k, allows each section to be used for both training and testing. Thus, possible problems of the model, such as overfitting and selection bias, are eliminated. The training and testing procedures were repeated ten times for each classification model, and so average values were calculated.

***Table 4**. Hyper Parameters Used in Algorithms*

Algorithms	*Hyperparameters*
k-NN	*metric='minkowski', n_neighbors=2, p=2, weights='uniform'*
SVM	*C=2.0, kernel='rbf'*
MLP	*activation='logistic', solver='lbfgs',alpha=0.1, hidden_layer_sizes= (100, 100), dropout value=0.1*
DT	*criterion='gini ', max_depth=10*
NB	*estimator=GaussianNB, iid='deprecated'*

3.1. Performance Metrics

This study used the confusion matrix to compare the performance of the classification models used. The confusion matrix is a table summarizing how successful the classification model has been (Table 5). It contains as many rows and columns as the number of classes the model has.

Table 5. *Binary Classification Confusion Matrix*

		Predicted Class	
		0	*1*
Actual Class	*0*	*True Negative (TN)*	*False Negative (FN)*
	1	*False Positive (FP)*	*True Positive (TP)*

While the samples of a particular class correctly defined by the classifier are placed in the True Positive (TP) indices, the samples of other accurately identified classes are placed in the True Negative (TN) indices. Similarly, specimens incorrectly predicted by the classifier are placed in the False Positive (FP) and False Negative (FN) indices in the confusion matrix. Accuracy, Precision, Sensitivity, F1 criterion (F-Score), Specificity, and ROC curves criteria are calculated using the values obtained in the complexity matrix (Equations 1-7). These criteria will be used to compare classification models.

- Accuracy: It is the criterion that gives the ratio of correctly classified samples to all samples (Equation 1).

$$\frac{TP+TN}{TP+FP+TN+FN} \tag{1}$$

- Precision: It is the ratio of correctly classified samples to total positive values (Equation 2).

$$\frac{TP}{TP+FP} \tag{2}$$

- Sensitivity: The ratio of correctly classified samples to true positive values (Equation 3).

$$\frac{TP}{TP+FN} \tag{3}$$

- Specificity: It is the ratio of the correct negative number to the sum of the correct negative and false positive numbers (Equation 4).

$$\frac{TN}{TN+FP} \tag{4}$$

- (F-Score): It is the harmonic mean of the Precision and Sensitivity values (Equation 5).

$$\frac{2*K*G}{K+G} \tag{5}$$

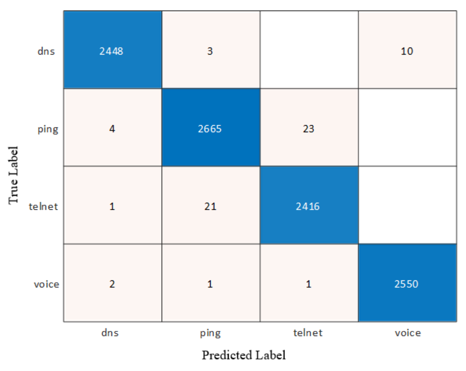
- ROC AUC: ROC (Receiver Operator Characteristic) is the probability curve used for different classes. It is often used to compare the ML algorithms' performances when unbalanced datasets are used. In the ROC curve, there are False Positive Ratio values (Equation 6) on the X-axis and True Positive Ratio values (Equation 7) on the Y-axis. AUC (Area Under the Curve) refers to the area under the ROC curve. The area under the ROC curve shows the performance of the classifier.

$$\frac{FP}{FP+TN} \tag{6}$$

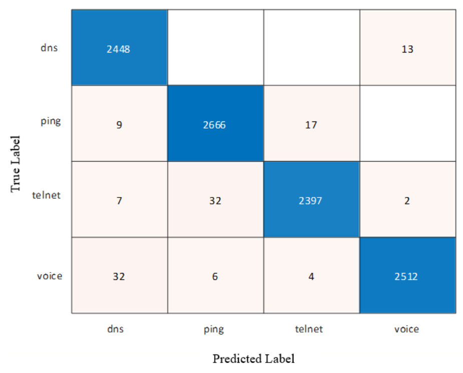
$$\frac{TP}{TP+FP} \tag{7}$$

3.1. Comparative Analysis

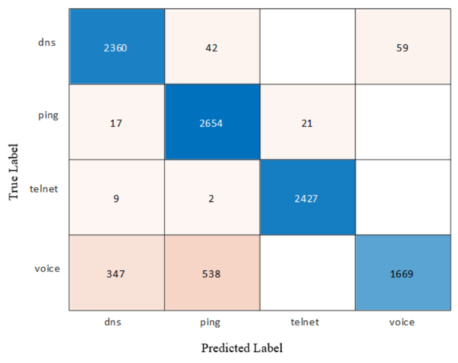
The complexity matrices of the different classification models used are shown in Figure 3.



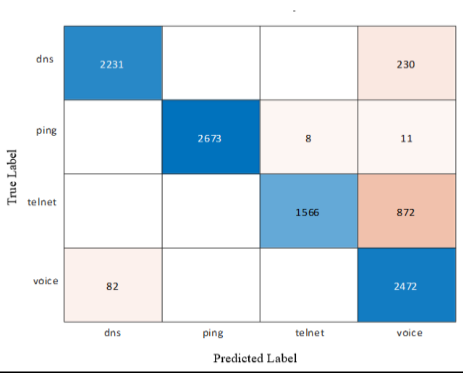
a)



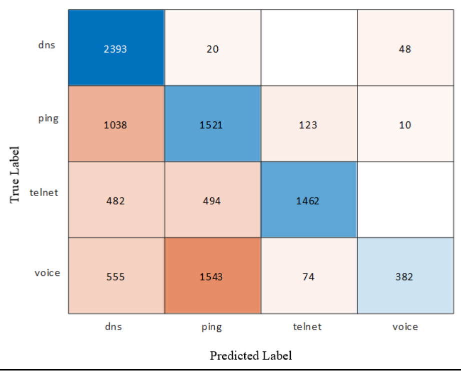
b)



c)



d)



e)

Figure 3*.Complexity Matrices of the models used; a) k-NN, b) MLP, c) DT, d) SVM,e) NB*

Comparison of classification models according to the criteria obtained by using confusion matrices is shown in Table 6.

Table 6. *Comparison of Models (average of 10 runs).*

Model	*Accuracy*	*Sensitivity*	*Specificity*	*Precision*	*F_1 Score*	*AUC*
k-NN	***0.9935***	***0.9935***	***0.9978***	***0.9935***	***0.9935***	***0.99895***
SVM	*0.8814*	*0.8774*	*0.9604*	*0.9122*	*0.8792*	*0.998625*
MLP	*0.9879*	*0.9879*	*0.9959*	*0.9880*	*0.9879*	*0.998875*
DT	*0.89798*	*0.89845*	*0.96572*	*0.91024*	*0.89429*	*0.967725*
NB	*0.5675*	*0.5716*	*0.8551*	*0.6775*	*0.5361*	*0.791825*

As shown in Table 6, although the MLP and the k-NN models gave outstanding results, the k-NN classification model gave better results than other models in all criteria. When the models' working times were compared, it was observed that while the k-NN was the fastest trained method, the decision tree was the fastest estimation method (Table 7). Although the MLPs' test performance was high, their training time was longer than other networks.

Table 7. *Comparison of Models Run Times (average of 10 runs).*

Model	***Training Time (sec)***	***Test Time (sec)***
k-NN	***0.2104***	*0.1783*
SVM	*21.812*	*0.8334*
MLP	*1401.3*	*0.1053*
DT	*0.73381*	***0.06723***
NB	*0.7492*	*0.11*

In Figure 4, a graphic was created using the AUC and F criteria of the models. According to this graph, the k-Nearest Neighbor model gave better results than other models. The ROC diagrams of the classification models used are given in Figure 5. ROC diagrams show the rate which the classifier estimates correctly. When this diagram is examined, the performance of the k-NN model is seen.

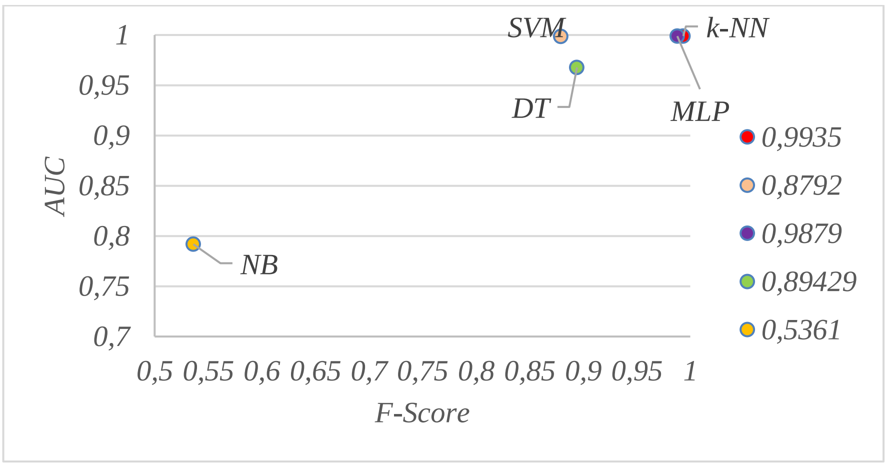
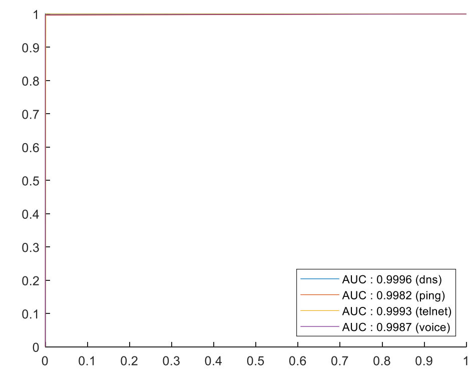
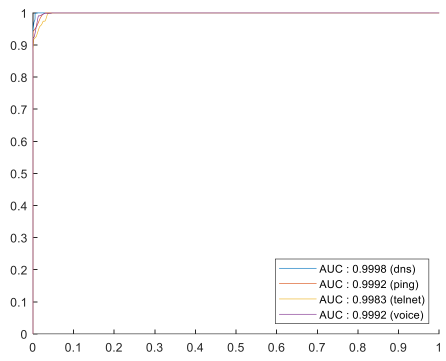


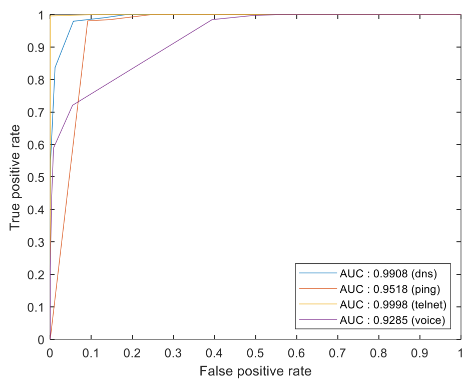
Figure 4. *Graph created with AUC and F1 values obtained as a result of running the models*



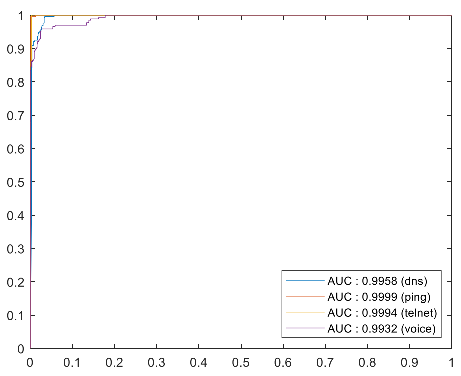
a)



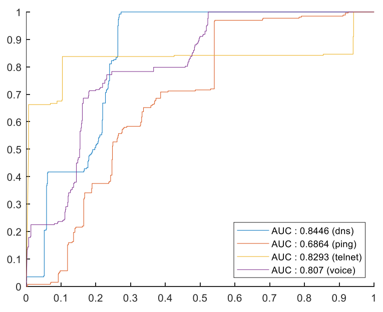
b)



c)



d)



e)

Figure 5.*ROC diagrams of the models used, a) k-NN, b) MLP, c) DT, d) SVM, e) NB*

The results of similar traffic classification studies using machine learning methods in SDN, whose reviews are in the introduction, are given in Table 8 comparatively. It is seen that the results obtained with the proposed traffic classification approach (k-NN and MLP) are more successful than the results obtained in other studies [8, 9, 11]. Comparison results are difficult to justify, as similar studies in the literature have been carried out using different models on different datasets.

Table 8. *Comparison of the results with similar studies in the literature.*

Dataset	***Feature Selection Method***	***Machine Learning Algorithm***	***Accuracy (%)***	***Ref.***
Real internet traffic data	*Forward Selection Algorithm*	*K-Means* *Laplacian SVM*	*Average 90.0*	*[8]*
Their own dataset	-	*SVM*	*Average 90.0*	*[9]*
KDD (2009)	-	*k-NN, SVM, MLP, DT, NB, AdaBoost*	*Average 95.6*	*[10]*
Their own dataset	*PCA*	*RF* *SGM* *EGM*	*86.4* *85.5* *87,2*	*[11]*
Their own dataset	-	*MLP* *SAE* *CNN*	*96.0* *98.0* *98.0*	*[12]*
PCAP	-	*LSTM* *CNN+LSTM*	*Average 95.0*	*[13]*
Their own dataset	-	*SVM* *NB* *NCC*	*92.3* *96.8* *91.02*	*[14]*
Our Dataset	*PCA*	***k-NN*** *SVM* *MLP* *Decision Tree* *NB*	***99.4*** *88.14* *98.8* *89.8* *56.8*	

4. CONCLUSION

Classification of traffic flows in today's IP networks has become a significant research area with the adoption of Machine Learning methods and Software Defined Network principles. Due to the dynamic and encrypted nature of the existing traffic, traditional traffic classification methods that involve the identification of traffic based on port number and load can be inadequate.

In this study, traffic classification was made using different machine learning models on SDN architecture. The real-time network packets were captured on the SDN architecture created closely similar to the real environment. Network traffic was classified using five different Machine Learning models (k-NN, SVM, MLP, DT and NB) used for classification problems today. The working time and classification performance of the models were compared. k-NN has been the most successful model with an accuracy of 99.4%. The MLP model has come next with a 98.8% performance rate. The results obtained have been promising for future studies on traffic prioritization and QoS.

Various issues need to be considered for further studies. The network environment used should be at the most similar confusion level to real environments, and also the factors affecting the network service quality, such as scalability and availability, should also be taken into account. Besides, since most of today's internet traffic consists of video and game applications, the application is planned to be developed to include high-density traffic packages. Models such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN) or CNN are also aimed to be used in future studies.

REFERENCES

[1] Tahaei, Hamid, et al. "The rise of traffic classification in IoT networks: A survey." Journal of Network and Computer Applications 154 (2020): 102538.

[2] Nguyen, Thuy TT, and Grenville Armitage. "A survey of techniques for internet traffic classification using machine learning." IEEE communications surveys & tutorials 10.4 (2008): 56-76.

[3] Dehghani, Fereshte, et al. "Real-time traffic classification based on statistical and payload content features." *2010 2nd International Workshop on Intelligent Systems and Applications*. IEEE, 2010.

[4] P. Barlet-Ros Co-Advisor and J. Solé-Pareta in, "Network Traffic Classification: From Theory to Practice Valentín Carela-Español," no. October, 2014.

[5] A. Mestres *et al.*, "Public Review for Knowledge-Defined Networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 2–10, 2017.

[6] F. Ieee *et al.*, "Software-Defined Networking : A Comprehensive Survey," vol. 103, no. 1, 2015.

[7] OpenFlow Switch Specification v1.1.0. Available online: http://archive.openflow.org/documents/openflowspec- v1.1.0.pdf, Erişim Tarihi Ağustos, 20, 2019.

[8] P. Wang, S. C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," in *Proceedings - 2016 IEEE International Conference on Services Computing, SCC 2016*, 2016, pp. 760–765, doi: 10.1109/SCC.2016.133.

[9] D. Rossi and S. Valenti, "Fine-grained traffic classification with Netflow data," *IWCMC 2010 - Proc. 6th Int. Wirel. Commun. Mob. Comput. Conf.*, pp. 479–483, 2010, doi: 10.1145/1815396.1815507.

[10] L. He, C. Xu, and Y. Luo, "VTC: Machine learning based traffic classification as a virtual network function," *SDN-NFV Secur. 2016 - Proc. 2016 ACM Int. Work. Secur. Softw. Defin. Networks Netw. Funct. Virtualization, co-located with CODASPY 2016*, pp. 53–56, 2016, doi: 10.1145/2876019.2876029.

[11] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *Proceedings - International Conference on Network Protocols, ICNP*, 2016, vol. 2016-December, doi: 10.1109/ICNP.2016.7785327.

[12] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: Deep learning based encrypted network traffic classification in SDN home gateway," *IEEE Access*, vol. 6, pp. 55380–55391, 2018, doi: 10.1109/ACCESS.2018.2872430.

[13] H. K. Lim, J. B. Kim, K. Kim, Y. G. Hong, and Y. H. Han, "Payload-based traffic classification using multi-layer LSTM in software defined networks," Appl. Sci., vol. 9, no. 12, 2019, doi: 10.3390/app9122550.

[14] M. M. Raikar, S. M. Meena, M. M. Mulla, N. S. Shetti, and M. Karanandi, "Data Traffic Classification in Software Defined Networks (SDN) using supervised-learning," Procedia Comput. Sci., vol. 171, no. 2019, pp. 2750–2759, 2020, doi: 10.1016/j.procs.2020.04.299.

[15] D. Manual, A. Botta, W. De Donato, A. Dainotti, S. Avallone, and A. Pescap, "D-ITG 2.8.1 Manual," pp. 1–35, 2013.

[16] Scikit-learn Tutorials https://scikit-learn.org/stable/user_guide.html Erişim Tarihi Ağustos, 20, 2019.

[17] S. Shekhar and H. Xiong, "Nearest Neighbor," Encycl. GIS, vol. I, pp. 771–771, 2008, doi: 10.1007/978-0-387-35973-1_862.

[18] Nello Cristianini and John Shawe-Taylor. An introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, New York, NY, 1999