SERBIAN JOURNAL OF ELECTRICAL ENGINEERING
Vol. 18, No. 2, June 2021, 237-254

UDC: 621.397:004.72 **DOI: https://doi.org/10.2298/SJEE2102237I**

Classification of Network Traffic Using Supervised Machine Learning Algorithms Within NFV Environment

Gjorgji Ilievski[1], Pero Latkoski[2]

Abstract: Deep Packet Inspection (DPI) of the network traffic is used on a regular basis within the traditional and virtualized environments. But changes in the network architecture with the introduction of containers, microservices, application functions, network functions, and the penetration of 5G access technology are adding more traffic complexity, especially in the so-called east-west flow direction. Network Functions Virtualization (NFV) has become an unavoidable step for further IP network development. In this context, DPI is becoming a challenge. Furthermore, the penetration of 5G allows access of various kinds of devices to the network with cloudification logic which drives them. This paper provides a performance analysis of a selected set of supervised machine learning (ML) algorithms for classification of network traffic within an NFV environment. The goal is to find a suitable algorithm that will classify the traffic from a point of both precision and speed, especially because in the 5G networks any packet delay may compromise the quality of service requirements. The research shows that out of the 6 algorithms tested, Decision Tree algorithm has the best overall performance, from both classification precision and time consumption point of view. It has proved as a reliable classifier that is performing evenly across different classes. Due to the specifics of the virtualized environments and encryption methods, payload data, source, destination, and port information of the network traffic packets are excluded from any statistical operation used for classification by the ML algorithms.

Keywords: Classification, Machine Learning, Network Functions Virtualization, Network traffic

1 Introduction

The classification of the network traffic has been a necessity since the beginning of the computer networks. Nevertheless, the network architectures are

[1]Gjorgji Ilievski is with IT Department, division for Cyber Security within Makedonski Telekom AD Skopje, Kej 13-ti Noemvri 6, 1000 Skopje, RN Macedonia; E-mail: gjorgji.ilievski@telekom.mk

[2]Pero Latkoski is with the Telecommunications Institute of Faculty of Electrical Engineering and Information Technologies, Ss. Cyril & Methodius University, Rugjer Boshkovikj 18, Postal box 574, Skopje, RN Macedonia; E-mail: pero@feit.ukim.edu.mk

changing continuously, especially now when Virtual Machines (VM), Software Defined Networking (SDN), private, public, and mixed clouds are common in the IT world. The trend is now moving towards microservices, containers, application functions, network functions in Network Functions Virtualization (NFV) environments [1], which is adding more complexity to the network flows. In such scenario, majority of the network traffic is moving in the cloud, usually within the same datacenter, in the east-west direction. This traffic never leaves the virtual plane and is often managed by the SDN elements in the NFV environment, which obstructs the capture or any other operation over the traffic. This is important both for the cloud operators and for entities using the services provided by the public clouds. Operations which are common practice and are considered trivial, such as implementing Quality of Service (QoS), network security, optimization, application management and monitoring are becoming a challenge.

In this paper, we are performing an experimental test to reveal the network traffic classification efficiency of several supervised machine learning (ML) algorithms. We have created a unique test environment that reassembles real life processes and simulates the east-west traffic in the virtual plane among virtual hosts where NFV is established. The efficiency of the ML algorithms is explored from a point of classification precision, but also from a point of calculation speed. This is very important when we take into consideration the penetration of 5G, because it is tightly integrated with the mentioned cloudification and the new technologies used in it. For example, 5G specification calls for user plane latency of just 1ms for ultra-reliable low-latency communications (URLLC) [2]. This is why the speed of the ML algorithm is crucial and must be performed in a manner that will minimize the expected latency added by the classification.

The study we have conducted provides a novel scenario that is comparable to the emerging architectures where NFV and 5G are implemented. It involves 6 different supervised ML algorithms: Bayes Net, NaiveBayes, J48, K-Nearest Neighbors (K-NN), Decision Tree, and AdaBoost because they are widely used in the traditional computer networks, are proven to be reliable while providing a valid classification, and are not expensive to be implemented in practice. We have used Weka [3] as a tool for classification.

The classification is made based on 6 classes, which are chosen by experience in traditional networks, and in alignment with the expected network traffic within the 5G radio, as well as the 5G core network. We have chosen 6 classes: VoIP, encrypted VoIP, DNS, Management, SSH, HTTP, and HTTPS traffic. NFV architecture is becoming the real 5G enabler, providing the ability to place initial workloads in the network, and allowing it to grow into the edge, thus providing the needed basis for the IoT expansion expected with the 5G penetration. The main goal and idea of our work is to present a method for creating datasets based on statistical characteristics of the network traffic, and then testing machine learning algorithms based on the resulting datasets, within a scenario based on an NFV architecture. Here we want to emphasize the VoIP and encrypted VoIP classifications which are crucial for QoS capabilities in 5G networks, allowing smart connectivity, with the ability to steer, secure and break out network traffic.

There is a variety of works that are using ML algorithms to perform packet inspection [4 – 7]. What distinguishes them is the novel experimental testbed, as well as the approach to classify the network data based only on statistical parameters of the packets and the packet flows, without the use of source and destination addresses (both MAC and IP addresses), without any examination of the payload, and without the communication ports.

The encrypted network traffic is in a rapid rise. Significant number of services and applications are using encryption as a primary method of securing information. But this has made traffic classification a challenge. The solution for traffic classification that we propose is applicable in practice without compromising privacy and data integrity, it provides an insight into the performance of supervised ML algorithms, and determines which one is the most suitable for NFV based environment.

There are also many examples of ML algorithms used for Deep Packet Inspection (DPI) in traditional networks [3, 8, 9]. Compared to these works, we focus on virtualization and the NFV environment. In such a scenario, the network packets are mostly moving in the east-west direction and are often encrypted, so classical DPI is impossible to be conducted. In our approach, it is not important whether the payload is encrypted or not. Also, the legal aspect of performing DPI in a cloud environment (especially public one) is satisfied, as the data carried within the payload is not compromised. We are using only statistical features of the network packets and the network flows to create datasets that are later used for training and testing of the ML algorithms. During the testing phase, we are evaluating the efficiency of the algorithm from a point of precision, but also from a point of speed. Network traffic is sniffed inside an open vSwitch directly. We are not introducing additional probe or SDN element to capture the traffic. We consider all network traffic, between the virtual elements inside the environment, but also the traffic that is used for management of the environment (including the one form the controllers), as well as the traffic that is going in and out to the internet. This is a realistic scenario with most cloud solutions in practice.

Besides the precision, the ML algorithm speed in many cases is even more important. If the time consumed to classify the data is adding significant latency in the network traffic and is consuming resources (CPU cycles, memory) of the cloud, the classification precision loses its relevance.

In the reminder of the paper we will go through the related work on the subject, briefly explained in the next section. The experimental setup and the dataset creation are explained in Section 3, while the results are analyzed in Section 4. Section 5 is reserved for the conclusion and our plans for future work.

2 Related Work

Many researches are focused on the DPI aspects in scenarios involving SDN elements [10 – 12]. Others are researching the security aspects when performing DPI [13, 14] by using SDN probes for network traffic sniffing and data processing. Our work distinguishes in terms of the NFV-based setup, while targeting a complete isolation of the packet payload. Some authors consider the classification of network traffic in traditional networks [15, 16] without tackling the specifics of the trendy virtualization, which on the other hand is an important aspect of our work.

Mohammad Reza Parsaei et al. [17] are using SDN to categorize traffic by application, by applying different variants of Neural Network estimator. They are using data mining techniques based on different ML algorithms and are proposing a controller that could dynamically allocate bandwidth on network flows and optimize resource allocation. They achieve classification accuracy of over 97%. Distinct to our work, they are using source and destination IPs, as well as the transport layer port for classification.

In [18], QoS in an SDN based network is being researched with an accent on overcoming the limitations of traditional networking architectures. Different flow routing mechanisms are categorized. In our research, we are exploring classification as a basic concept from which QoS can benefit significantly.

Reference [4] is a study where NFV environment is prepared to classify different types of TCP traffic using three supervised ML algorithms: NaiveBayes, Bayes Net, and J48. Network packets are analyzed individually resulting in three different datasets: traditional, virtual, and combined in order to compare the classification performance. In our case, we use TCP and UDP based traffic and we analyze the statistical parameters of the packet flows within an NFV environment that closely reassembles to cloud platforms.

Luong-Vy Le et al. [19] applied big data, ML algorithms, SDN, and NFV to build a practical and powerful framework for clustering, forecasting, and managing traffic behaviors for a huge number of base stations with different statistical traffic characteristics of different types of cells (GSM, 3G, 4G). The framework is intended to be used for developing future 5G Self-Organizing Network (SON) applications. Several applications based on traffic forecasting were also introduced. Five ML algorithms are used to classify the traffic generated by the mobile applications, with QoS implemented to enable bandwidth guarantees. The conclusion is that from the selected algorithms, Decision Tree has the best overall performance. Our experiment is bound on the transport network layer with an aim to classify the traffic that is mostly east-west based using ML algorithms, but also to evaluate the time needed for classification which is crucial for the future 5G environments.

Alshammari et al. [5] is focused on VoIP traffic within traditional networks. Data is extracted from an existing network environment with a complex topology. The authors evaluate classification of both encrypted and unencrypted VoIP using three ML algorithms: C5.0, ADA Boost, and GP Classifier, and using a subset sampling technique. In the experiments, C5.0 had the best performance and the highest precision rate. In our case, cloud-based environment with NFV implemented is used to benchmark the ML algorithms with various types of network traffic.

In [20], Machine Learning classification of multi-service internet traffic is used to evaluate resources consumption (CPU time and usage of system memory). We are complementing this research, as we are evaluating the ML algorithms time needed to perform the classification.

Shu et al. [21] propose network traffic classification based on deep learning network structure. The experimental dataset is created from ten types of data, each one abstracted from a complete TCP bidirectional stream containing 249 network flow attributes. Google's TensorFlow deep learning framework is used in the experimental environment. NaiveBayes and Decision Tree ML algorithms are used to compare the classification efficiency in respect to the deep learning network. Compared to this work, we are targeting six different supervised ML algorithms classification, having in mind that not only classification precision, but also the time needed to perform the classification is important, as any delay added to the network packet speed can produce a functional problem in the environment.

The effect of NFV elements placement on the network traffic, especially on the increase or decrease of the volume of the processed traffic, is researched in [22]. The authors develop an algorithm that determines the flow path and then propose a Least-First-Greatest-Last routing.

Bonfiglio et al. [23] are researching the traffic specifics of Skype as an application that is based on encrypted VoIP for voice calls. The traffic is explored in real time, with two different approaches by using the statistical parameters of the traffic generated by Skype. The approaches are then assessed using flow correlation.

To summarize, our testing setup is similar to that introduced in [4], with additional elements added to the environment, such as virtual machines connected to internet and virtual network elements with bridged IP addresses. Both TCP and UDP traffic is generated, with and without encryption. The classification groups and labels are chosen in a manner that various traffic is classified. Viber and Skype are used to generate VoIP traffic, whereas scripts are used to open ssh management sessions to different hosts. Furthermore, a novel testbed is proposed in context of 5G and usage of NFV elements within the virtualized environment which is expected in a real-life setup. The network packets are analyzed directly within the virtual switch, without the use of a probe or an SDN element. Statistical characteristics are extracted from TCP and UDP packet flows and used to perform further analysis.

The next section shows the details of the experimental environment and the creation of the datasets used to train and to test the six supervised ML algorithms.

3 Experimental Setup and Dataset Creation

To simulate the east-west traffic within a virtualized NFV based network, the experimental environment is based on Oracle VirtualBox [24], which is installed on a single physical host with Ubuntu 18.04 Server. All elements are connected with Open vSwitch (OVS) [25, 26] that provides the network connectivity. The switch is connected to the internet through the host in a bridge mode. All network packets flow through the OVS switch, the packets inside the environment, and the packets to and from internet. We are capturing the traffic directly on the OVS using Wireshark and tshark [27].

Mininet [28] is used as a network simulator. There are two different installations on two separate virtual machines, each with different network topology having 100 hosts, 20 switches and links among them and to the OVS. The hosts within the simulated networks are having private IP addresses and are able to communicate with each other. GRE tunneling is used to link the two simulated Mininet networks. Some of the hosts within Minines have NAT-ed IP addresses and are able to communicate to internet.

Ryu Controller [29] is used to control the simulated Mininet networks. It is installed and configured in a separate virtual machine.

There are four other virtual machines connected to the OVS that are also used for traffic generation. Skype and Viber are installed onto them to simulate the VoIP traffic. When initiated, VoIP needs access to internet, but after that peer-to-peer communication can be observed within the OVS in a completely east-west direction. Script that starts ssh sessions is enabled on the VMs. Python script that starts ssh sessions to the Mininet hosts was developed, as well. The SSH sessions were started in intervals that followed Poisson distribution.

Distributed Internet Traffic Generator (D-ITG) [30] generates various TCP and UDP traffic among the hosts within Mininet. Different scripts are used to generate traffic at packet level, replicating appropriate stochastic processes for both IDT (Inter Departure Time) and PS (Packet Size) random variables.

Fig. 1 is an overview of the experimental setup, showing the components symbolically.

We have made 50 different experiments to generate various traffic (using D-ITG, Skype, Viber, custom scripts) and to analyze it. The experiments were conducted in time intervals from 4 to 20 minutes in which VoIP calls lasted from 10 seconds to 10 minutes, following Poisson distribution. One dataset per experiment was generated. Different D-ITG scripts for different traffic simulation were used in every experiment. The scripts used different Mininet hosts and different paths in every try. The average number of captured packets was 1.262.375 and the average number of flows was 4090.

We have made specific classification of the traffic, using classes that are commonly used, based on experience from the traditional networks. As it will be shown in the results, the classification precision was calculated as an overall, but also for every class independently, in order to calculate the macro-average precision in which every class contribution to the precision is treated equally (as the number of packets and flows varies for every class).

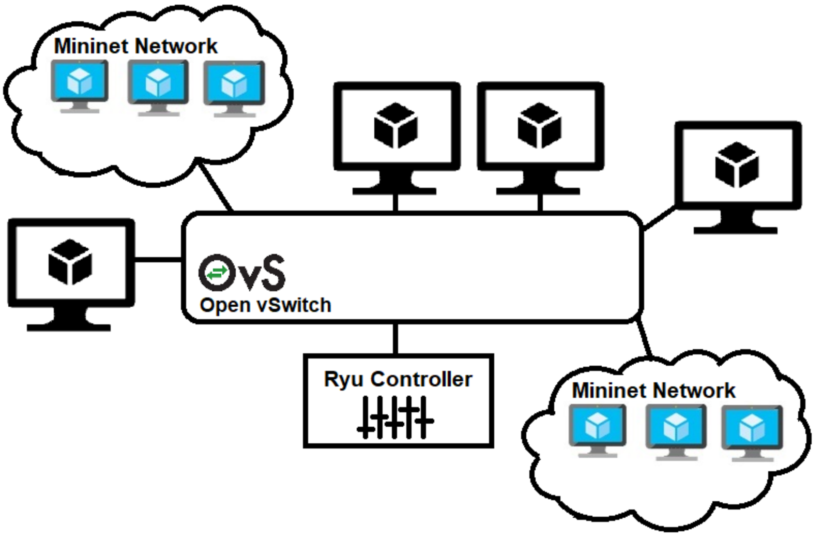

Fig. 1 – *Experimental Environment.*

We used the following labels for the classes: DNS – for all the traffic used for name resolution, NETMGMT – all traffic used for hosts and network management, SSH – for the ssh sessions in the environment, WEB – for HTTP and HTTPS traffic, VOIP – for VoIP traffic, SVOIP – for encrypted VoIP.

From the generated Wireshark pcap files, UDP and TCP packet flows, and the classes used for ML training and latter for test precision and confirmation are identified using Argus [31]. Similar to [5], we define a flow as a bi-directional connection between two hosts. TCP flows are terminated either by flow time-out or by connection tear-down, whereas UDP flows are ended by flow time-out. When we observe the flows within the OVS, it can be seen that most of the traffic is east-west based, inside the virtual layout and between the hosts, but also the flows from the management generated by the hypervisor and the Ryu controller are detected. Because our focus is an NFV based environment, some of the flow features are not taken into consideration, such as the source and destination IP and MAC address, as well as the communication port that can vary inside the virtual environment.

To train and to test the supervised ML algorithms, we have used Weka [3, 32]. 2/3 of every dataset was used for training, while 1/3 was used for testing each of the algorithms. As not all the attributes have the same contribution to the classification, the AttributeSelectedClassifier with Ranker as an attribute ranking algorithm was used. InfoGainAttributeEval was used as an evaluator that determines the gain of information that the attributes carry. With this approach we are ranking the attributes that are used for the algorithms after which the information gain of every attribute is evaluated. This approach prevents a possible data leakage.

Table 1
Flow Attributes.

No.	Abbrevation	Feature
1	proto	transaction protocol
2	rate	packets per second
3	srate	source packets per second
4	drate	destination packets per second
5	sintpkt	source interpacket arrival time
6	dintpkt	destination interpacket arrival time
7	sjit	source jitter
8	djit	destination jitter
9	mdoffset	mean of the data offset values of the packets in the flow.
10	smeansz	mean of the flow packet size transmitted by the source
11	dmeansz	mean of the flow packet size transmitted by the destination
12	smaxsz	max packet size for source
13	dmaxsz	max packet size for destination
14	sminsz	min packet size for source
15	dminsz	min packet size for destination

Based on experience from traditional networks and with careful observation of the gained datasets, we have selected the attributes given in **Table 1**, as features that characterize the flows. The payload is not used for reason of privacy within cloud environments and usage of different encryption methods that will make the payload irrelevant for the classification. The labels in transport layer header (e.g., the port numbers) are also not used as they can be easily changed. Short explanation of each of the selected attributes is provided inside the table.

The following section provides the results from the testing of the supervised ML algorithms and the analysis.

4 Results and Analysis

To create 50 datasets, we have performed as many experiments and all 6 supervised ML algorithms were tested. The performance of each algorithm is a combination of its precision and the time needed to make the classification. Because the time consumption is correlated to the performance of the machine where the analysis is performed, all classification tasks were performed on the same machine with a careful observation of all the processes on the machine that can influence the performance. A mean value of the 50 results was derived for all target metrics.

True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (TN) rates are defined as:

– TP is number of instances that are truly identified of a class.

– FP is number of instances that are falsely identified of a class.

– TN is number of instances that are truly identified that are not of a class.

– FN is number of instances that are falsely identified that are not of a class.

The overall precision of the algorithms is calculated as the proportion between TP instances and all instances in the dataset [32]:

$$Precision = TP/(TP + FP). \quad (1)$$

Table 2
Algorithm Precision.

No.	ML Algorithm	Precision
1	AdaBoost	0.744±0.0292
2	BayesNet	0.9672±0.0189
3	J48	0.9906±0.0027
4	KNN	0.9172±0.0438
5	NaiveBayes	0.8634±0.0170
6	Decision Tree	0.9914±0.0033

Table 2 shows the average precision of the algorithms in all 50 experiments with the statistical standard deviation across the experiments, as a weighted average value.

It can be seen that overall Decision Tree algorithm has the best precision. It is followed by J48 and BayesNet. On the other hand, AdaBoost algorithm has the worst overall performance with the lowest precision of 74.4%.

To further explore the precision, we have calculated the micro average precision, which aggregates the contribution of all classes and calculates the average metric, as given by (2):

$$Precision\_MIC = \frac{TP1 + TP2 + \cdots + TPN}{TP1 + FP1 + TP2 + FP2 + \cdots + TPN + FPN}. \tag{2}$$

The results are presented in **Table 3**.

Table 3
Algorithm Micro Average Precision.

No.	ML Algorithm	Micro Average Precision
1	AdaBoost	0.8450±0.0176
2	BayesNet	0.9954±0.0027
3	J48	0.9984±0.0006
4	KNN	0.9856±0.0073
5	NaiveBayes	0.9752±0.0027
6	Decision Tree	0.9984±0.0010

Not all classes have the same or similar number of packets and flows and the data distribution is skewed. To avoid data balancing problem and to come to valid conclusions we are calculating the macro average precision, the recall and the F-1 score.

Macro average precision is an average of precisions of each class. This means that every class will weigh the same in the macro average precision. The following equation is used to calculate the macro average precision (Precision_MAC), where $Pr1$, $Pr2$ etc., denote the precision of the algorithm regarding the individual classes. These results are shown in **Table 4.** In this table the statistical standard deviation is calculated for the precision between classes:

$$Precision\_MAC = \frac{Pr1 + Pr2 + \cdots + PrN}{Count(Pr)}. \tag{3}$$

If we evaluate **Table 4**, it becomes clear that the algorithms are not performing the same on all the classes. Decision Tree algorithm is the most constant with the highest macro average precision and the lowest standard deviation between classes, showing that it classifies all classes similarly. J48 is very close to Decision Tree, with over 98% precision. Opposite to them, AdaBoost algorithm shows very low macro average precision with high standard deviation, which means that it performs poorly on different classes. K-Nearest Neighbor algorithm is also underperforming, with just over 82% macro average precision. When we compare these results with the standard weighted precision in **Table 2**, we can see that the algorithms have the same order, but the macro precision of the lower end algorithms is worse, drawing the conclusion that AdaBoost and KNN have different precision for different classes.

Table 4
Algorithm Macro Average Precision.

No.	ML Algorithm	Macro Average Precision
1	AdaBoost	0.20335±0.3064
2	BayesNet	0.8899±0.1489
3	J48	0.9824±0.0148
4	KNN	0.82735±0.2202
5	NaiveBayes	0.78915±0.2048
6	Decision Tree	0.9848±0.0107

To evaluate the impact of the false negative classified instances, Recall is used as a model metric. It is the proportion of true positive instances and total actual instances:

$$Recall = \frac{TP}{TP + FN}. \tag{4}$$

We’ve used Recall to calculate the F1-score of the supervised ML algorithms in our experiments. It is a metric that balances between the precision and the recall, so that false negative instances are taken into consideration. F1-score is calculated as a harmonic mean of the precision and the recall:

$$F1 - Score = 2\frac{Precision \cdot Recall}{Precision + Recall}. \tag{5}$$

Table 5 shows the F1-score values calculated for our experiments.

Decision Tree ML algorithm has the best F1-score, followed by J48, BayesNet, KNN, NaiveBayes and AdaBoost. The last one has F1-score of only 23.2% with very high standard deviation.

The tables are visually represented in Figs. 2 – 5.

Table 5
F-1 Score.

No.	ML Algorithm	F1-score
1	AdaBoost	0.231575±0.3356
2	BayesNet	0.913425±0.1055
3	J48	0.975425±0.0212
4	KNN	0.797425±0.2295
5	NaiveBayes	0.782125±0.1510
6	Decision Tree	0.980475±0.0152

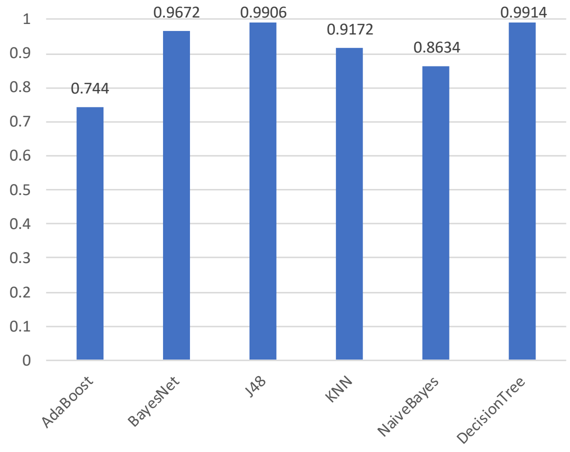

Fig. 2 – *Algorithm Precision.*

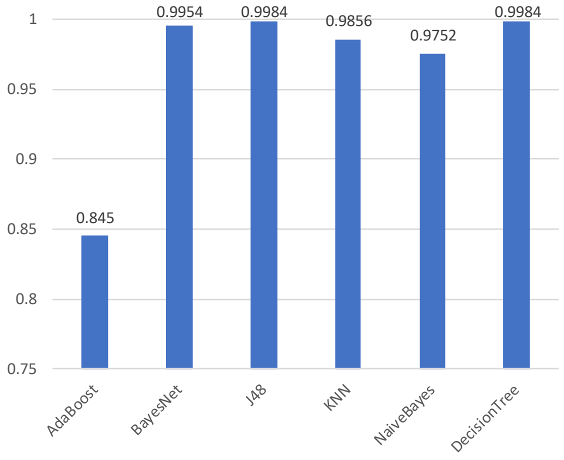

Fig. 3 – *Micro Average Precision.*

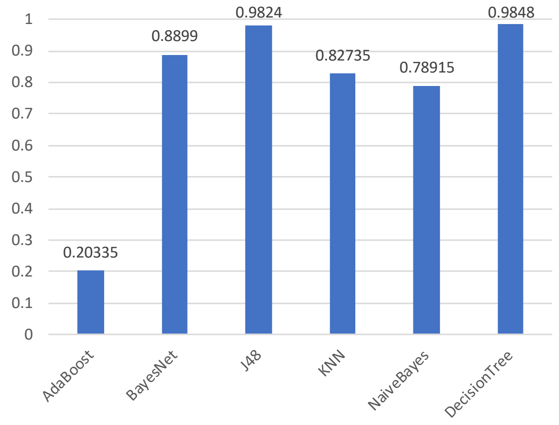

Fig. 4 – *Macro Average Precision.*

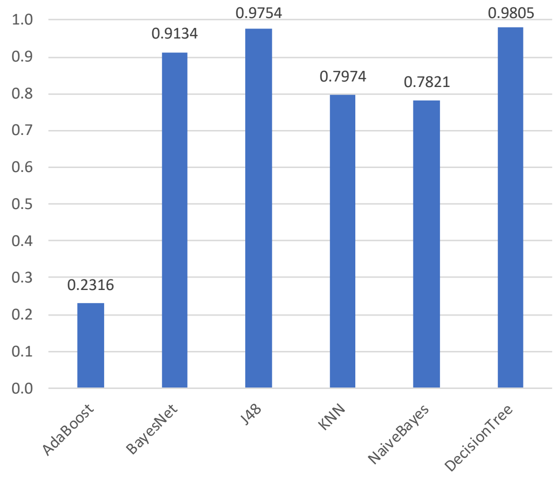

Fig. 5 – *F-1 Score.*

The algorithm precision is only the first feature to determine the usability of the algorithm. The second important aspect is the time needed to perform the classification. If the time needed for classification is too high, the process will add latency to the network communication, thus making the benefit of the classification too costly. This is important especially in the protocols where latency can degrade the service, such as VoIP. Furthermore, this is also crucial in the 5G scenarios, where latency is one of the major concerns. Another point is that if the time spent by the algorithm is high, more resources the process will consume. The two metrics (precision and time consumption) combined will show the overall performance of the algorithms.

The time that we have measured is relative to our testbed environment. All experiments are performed on a same environment, where special care has been taken to isolate all unnecessary processes. The average value of the time consumption was calculated from 50 experiments.

The algorithm precision is only the first feature to determine the usability of the algorithm. The second important aspect is the time needed to perform the classification. If the time needed for classification is too high, the process will add latency to the network communication, thus making the benefit of the classification too costly. This is important especially in the protocols where latency can degrade the service, such as VoIP. Furthermore, this is also crucial in the 5G scenarios, where latency is one of the major concerns. Another point is that if the time spent by the algorithm is high, more resources the process will consume. The two metrics (precision and time consumption) combined will show the overall performance of the algorithms.

The time that we have measured is relative to our testbed environment. All experiments are performed on a same environment, where special care has been taken to isolate all unnecessary processes. The average value of the time consumption was calculated from 50 experiments.

Table 6 shows the average time needed for the six supervised algorithms to perform the classification within the chosen 8 classes.

Table 6
Average time needed for classification (*in seconds*).

No.	ML Algorithm	Average time in seconds
1	AdaBoost	0.012
2	BayesNet	0.016
3	J48	0.022
4	KNN	0.272
5	NaiveBayes	0.104
6	Decision Tree	0.016

The results for the average time needed for classification show that AdaBoost algorithm performs the best, with a highest speed. Decision Tree and BayesNet share the second and third place being 25% slower than AdaBoost. J48 has also a satisfactory speed. NaiveBayes is almost 9 times slower than AdaBoost and more than 6 times slower than Decision Tree KNN algorithm is the slowest. Decision Tree and AdaBoost spend only 5.9% of the time needed by KNN to perform the classification.

Fig. 6 graphically represents the average time consumed by the algorithms.

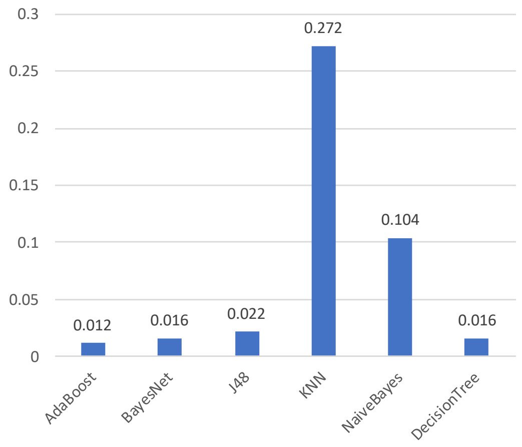

Fig. 6 – *Time needed for classification* (*in seconds*).

To summarize, when we take a look at both precision and time needed for classification, Decision Tree supervised ML algorithm has the best overall performance. Although AdaBoost is the fastest algorithm, the classification precision is poor and unsteady across different classes, which makes this algorithm unreliable for our scenario. J48 has also high precision that is evenly distributed among classes, but it is slower than Decision Tree and BayesNet. Nevertheless, its speed is in the scale of Decision Tree and BayesNet, and it is also a valid choice. BayesNet has a high precision, but the macro average precision and the F1-score show that precision distribution among classes is not as good as Decision Tree and J48.

NaiveBayes is in the middle from both precision and time perspective, while KNN algorithm has about 83% macro average precision and 80% F1-score, but it is by far the slowest algorithm which makes it useful only in cases where time needed for classification has low importance.

5 Conclusion and Future Work

The main goal and idea of our paper is to present a method for creating datasets based only on the statistical characteristics of the network traffic flows, and then to test machine learning algorithms performance based on the created datasets. All this is done in an experimental testbed where NFV architecture is used.

The efficiency of the algorithms is explored from a point of precision of the algorithm but also form a point of time consumption needed to perform the classification. This is important from a virtualization point of view, where mixed cloud scenarios are a common practice, but also for the incoming penetration of 5G, where the network latency is of high importance.

Our experimental testbed is used to perform multiple experiments and to collect network traffic data from which IP flows are extracted. The statistical features of the flows are used as attributes for classification. Because attributes such as source and destination IP and MAC addressed and communication ports can vary inside a virtualized environment, they are not taken into consideration. Due to encryption and data privacy concerns, the payload of the data packets is also excluded from the datasets and it is not used for classification.

The environment that we use is not introducing any kind of network probes or SDN elements to perform the data collection, so that east-west traffic is completely unchanged. The traffic is completely intercepted within the virtual layer where it naturally resides. This has also an impact on the resource consumption, minimizing the additional latency that can be added to the network packets by redirecting or port replication used in traditional DPI.

The results have shown that Decision Tree algorithm has the best overall performance, from both classification precision and time consumption point of view. It has proved as a reliable classifier that is performing evenly across different classes. J48 and BayesNet are also performing well, with J48 having slightly better precision and BayesNet being faster. K-Nearest Neighbour and NaiveBayes have an average classification precision in a range of about 80%, but they are slow, especially KNN which is almost 20 times slower than Decision Tree and BayesNet. AdaBoost shows the worst performance with precision that varies a lot among different classes, which can be seen from the macro average precision and the $F1$-score.

The analysis in our paper can be used in practice within multiple systems that are built on top of cloud environments. NFV elements are now unavoidable part of such infrastructures. 5G infrastructure is relying onto these types of systems, but also connectivity to such systems is most likely to be done through 5G access technology. In those examples performing QoS, network and application security, data management, system and process monitoring and control is depending on valid network traffic classification that has to be precise and fast without taking considerable amount of system resources.

For future work we are planning to evaluate the impact of the number of classes on the classification results and the time consumption of the supervised ML algorithms by introducing a large number of classes and reducing the classes. Another stream is to expand the experimental testbed to multiple hosts and distributed switches and to evaluate the network that is moving across multiple hosts.

6 References

[1] M. Chiosi, et. al, Microsoft word - Network Functions Virtualisation, Introductory White Paper, 2012, pp. 1 – 16.

[2] M. Eiman: Minimum Technical Performance Requirements for IMT-2020 radio interface(s), Presentation, Available at: https://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2020/Documents/S01-1_Requirements%20for%20IMT-2020_Rev.pdf

[3] I. Witten, E. Frank, M. Hall, C. Pal: Data Mining: Practical Machine Learning Tools and Techniques, 4th Edition, Morgan Kaufmann Publishers Inc., San Francisco, 2016.

[4] J. Vergara-Reyes, M.C. Martinez-Ordonez, A. Ordonezy, O.M.C. Rendon: IP Traffic Classification in NFV: A Benchmarking of Supervised Machine Learning Algorithms, Proceedings of the IEEE Colombian Conference on Communications and Computing (COLCOM), Cartagena, Colombia, August 2017, pp. 1 – 6.

[5] R. Alshammari, A. Nur Zincir-Heywood: Identification of VoIP Encrypted Traffic Using a Machine Learning Approach, Journal of King Saud University - Computer and Information Sciences, Vol. 27, No. 1, January 2015, pp. 77 – 92.

[6] B. Ma, H. Zhang, Y. Guo, Z. Liu, Y. Zeng: A Summary of Traffic Identification Method Depended on Machine Learning, Proceedings of the International Conference on Sensor Networks and Signal Processing (SNSP), Xi'an, China, October 2018, pp. 469 – 474.

[7] U. Trivedi, M. Patel: A Fully Automated Deep Packet Inspection Verification System with Machine Learning, Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bangalore, India, November 2016, pp. 1 – 6.

[8] S. Rezaei, X. Liu: Deep Learning for Encrypted Traffic Classification: An Overview, IEEE Communications Magazine, Vol. 57, No. 5, May 2019, pp 76 – 81.

[9] M. Shafiq, X. Yu, A. Ali Laghari, L. Yao, N.K. Karn, F. Abdessamia: Network Traffic Classification Techniques and Comparative Analysis Using Machine Learning Algorithms, Proceedings of the 2nd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, October 2016, pp. 2451 – 2455.

[10] U. Huang, P. Li, S. Guo: Traffic Scheduling for Deep Packet Inspection in Software-Defined Networks, Concurrency and Computation: Practice and Experience, Vol. 29, No. 16, August 2017, pp. 1 – 8.

[11] M. Mousa, A.M. Bahaa-Eldin, M. Sobh: Software Defined Networking Concepts and Challenges, Proceedings of the 11th International Conference on Computer Engineering & Systems (ICCES), Cairo, Egypt, December 2016, pp 79 – 90.

[12] L. Polčák, L. Caldarola, A. Choukir, D. Cuda, M. Dondero, D. Ficara, B. Franková, M. Holkovič, R. Muccifora, A. Trifilo: High Level Policies in SDN, Proceedings of the International Conference on E-Business and Telecommunications, Lisbon, Portugal, July 2016, pp. 39 – 57.

[13] J. Arevalo Herrera, J.E. Camargo: A Survey on Machine Learning Applications for Software Defined Network Security, Proceedings of the Applied Cryptography and Network Security Workshop, Bogota, Colombia, June 2019, pp. 70 – 93.

[14] A. Chowdhary, D. Huang, A. Alshamrani, A. Sabur, M. Kang, A. Kim, A. Velazquez: SDFW: SDN-based Stateful Distributed Firewall, November 2018, pp. 1 – 6.

[15] S. Choudhury, A. Bhowal: Comparative Analysis of Machine Learning Algorithms Along with Classifiers for Network Intrusion Detection, Proceedings of the International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Avadi, India, May 2015, pp. 89 – 95.

[16] M. Shafiq, X. Yu, A. Ali Laghari, L. Yao, N.K. Karn, F. Abdesssamia, Salahuddin: WeChat Text and Picture Messages Service Flow Traffic Classification Using Machine Learning Technique, Proceedings of the 18th International Conference on High Performance Computing and Communications / 14th International Conference on Smart City / 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Sydney, Australia, December 2016, pp. 58 – 62.

[17] M.R. Parsaei, M. J. Sobouti, S. Raouf khayami, R. Javidan: Network Traffic Classification Using Machine Learning Techniques over Software Defined Networks, International Journal of Advanced Computer Science and Applications, Vol. 8, No. 7, July 2017, pp. 220 – 225.

[18] M. Karakus, A. Durresi: Quality of Service (QoS) in Software Defined Networking (SDN): A Survey, Journal of Network and Computer Applications, Vol. 80, February 2017, pp. 200 – 2018.

[19] L.- V. Le, D. Sinh, B.- S. P. Lin, L.- P. Tung: Applying Big Data, Machine Learning, and SDN/NFV to 5G Traffic Clustering, Forecasting, and Management, Proceedings of the 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, Canada, June 2018, pp. 168 – 176.

[20] S. Zander, G. Armitage: Practical Machine Learning Based Multimedia Traffic Classification for Distributed QOS Management, Proceedings of the 36th IEEE Conference on Local Computer Networks, Bonn, Germany, October 2011, pp. 399 – 406.

[21] J. H. Shu, J. Jiang, J. X. Sun: Network Traffic Classification Based on Deep Learning, Proceedings of the 1st International Conference on Advanced Algorithms and Control Engineering, Pingtung, Taiwan, August 2018, pp. 1 – 5.

[22] W. Ma, C. Medina, D. Pan: Traffic-Aware Placement of NFV Middleboxes, Proceedings of the IEEE Global Communications Conference (GLOBECOM), San Diego, USA, December 2015, pp. 1 – 6.

[23] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, P. Tofanelli: Revealing Skype Traffic: When Randomness Plays with You, Computer Communication Review, Vol. 37, No. 4, October 2007, pp 37 – 48.

[24] Oracle VirtualBox. 2019 Cited 2019-09-10. Available at: https://www.virtualbox.org

[25] M.V. Bernal, I. Cerrato, F. Risso, D. Verbeiren: Transparent Optimization of Inter-Virtual Network Function Communication in Open vSwitch, Proceedings of the 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, Italy, October 2016, pp. 76 – 82.

[26] Linux Foundatrion, Open vSwitch Project, 2016 [Online]. Available at: http://www.openvswitch.org

[27] Wireshark, 2006 Cited 2019-09-10. Available at: https://www.wireshark.org/

[28] M. Team, 2017 Mininet: An instant virtual network on your laptop (or other pc) - mininet. Cited 2019-09-12. Available at: http://mininet.org

[29] Ryu Framework, 2019. Cited 2019-09-10. Available at: http://osrg.github.io/ryu/

[30] A. Botta, A. Dainotti, A. Pescapè: A Tool for the Generation of Realistic Network Workload for Emerging Networking Scenarios, Computer Networks, Vol. 56, No. 15, October 2012, pp. 3531 – 3547.

[31] Argus Quosient, 2015. Cited 2019-09-10. Available at: https://qosient.com/argus/

[32] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten: The WEKA Data Mining Software: An Update, Explorations Newsletter, Vol. 11, No. 1, June 2009, pp. 10 – 18.