Mansoura Engineering Journal

Volume 47 | Issue 3 Article 9

9-27-2022

Network Slicing Based on Real-Time Traffic Classification in Software Defined Network (SDN) using Machine Learning

Aya A. El-serwy
Researcher of Electronics and Communications Engineering Department, Faculty of Engineering. Mansoura University, 35516 Mansoura City, Egypt, ayaelserwy@mans.edu.eg

Eman AbdElhalim
Assistant Professor of Electronics and Communications Engineering Department, Faculty of Engineering. Mansoura University, 35516 Mansoura City, Egypt, eman_haleim@mans.edu.eg

Mohamed A. Mohamed
Professor of Electronics and Communications Engineering Department, Faculty of Engineering. Mansoura University, 35516 Mansoura City, Egypt, mazim12@mans.edu.eg

Recommended Citation
A. El-serwy, Aya; AbdElhalim, Eman; and A. Mohamed, Mohamed (2022) "Network Slicing Based on Real-Time Traffic Classification in Software Defined Network (SDN) using Machine Learning," *Mansoura Engineering Journal*: Vol. 47 : Iss. 3 , Article 9.
Available at: https://doi.org/10.21608/bfemu.2022.261455

Mansoura University
Faculty of Engineering
Mansoura Engineering Journal

Network Slicing Based on Real-Time Traffic Classification in Software Defined Network (SDN) using Machine Learning

Aya A. El-serwy*, Eman AbdElhalim and Mohamed A. Mohamed

KEYWORDS:
Network slicing, Software defined networks (SDNs), Traffic classification, machine learning.

Abstract— **With the increase in smart devices, performance of traditional networks is limited by this huge amount of generated traffic flows. A scalable and programmable networking solution can be achieved in software defined networks (SDNs) through the separation between the control plane and the data plane. This advantage can allow machine learning (ML) applications to control and automate networks. Concurrently, network slicing (NS) is a promising technology. It is necessary to meet the variety of service needs and requirements. It provides the network as a service (Naas). So, combining NS and ML in SDNs can achieve good network resources management. This paper focuses on applying real-time network traffic analysis to assign each traffic to its suitable network slice according to traffic flows classification. In the proposed model, robust scale is used to scale the features instead of max/min normalization. Also, the k-means clustering algorithm is used to separate the dataset into the optimum number of different clusters (slices). Five different supervised models are applied to achieve high classification accuracy. The highest accuracy that can be obtained from feed-forward artificial neural network is (98.2%), while support vector machine (SVM) with linear function gives an accuracy of (96.7%). The challenges faced are collecting data from SDN's controller to apply real-time traffic flow classification, which is a primary step to assign each flow to its suitable network slice (Bandwidth).**

I. INTRODUCTION

WITH the increase in the number of smart devices, the restrictions on delay, security, bandwidth and user experience have increased. Networks are becoming more complex and dynamic, and this has forced network operators to find effective methods to manage the networks. Therefore, the design of network architecture that can manage the heterogeneity and maximize resources utilization efficiently is a serious issue [1]. Several solutions have been offered at this point. Two examples of these solutions are network slicing and machine learning. When these ideas are combined, they can apply intelligent and automatic management for network resources. Intelligence must be built in network devices for ease of organization, accurate management and optimization of resources. The traditional design of the network makes it difficult to control network

Received: (04 April, 2022) - Revised: (10 June, 2022) - Accepted: (18 June, 2022)

****Corresponding Author:** Aya A. El-serwy, Researcher of Electronics and Communications Engineering Department, Faculty of Engineering. Mansoura University, 35516 Mansoura City, Egypt (e-mail: ayaelserwy@mans.edu.eg)*

Eman AbdElhalim, Assistant Professor of Electronics and Communications Engineering Department, Faculty of Engineering. Mansoura University, 35516 Mansoura City, Egypt (e-mail: eman_haleim@mans.edu.eg)

Mohamed A. Mohamed, Professor of Electronics and Communications Engineering Department, Faculty of Engineering. Mansoura University, 35516 Mansoura City, Egypt (e-mail: mazim12@mans.edu.eg)

devices by ML. The software defined network framework allows networking devices to have intelligence built into them. The new SDN paradigm conceptually centralizes the controlling component and separates the data plane and the control plane from network devices. Instead of relying on network equipment vendors, the controller has a global/overall view of the network and allows the network operator to implement their own policies. This paradigm with its underlying data gathering systems offers adaptability, scalability and programmability to network. Figure 1 clarifies the difference between traditional network and SDN.

OpenFlow [2] is the standard southbound application programming interface (API) protocol for SDNs. It sets a way of communication between the software-based control plane and the hardware-based data plane.

ML is used to handle complicated issues without the need for explicit programming by allowing the algorithm to model and learn the basic behavior from a training dataset or environment. It has shown effective results in a variety of fields, including network management [3]. On the other hand, NS divides the infrastructure of the network into separated slices where each slice has different performance requirements and resources. This allows sharing the same physical network infrastructure and maximum utilization of the limited resources which are the two most important advantages of NS.

In this research, a solution for network slicing is introduced by a ML-based traffic classification model. The proposed architecture uses supervised and unsupervised learning algorithms to achieve accurate detection of the suitable slice for each traffic. Previous results of this work appeared in [4] but the steps of pre-processing weren't enough to achieve higher classification accuracy in ML models.

Therefore, the major contributions to this research are:

(i) using the robust scaler, which scales the features using statistics that are robust to outliers, instead of min/max scaler, (ii) real time detection:, the proposed controller's application is able to classify each traffic flowing through each switch connected to the controller without the need to run the application for every traffic flow, (iii) extracting more features from the controller, and (iv) use more evolution metrics to assess the performance of ML models used in classification.

The remainder sections are arranged as the following, section II discusses relevant work from similar studies that use ML for traffic analysis and network slicing. Section III provides an overview of the system environment and framework. The proposed model is described in section IV along with the dataset that was utilized in this research. Section V presents performance evaluation and validation metrics. Section VI presents the experimental results along with their analysis. Section VII represents the implementation of the ML model on SDN's controller, and finally section VIII presents the paper's conclusions and suggested future work

II. RELATED WORK AND MOTIVATIONS

Classifying network flows is a prerequisite for network slicing to perform successful management of the network. Several techniques for traffic classification have been proposed and utilized in communication systems over a long time.

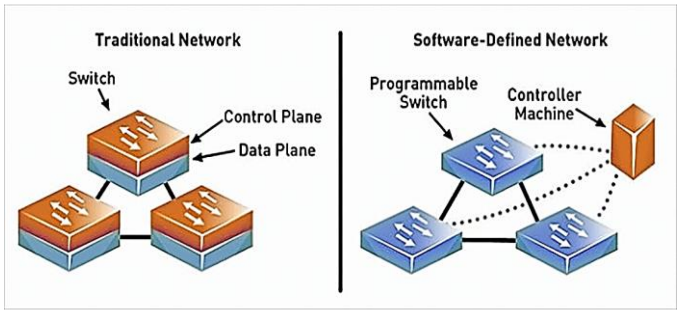


Figure 1: Traditional network Vs software defined network (SDN).

A. Classification based on port IP

One of the most used approaches in the past was the port-based classification, and it was to some degree effective as several applications used constant port numbers set by the Internet Assigned Numbers Authority (IANA). However, some drawbacks to this strategy have become evident over time, as many of modern applications have unregistered port numbers and use dynamic ports, like peer-to-peer (P2P) applications [5]. Thus, the port-based techniques are no longer helpful. Multiple evaluation studies in recent years on port-based approaches have shown that they are not effective. Zander et al. [6] found that 30-70% of the traffic flows they analyzed can't be defined by port-based techniques.

B. Classification based on Payload IP (DPI)

Recently, one of the most used techniques is the payload method, also identified as deep packet inspection (DPI). By inspecting the content of the packet, DPI identifies the application and yields better classification results. It presents two main challenges as reviewed by Valenti et al [7]. First, it consumes resources, as identifying a pattern within a packet has a high computational cost. So, Li et al. [8] developed a traffic classification model called multi-classifier, this model combined DPI with ML. The multi-classifier model gave high priority to ML. DPI was only used when ML results are unreliable. DPI had the second priority because it consumes resources. Second, it is unable to recognize encrypted traffic [9], which is becoming common these days.

C. Machine Learning Classification (Ml)

As of late, ML based traffic classification methods have been broadly utilized to alleviate limitations forced by conventional classification techniques.

The previous research with the most similar context to this research is presented by Perera et al [4], they used ”IP Network Traffic Flows Labeled with 75 Apps” dataset to implement a network slicing model based on traffic classification in SDN by using simple topology of mininet and RYU controller. K-means was used to cluster dataset into four different classes. Before supervised learning was applied, features were selected as the following: source and destination media access control address

(MAC) addresses, source and destination and port addresses, flow bytes count, flow packets count, average packet size and flow duration. Five supervised ML models were used, the highest accuracy was 96.7% and obtained from SVM. Although several operations were applied to clean the data such as removing duplicated instances and removing rows which contained missing values, using max/min normalization was not suitable as it isn't robust to outliers and there is no explanation for how data was captured from SDN controller.

An open access VPN-Non_VPN (ISCXVPN2016) dataset was used by trang et al [10] to build a network traffic classification model. A two-scenarios testing process was created. The encrypted traffic was first classified into two main/general classes: VPN and non-VPN traffic. The traffic may then be classified into seven classes (browsing, email, streaming, chat, file transfer, VoIP and Trap2p) for each general class. The traffic was classified using Random Forest, K-Nearest Neighbors, and Artificial Neural Networks algorithms. The performance of the ML model was assessed using accuracy, precision, recall, and the F1-score. The results showed that random forest can achieve the highest accuracy, about 92% in the first scenario and about 90% in the second scenario, but, it was not accurate in the long-time data samples. Also, there is no explanation of which features were chosen for the training process.

Aouedi et al [11] proposed a model to select the best features to apply network traffic classification by using"IP Network Traffic Flows Labeled with 75 Apps" dataset. Recursive Features Elimination (RFE) was used to select the best features. The results were compared on top 10 features, top 15 features and top 25 features out of 87 features in the dataset. The 15-features set was selected and the top three classifiers having the best accuracy are XGBoost, Random Forest, and AdaBoost with 89.09%, 85.49%, and 84.57%. The authors didn't pay attention to the problem of class imbalance and also no solution for handling outliers was provided.

Wang et al [12] described a slicing prototype based on SDN and home devices. Three different slicing strategies were presented. However, the ML approach was not employed in this study since the authors manually selected home user needs based on the home network's restrained resources. As a result, this solution cannot scale well.

Raikare et al [13] used supervised learning to classify their internal dataset generated from SDN topology. POX controller and simple mininet topology were used for traffic generating. The generated dataset contained only three types of traffic: streaming, browsing and e-mail, but the software package" net-mate" was used to obtain the flow statistics and no techniques for selecting features have been employed.

Software-defined networks, machine learning, big data, network function virtualization (NFV) and network slicing for 5G were all introduced by Le et al [14]. They presented an architecture for classifying network traffic and used the results to slice the network into three slices through SDN environment. K-means clustering algorithm with (k = 3) was used to group the dataset into three different clusters. Five classification models were used including Naive Bayes, Support Vector Machine, Tree Ensemble, Random Forest and Neural Network. The network slicing model was tested on YouTube traffic with its default bandwidth (1 Mbps). Results showed that the model classified YouTube traffic to the third slice which had a higher bandwidth (3 Mbps). So, user played the video with high quality and without any dropped frames. But, the technique used for selecting the best features was not specified and no criterion was used to select the optimum number of clusters.

Kwon et al [15] presented a model for network traffic classification based on ML algorithms. The traffic was collected from SDN/NFV environment, which included ONOS controller and simple topology from mininet with two hosts and one OpenVswitch (OVS). Although results between different ML algorithms for two different network scenarios were compared, there weren't any pre-processing steps applied on the data and class imbalance problem affected a lot on the resilts.

III. SYSTEM ENVIRONMENT: AN OVERVIEW

This section provides a simple overview of the system's environment used to implement the ML model in SDN. The framework of the system is presented in figure 2, which consists of the following:

(i) the application plane which contains all the software applications running on the controller.

(ii) the control plane, represented by the controller which is the basic component of the SDN environment, and this allows centralized control automation and intelligent management across physical and logical networks. Collecting flow statistics, extracting the features and classifying the traffic are performed through controller via our software classification application.py. Northbound APIs are used to transfer information between the controller and the programmed applications in application plane. So, SDN seems as single logical network device, and (iii) the data plane which contains network devices which are used to forward packets only. Southbound APIs like open flow protocol transfer the data between the controller and the network devices in the data plane. Table 1 represents our system configuration.

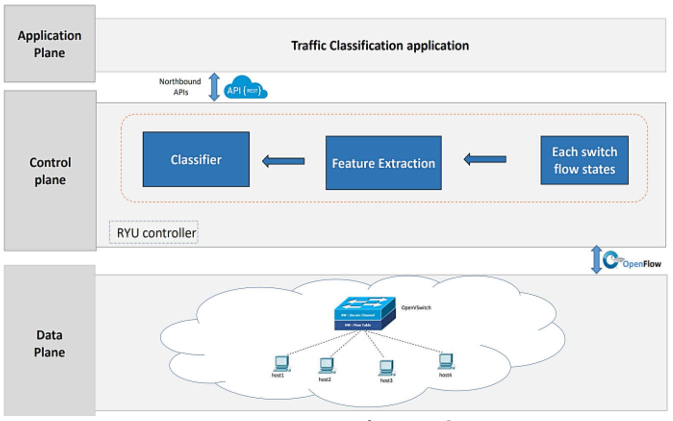


Figure 2: system framework.

TABLE 1
SYSTEM CONFIGURATIONS

Environment	VMware
Emulator	Mininet (2.2.2)
Operating System	Ubuntu (20.04.2)
controller	RYU(4.30)

IV. PROPOSED MODEL

The proposed model consists of six stages as shown in figure 3. The dataset pre-processing is performed in the first stage. In the second stage, suitable features are selected from the dataset to train the models. Then, an unsupervised ML model is employed to cluster the data into separated clusters, each cluster has its own characteristics so, the optimum number of clusters represents the available number of slices in the network. Then, five supervised ML models are used to classify the flows into one of the clusters and this in turn will assign this flow to its suitable slice. After training the models, the trained models will be tested by the testing part from the dataset and this represents the offline test. The model which will achieve the highest accuracy in the offline test, will be used to apply a real-time test on the traffic flows that will be generated through the SDN created topology.

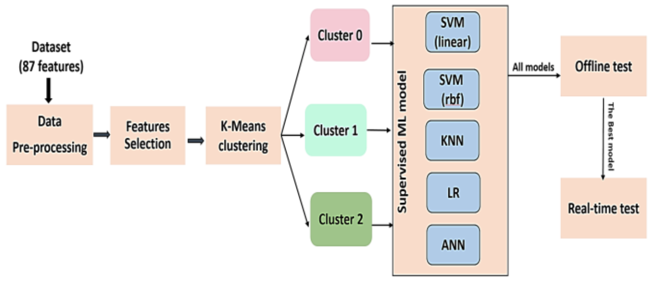


Figure 3: Steps of the proposed model

A. Data Description

For this research, dataset “IP Network Traffic Flows, Labeled with 75 Apps” from Kaggle [16] is used. This data was recorded in a network department of Universidad Del Cauca, Popayán, Colombia by applying packet capturing over six days at morning and afternoon hours. It contains 87 features for a total of 3,577,296 instances. But, just a small fraction of this dataset is required for this research (3600 flows). Each instance is a flow between a source and a destination. It is primarily designed for application classification as it contains flows of 75 different applications such as YouTube, Facebook, Twitter, etc. Although it is a real world traffic dataset generated from a TCP/IP-based network, it is suitable for our proposed model because SDN also uses TCP/IP so all the header fields will be identical. This dataset will be used for training and offline evaluating the performance of the ML model. Mininet based SDN environment will be used for generating some test traffic for real-time classification testing.

B. Data Pre-Processing

Several pre-processing steps must be completed before the data is used to train ML models. These steps are as the following: (i) duplicated instances should be removed to avoid biasing in the model, (ii) ML models can’t handle non-numerical data and hence two python functions are used, label encoder and ip2int. Label encoder is used to encode features containing non-numeric values such as protocol name. While ip2int is used to convert source-IP and destination IP, and (iii) it is important to normalize/scale the range of values of the feature attributes in the dataset before training the ML models. This step is necessary because all dimensions of feature vectors should be in the same range. This made the convergence of ML models faster during the training process. There are many normalization and scaling techniques that can be used but, in the proposed model, robust scaler is used as it is suitable for input variables containing outliers. This method depends on Inter Quartile Range (IQR) [17]. Robust data scaling is represented in Equation (1)

$$x(scaled) = \frac{\mathrm{x - median}}{P75 - P25} \quad (1)$$

where x is the value before scaling, median is (50th percentile), P75 is 75th percentile and P25 is 25th percentile. The denominator represents IQR. And, (iv) Class balance is improved by using NearMiss algorithm [18] to obtain balance classes.

C. Feature Selection

The target of our application is to use the best model to implement real-time classification in SDN so, the model must be trained only on the features that can be generated from the controller. Table 2 represents all the features that can be extracted from the controller. These features can be extracted directly from SDN’s controller without using any open source tools like CICFlowmeter [19] to extract the features, as these tools are inappropriate for applying real-time classification. To extract these features from the controllers, just a simple flow monitor function (a function that can monitor open-flow switch statistical information) was added to the controller’s application. By using this function, the controller will send openflow states/metrics request to every switch connected with it. This request is to collect statistics about each flow occurs in the switch. The switch replies with each flow statistics through an openflow states reply thus, features can be extracted. Using this function enabled us to generate limited number of features as mentioned in table 2 and can’t generate all the features existing in the dataset. So, for real-time test, models must be trained only on features that can be extracted from the controller and also existing in the dataset. These features are flow.bytes.s, flow duration, flow.packets.s, protocol, average packet size, total.fwd.packets and total.bwd.packet. Also, source and destination IP addresses, source and destination port address can be extracted from the controller but, they can’t be used due to the environment difference between dataset (real-network) and generated traffic (mininet emulator) which will be used in the real-time test.

TABLE 2
EXTRACTED FEATURES FROM SDN CONTROLLER BY USING OPENFLOW STATES REPLY

FEATURE NAME	DESCRIPTION
Duration	Total flow duration in both forward and reverse direction in seconds
Source IP	The source IP-address of the flow
Destination IP	The destination IP-address of the flow
Protocol	protocol name of the flow
Source port	The number of the source port
Destination Port	The number of the destination port
Fwd.byte.count	Number of bytes in the forward direction
Total.fwd.packet	Total number of packets in the forward direction
Fwd.bytes.rate	Bytes rate in forward direction
Fwd.packet.rate	Packets rate in the forward direction
Fwd.avg.bytes.rate	Average bytes rate in the forward direction
Fwd.avg.packet.rate	Average packet rate in forward direction
Bwd.byte.count	Number of bytes in the backward direction
Total.pwd.packet	Total number of packets in the backward direction
Bwd.bytes.rate	Bytes rate in backward direction
bwd.packet.rate	Packet rate in backward direction
Average.packet.size	The average size of each packet
Flow.bytes.s	Total number of bytes in both direction divided by flow the duration
Flow.packets.s	Total number of packets in both direction divided by flow the duration

D. Unsupervised Learning

With the increase in complexity and the size of data, unsupervised learning has become very important. It seeks to detect relations between the inputs without knowing information about the output characteristics. Clustering algorithms are one of the commonly used unsupervised learning approaches. These algorithms are utilized to group the input data into different clusters based on how similar they are. Instances in the same cluster have a higher degree of similarity as compared to instances in the other clusters. There are many available partition-based clustering algorithms [20]. K-means algorithm [21] is the most often used technique. It is selected because it is simple, fast, easy to implement, scalable and able to cluster big datasets efficiently. Its goal is to divide the data into K clusters by measuring the similarity. There is a category center in each cluster µk. The Euclidean metric is chosen as the similarity criteria. It is calculated by minimizing the total sum of squares of distances between points in each cluster and the cluster center µk. Equation (2) represents the objective function:

$$A = \sum_{i=1}^{K} \sum_{n=1}^{N} | X_i - \mu k|^2 \tag{2}$$

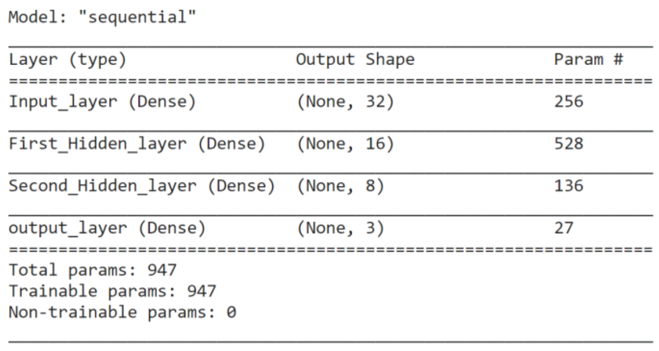
where X_i denotes the clustering sample, µk is the center of the cluster, K represent the number of clusters and N is the number of samples.

E. Network traffic classification

After applying K-means clustering, each traffic is labeled based on its cluster. Then, five supervised learning models from scikit- learn library including Support Vector Machine (SVM) with linear function [22], SVM with Radial Basis Function (RBF) kernel [23], logistic Regression (LR) [24], K-nearest Neighbors (KNN) and feed-forward artificial neural network (ANN) are trained and evaluated. The new labeled data is separated into two parts, training part and testing part with a 70%: 30% ratio. All of the five models are trained by the training part of the dataset separately. Table 3 represents the algorithms and hyperparameter values used and figure 4 represents the structure of the feedforward ANN model.

TABLE 3
MODELS HYPERPARAMETERS

Algorithms	Hyperparameters
Logistic regression	penalty='l2', solver='sag', C=.6
SVM(linear)	kernel='linear', C=.01, gamma='auto'
SVM(RBF)	kernel='rbf', C=.1, gamma='auto'
KNN	n_neighbors=3



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input_layer (Dense)          (None, 32)                256
_________________________________________________________________
First_Hidden_layer (Dense)   (None, 16)                528
_________________________________________________________________
Second_Hidden_layer (Dense)  (None, 8)                 136
_________________________________________________________________
output_layer (Dense)         (None, 3)                 27
=================================================================
Total params: 947
Trainable params: 947
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: The structure of the ANN model

V. PERFORMANCE EVALATION AND VALIDATION

Reliability and validity are the two primary factors that need attention after an ML model is trained and tested. Reliability refers to the amount of trust we have on the model to deliver consistent outcomes in similar situations, it is also called precision of the model. On the other hand, the accuracy of the model on the test data, or how excellent the results are, is called validity. Recall is the ratio of correctly classified samples to true positive values, and F1_score, the harmonic mean of precision and recall, are two common metrics that can be used to evaluate the quality of the model also. These metrics [25] (precision, accuracy, recall, and F1-Score) mainly depend on four outcomes: True Positives, True Negatives, False Positives and False negatives, which come from the confusion matrix as shown in figure 5. These metrics are respectively and mathematically defined in Equations (3), (4), (5) and (6).

$$Precision = \frac{TN}{TP + FP} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100\% \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F1.score = \frac{TP}{TP + 0.5(FP + FN)} \tag{6}$$

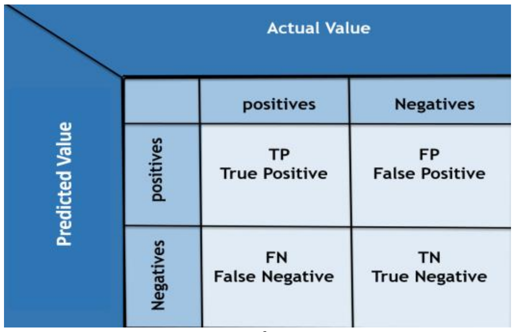


Figure 5: Confusion Matrix

For evaluation, Cohen’s Kappa [26] is also used. It is a statistical metric that indicates the reliability of two raters who are rating the same quantity and identifies the degree of agreement between the two raters. Equation (7) represents Cohen’s Kappa for multi-class cases

$$K = \frac{C * S - \sum_{k}^{K} Pk * tk}{s^2 - \sum_{k}^{K} Pk * tk} \quad (7)$$

where C is the total number of correctly predicted samples, S is the total number of samples, Pk is the number of times that class k was predicted (column total) and tk the number of times that class k truly occurs (row total). This metric is more suitable than the accuracy in case of imbalanced classes.

VI. RESULTS AND DISCUSSION

In this section, results from the proposed model will be discussed and analyzed. As K-means clustering requires the number of clusters, the Davies-Bouldin Index (DBI), which is an evaluation metric for clustering algorithms [27], is used to determine the optimum number of clusters by measuring inter and intra-cluster distances. Three clusters, which also represent three network slices, are used as K=3 gets the lowest Davies-Bouldin score about 0.28 which will achieve the highest clustering accuracy as presented in figure 6. Figure 7 determines the number of flows in each cluster and shows an imbalanced class distribution. To avoid this problem and overfitting too, only 1200 flows for each cluster were chosen to solve class imbalance problem and were enough to train the model.

Table 4 represents the classification accuracy of the proposed model compared to the traditional model and shows higher accuracy for the proposed model than traditional one. This is due to the usage of the suitable algorithm for scaling the data and solve the problem of imbalance class distribution. Table 4 also shows that ANN achieved the highest accuracy 98.2%. In the proposed model, tree-based algorithms such as decision tree and random forest weren’t used because they aren’t affected by scaling. Figure 8 illustrates the calculated training accuracy and loss of the ANN model graphically. The figure shows that the accuracy increases continuously while the mean squared error for loss decreases during the ten training epochs. After the 8th epoch, the ANN model converged, indicating that the model was fitted well by the dataset and the fine-tuned parameters. Figure 9 represents the precision, recall, f1_score and kappa score for each model. The figure shows that ANN and SVM with linear function gave outstanding results and gave better results than other models in all metrics. The confusion matrices for all the models are represented in figure 10. From this figure we can deduce that even though the ANN model can classify the second and the third cluster (slice) with high accuracy of 99.1% and 99.7% respectively, it has some confusion to classify the first cluster only 95.9%.

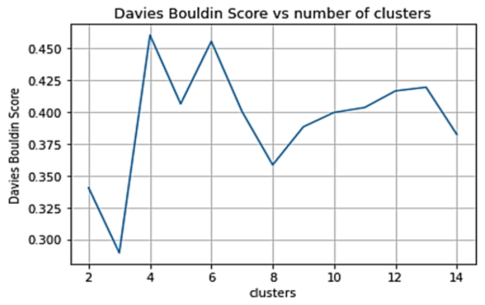


Figure 6: Davies Bouldin score vs different number of clusters

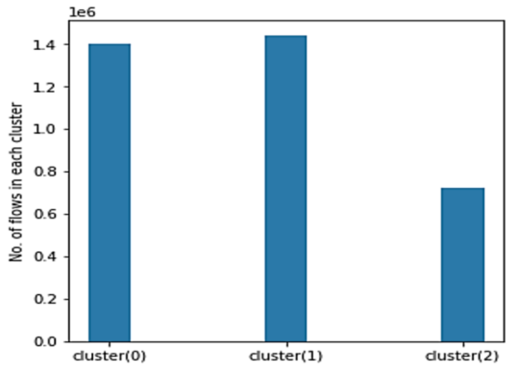


Figure 7: Number of flows in each cluste

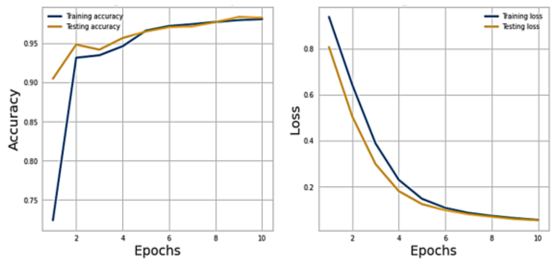


Figure 8: accuracy and loss values for training and testing during 10 epochs

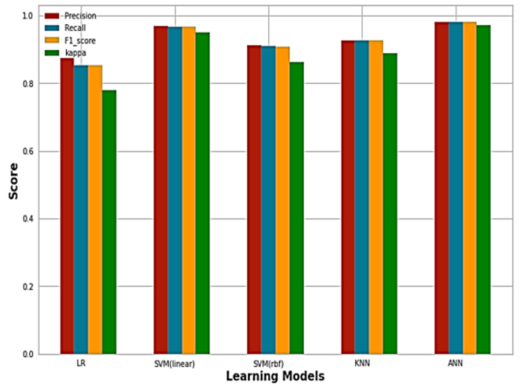
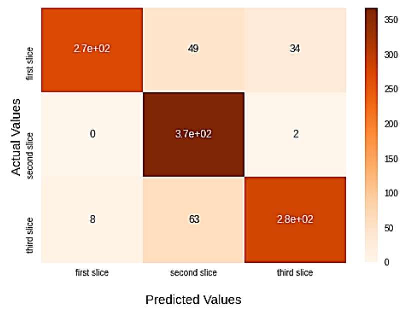


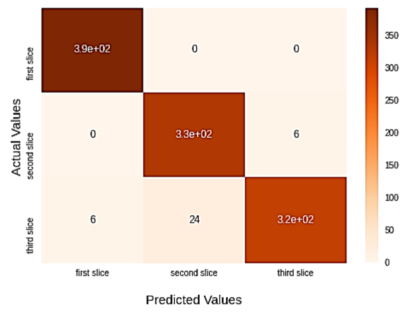
Figure 9: Precision, recall,f1_score and Cohen’s Kappa for different models

TABLE 4
COMPARE ACCURACY BETWEEN TRADITIONAL AND THE PROPOSED MODEL

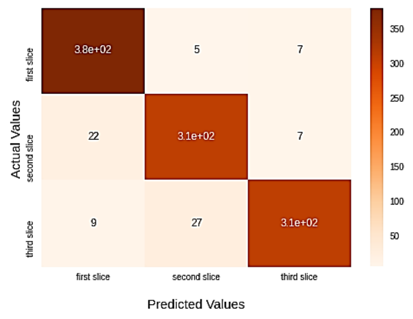
Models	**Proposed Model Accuracy**	**Traditional Model Accuracy** [4]
SVM (Linear)	96.7%	96.37%
SVM (RBF)	90.7%	70.4%
KNN	92.5%	71.4%
Decision Tree	-	95.76%
Logistic Regression	85.5%	-
Random Forest	-	94.9%
ANN	98.2%	-



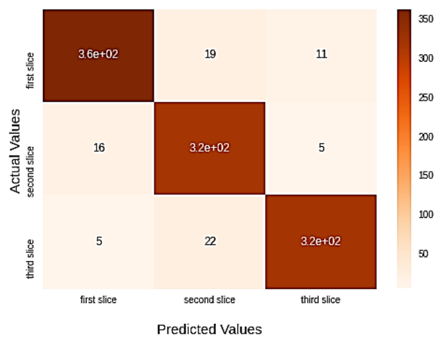
(a)



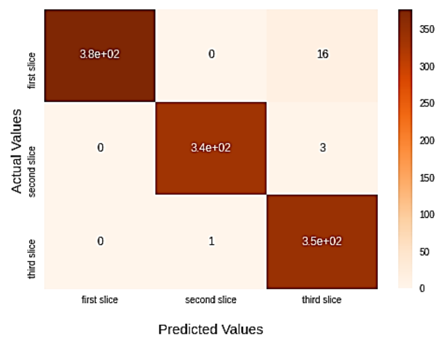
(b)



(c)



(d)



(e)

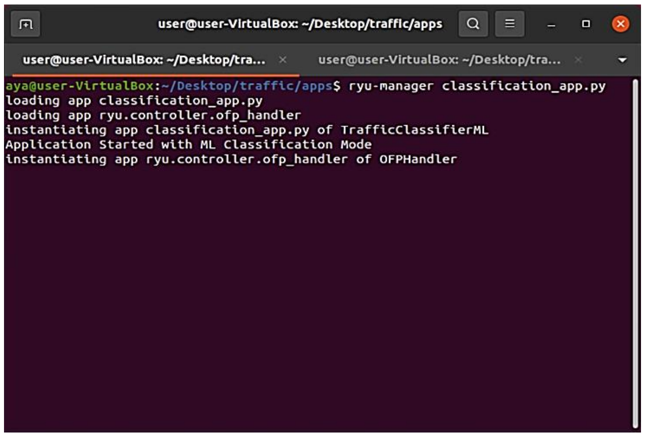
Figure 10: (a) confusion matrix of Logistic regression (b) confusion matrix of SVM with linear function (c) confusion matrix of SVM with RBF kernel (d) confusion matrix of KNN (e) confusion matrix of ANN.

Based on the previous evaluations, the ANN model will be used to implement the real-time test in the created SDN topology.

VII. IMPLEMENT CLASSIFICATION MODEL IN SDN

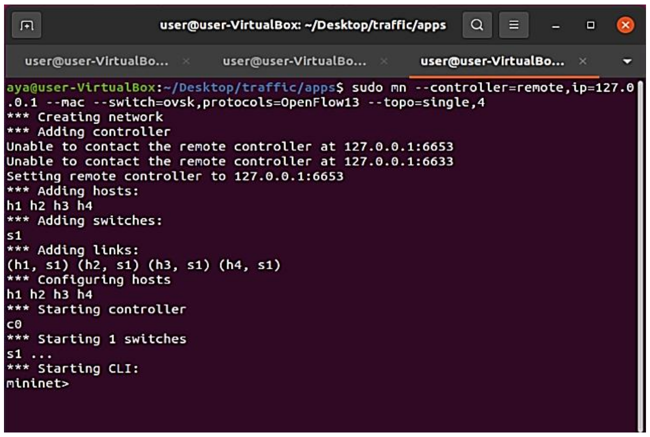
Applying ML with SDN can achieve great results. As the ANN model achieved the highest classification accuracy, it is used to implement real-time traffic classification on the SDN topology. For the Implementation, a simple virtual network is created on Mininet [28]. A simple topology is used with four hosts, one openVswitch, and one RYU controller. We created a python script can be run on the RYU controller. Starting from the simple-switch.py script we modified it to achieve our target. This python application is able to classify each flow that occurs in the network. The application is tested by generating DNS traffic using the DITG tool [29] and figure 11 clarifies the steps we followed to apply the real-time test on the created SDN topology. First, ryu-manager was used to run the application on the controller then, connecting the mininet topology (simple topology) with the controller through '6653' port. As soon as the connection was established, ten DNS traffic flows were generated between h1 and h2, figure 12 shows the configurations to generate the traffic between h1(sender) and h2 (receiver).

Besides the result, we also displayed some additional information about the flow such as source_IP, destination_IP and protocol (17 refers to UDP) in figure 11. Finally, the figure shows the ability of our application to classify the DNS traffic to the first slice (cluster).



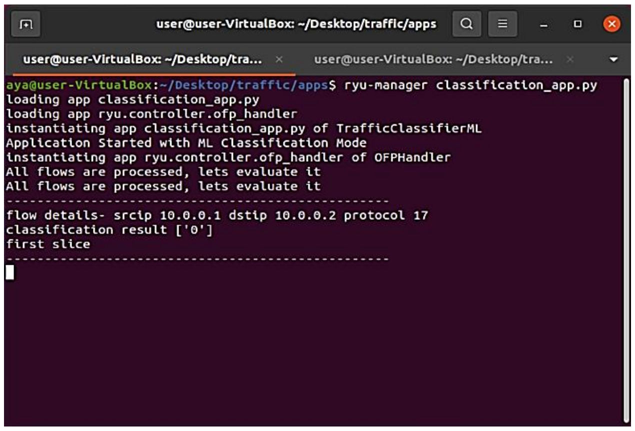
```
user@user-VirtualBox: ~/Desktop/traffic/apps
user@user-VirtualBox: ~/Desktop/tra...   user@user-VirtualBox: ~/Desktop/tra...
aya@user-VirtualBox:~/Desktop/traffic/apps$ ryu-manager classification_app.py
loading app classification_app.py
loading app ryu.controller.ofp_handler
instantiating app classification_app.py of TrafficClassifierML
Application Started with ML Classification Mode
instantiating app ryu.controller.ofp_handler of OFPHandler
```

(a)



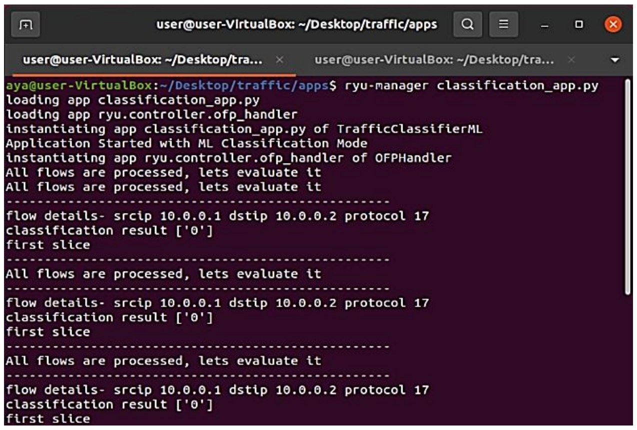
```
user@user-VirtualBox: ~/Desktop/traffic/apps
user@user-VirtualBo...   user@user-VirtualBo...   user@user-VirtualBo...
aya@user-VirtualBox:~/Desktop/traffic/apps$ sudo mn --controller=remote,ip=127.0
.0.1 --mac --switch=ovsk,protocols=OpenFlow13 --topo=single,4
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

(b)



```
user@user-VirtualBox: ~/Desktop/traffic/apps
user@user-VirtualBox: ~/Desktop/tra...   user@user-VirtualBox: ~/Desktop/tra...
aya@user-VirtualBox:~/Desktop/traffic/apps$ ryu-manager classification_app.py
loading app classification_app.py
loading app ryu.controller.ofp_handler
instantiating app classification_app.py of TrafficClassifierML
Application Started with ML Classification Mode
instantiating app ryu.controller.ofp_handler of OFPHandler
All flows are processed, lets evaluate it
All flows are processed, lets evaluate it
--------------------------------------------------
flow details- srcip 10.0.0.1 dstip 10.0.0.2 protocol 17
classification result ['0']
first slice
--------------------------------------------------
```

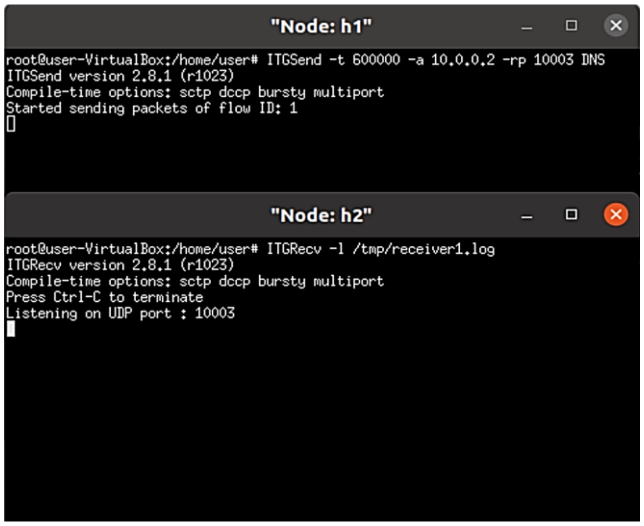
(c)



```
user@user-VirtualBox: ~/Desktop/traffic/apps
user@user-VirtualBox: ~/Desktop/tra...   user@user-VirtualBox: ~/Desktop/tra...
aya@user-VirtualBox:~/Desktop/traffic/apps$ ryu-manager classification_app.py
loading app classification_app.py
loading app ryu.controller.ofp_handler
instantiating app classification_app.py of TrafficClassifierML
Application Started with ML Classification Mode
instantiating app ryu.controller.ofp_handler of OFPHandler
All flows are processed, lets evaluate it
All flows are processed, lets evaluate it
--------------------------------------------------
flow details- srcip 10.0.0.1 dstip 10.0.0.2 protocol 17
classification result ['0']
first slice
--------------------------------------------------
All flows are processed, lets evaluate it
--------------------------------------------------
flow details- srcip 10.0.0.1 dstip 10.0.0.2 protocol 17
classification result ['0']
first slice
--------------------------------------------------
All flows are processed, lets evaluate it
--------------------------------------------------
flow details- srcip 10.0.0.1 dstip 10.0.0.2 protocol 17
classification result ['0']
first slice
```

(d)

Figure 11: (a) running our application on SDN controller using RYU manager (b) connecting the simple topology to the SDN controller (c) processing all flows through the switch and identify first flow (d) identify real time flows.



```
"Node: h1"
root@user-VirtualBox:/home/user# ITGSend -t 600000 -a 10.0.0.2 -rp 10003 DNS
ITGSend version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
Started sending packets of flow ID: 1

"Node: h2"
root@user-VirtualBox:/home/user# ITGRecv -l /tmp/receiver1.log
ITGRecv version 2.8.1 (r1023)
Compile-time options: sctp dccp bursty multiport
Press Ctrl-C to terminate
Listening on UDP port : 10003
```

Figure12: Generate DNS traffic between mininet hosts, h1 (sender) and h2 (receiver) using D-ITG tool.

VIII. CONCLUSIONS AND SUGGESTED FUTURE WORK

This research has been performed to prove that integrating ML with SDN can achieve high performance, especially for network traffic slicing via traffic analysis. It is clear to observe that classification of traffic using ML algorithms can deliver good results through SDN environment, which will improve or replace traditional networking administration. This is achievable due to the ability of programmability. This research shows how ML can be used to apply intelligent traffic detection and also construct intelligent network slices. To improve the efficiency of the ML models and their ability to classify the traffic, more pre-processing steps (using balanced classes and robust scalar instead of min/max scaler) are used. After applying K-means clustering, five different models were used in classification stage including SVM (linear), SVM (RBF), Logistic Regression, KNN and finally ANN which achieves accuracy higher than 98%. For future work, each flow should be assigned to a suitable bandwidth value by using meter tables and flow tables of the OpenFlow, usage of more complicated topology instead of the simple topology to be close to real network. Also, the controller application can be modified to have two modes: one for ML classification and the other to collect data from network which will help to train model using real-time dataset through SDN. Finally, only five ML models were used for classification. However, there might be other models that can fit this problem well.

AUTHORS CONTRIBUTION:

The following is a summary of the authors' contributions to the paper based on their relevant roles:

1. ***Aya A. Elserwy***: finding suitable dataset, data interpretation and analysis, system development and implementation, software, methods and techniques, paper writing.
2. ***Eman AbdElhalim***: supervision, software, and validation.
3. ***Mohamed A. Mohamed***: work conception and design, supervision, software, validation, methodology, editing, review and final approval of the version to be published.

FUNDING STATEMENT:

No research grants were received.

DECLARATION OF CONFLICTING INTERESTS STATEMENT:

The author confirmed that there are no possible conflicts of interest regarding the research authorship or publication of this article.

IX. References

[1] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li and J. Rao, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open Journal of Vehicular Technology,* vol. 1, pp. 45-66, 2020.

[2] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *Proceedings of the 9th central & eastern european software engineering conference in russia*, 2013.

[3] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications,* vol. 9, pp. 1-99, 2018.

[4] M. Perera Jayasuriya Kuranage, K. Piamrat and S. Hamma, "Network traffic classification using machine learning for software defined networks," in *International Conference on Machine Learning for Networking*, 2019.

[5] N. Duffield, M. Roughan, S. Sen and O. Spatscheck, *Statistical, signature-based approach to IP traffic classification,* Google Patents, 2011.

[6] S. Zander, T. Nguyen and G. Armitage, "Automated traffic classification and application identification using machine learning," in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) l*, 2005.

[7] A. Valenti, A. Silvio "Reviewing traffic classification," in *Data Traffic Monitoring and Analysis*, Springer, 2013, pp. 123--147.

[8] Y. a. L. J. Li, "MultiClassifier: A combination of DPI and ML for application-layer classification in SDN," in *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*, IEEE, 2014, pp. 682--686.

[9] R. a. Y. X. Liu, "A survey on encrypted traffic identification," in *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies}*, 2020, pp. 159--163.

[10] K. Trang , A. Nguyen "A Machine Learning-based Approach for Network Traffic Classification," *Knowledge Engineering and Data Science,* vol. 4, no. 2, 2022.

[11] O. Aouedi, K. Piamrat, S. Hamma and J. K. Perera, "Network Traffic Analysis using Machine Learning: an unsupervised approach to understand and slice your network," *Annals of Telecommunications,* pp. 1-13, 2021.

[12] S. Wang, X. Wu, H. Chen, Y. Wang and D. Li, "An optimal slicing strategy for SDN based smart home network," in *2014 International conference on smart computing*, 2014.

[13] 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 Computer Science,* vol. 171, pp. 2750-2759, 2020.

[14] P. Tung and D. Sinh, "SDN/NFV, Machine Learning, and Big Data Driven Network Slicing for 5G," in *2018 IEEE 5G World Forum (5GWF)*, 2018.

[15] J. Kwon, D. Jung and H. Park, "Traffic Data Classification using Machine Learning Algorithms in SDN Networks," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020.

[16] J. S. Rojas, Á. R. Gallón and J. C. Corrales, "Personalized service degradation policies on OTT applications based on the consumption behavior of users," in *International Conference on Computational Science and Its Applications*, 2018.

[17] J. S. Rojas, A. Pekar, Á. Rendón and J. C. Corrales, "Smart user consumption profiling: Incremental learning-based OTT service degradation," *IEEE access,* vol. 8, pp. 207426-207442, 2020.

[18] S.-J. Yen and Y.-S. Lee, "Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset," in *Intelligent Control and Automation*, Springer, 2006, pp. 731-740.

[19] R. Kaushik, M. Dave "Malware Detection System Using Ensemble Learning: Tested Using Synthetic Data," in *Data Engineering and Communication Technology*, springer, 2021, pp. 153--164.

[20] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks,* vol. 16, pp. 645-678, 2005.

[21] A. K. Jain, M. N. Murty and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR),* vol. 31, pp. 264-323, 1999.

[22] T. Joachims, "Training linear SVMs in linear time," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[23] S. Han, C. Qubo and H. Meng, "Parameter selection in SVM with RBF kernel function," in *World Automation Congress 2012*, 2012.

[24] M. Maalouf, "Logistic regression in data analysis: an overview," *International Journal of Data Analysis Techniques and Strategies,* vol. 3, pp. 281-299, 2011.

[25] R. Padilla, S. Netto "A survey on performance metrics for object-detection algorithms," in *2020 international conference on systems, signals and image processing (IWSSIP)*, IEEE, 2020, pp. 237--242.

[26] M. Grandini, E. Bagli, G .Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

[27] B. Aouedi, "Performance evaluation of feature selection and tree-based algorithms for traffic classification," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops*, IEEE, 2021, pp. 1--6.

[28] K. Kaur, J. Singh and N. S. Ghumman, "Mininet as software defined networking testing platform," in *International Conference on Communication, Computing & Systems (ICCCS)*, 2014.

[29] S. Avallone, S. Guadagno, D. Emma, A. Pescapè and G. Ventre, "D-ITG distributed internet traffic generator," in *First International*

Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings., 2004.

TITLE ARABIC:

تقسيم الشبكة الي شرائح استنادًا إلى تصنيف حركة الاشارات داخل الشبكة المعروفة بالبرمجيات (SDN) باستخدام التعلم الآلي.

ARABIC ABSTRACT

مع زيادة الأجهزة الذكية، أصبحت الشبكات التقليدية أقل فاعلية في إدارة هذا العدد الهائل من حركة الاشارات المتولدة خلالها. يقدم الفصل بين مستوى التحكم ومستوى إعادة التوجيه في الشبكات المعرفة بالبرمجيات (SDN) حلاً للشبكات قابل للبرمجة وقابل للتطوير يمكّن تطبيقات التعلم الآلي (ML) من تحكم وميكنة الشبكات، وفي الوقت نفسه يعد تقسيم الشبكة تقنية واعدة لأنها توفر الشبكة كخدمة (NaaS)، لذلك حظي تقسيم الشبكة (NS) باهتمام كبير من كل من الصناعة والأوساط الأكاديمية حيث أنه يوفر تحكم جيد في موارد الشبكة، لذا فإن الجمع بين (NS) و(ML) في (SDNs) سيحقق إدارة جيدة للموارد. عند تطبيق تحليل بيانات الشبكة مع شريحة الشبكة الموضوعية وفقًا لتصنيف تدفق حركة الاشارات في النموذج المقترح، تم استخدام (Robust scaler) لتوسيع نطاق المميزات بدلاً من(min/max) ، تم استخدام خوارزمية (K-Means) لفصل البيانات الي العدد الأمثل من الشرائح المختلفة وأيضًا يتم تطبيق خمسة نماذج مختلفة خاضعة للإشراف وذلك لتحقيق تصنيف بدقة عالية.

أعلى دقة تم الحصول عليها من الشبكة العصبية الاصطناعية (98.2%)، يلي هذا SVM بدالة خطية (96.7%).

تتمثل التحديات التي تمت مواجهتها التحديات في جمع البيانات من وحدة تحكم (SDN) لتطبيق خوارزمية تصنيف الاشارات، وهي خطوة أساسية لتعيين الشريحة المناسبة لكل إشارة (النطاق الترددي)