Paper

Network Traffic Classification in an NFV Environment using Supervised ML Algorithms

Gjorgji Ilievski[1] and Pero Latkoski[2]

[1] *Makedonski Telekom AD Skopje, Skopje, RN Macedonia*
[2] *Ss. Cyril & Methodius University, Skopje, RN Macedonia*

https://doi.org/10.26636/jtit.2021.153421

Abstract—We have conducted research on the performance of six supervised machine learning (ML) algorithms used for network traffic classification in a virtual environment driven by network function virtualization (NFV). The performance-related analysis focused on the precision of the classification process, but also in time-intensity (speed) of the supervised ML algorithms. We devised specific traffic taxonomy using commonly used categories, with particular emphasis placed on VoIP and encrypted VoIP protocols serve as a basis of the 5G architecture. NFV is considered to be one of the foundations of 5G development, as the traditional networking components are fully virtualized, in many cases relaying on mixed cloud solutions, both of the premise- and public cloud-based variety. Virtual machines are being replaced by containers and application functions while most of the network traffic is flowing in the east-west direction within the cloud. The analysis performed has shown that in such an environment, the Decision Tree algorithm is best suited, among the six algorithms considered, for performing classification-related tasks, and offers the required speed that will introduce minimal delays in network flows, which is crucial in 5G networks, where packet delay requirements are of great significance. It has proven to be reliable and offered excellent overall performance across multiple network packet classes within a virtualized NFV network architecture. While performing the classification procedure, we were working only with the statistical network flow features, leaving out packet payload, source, destination- and port-related information, thus making the analysis valid not only from the technical, but also from the regulatory point of view.

Keywords—classification, machine learning, network functions virtualization, network traffic.

1. Introduction

Classification of network traffic is always important, as network architectures are changing continuously, especially now, when virtual machines (VM), software defined networking (SDN), private, public, and mixed clouds are commonplace solutions used in the IT world. The current trend favors microservices, containers, application functions and, network functions in network functions virtualization (NFV) environments [1], meaning that network flows are becoming ever more complex. Currently, the majority of 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 SDN components in the NFV environment, thus obstructing the capture or any other operations over the same traffic. This is important both for cloud operators and for entities using the services provided via public clouds. Operations which are common practice and are considered trivial, such as quality of service (QoS), network security, optimization, application management and monitoring functionalities, are becoming a challenge.

In this paper, we are performing an experimental test to reveal network traffic classification efficiency of several supervised machine learning (ML) algorithms. We have created a unique test environment that resembles real life processes and simulates the east-west traffic on the virtual plane, exchanged between virtual hosts, with NFV established. Efficiency of ML algorithms is explored from the point of view of classification precision, but also from the point of view of computational speed. This is very important when we take into consideration the penetration of 5G, as it is tightly integrated with the cloudification of networking operations. For example, the 5G specification calls for a user plane latency of as little as 1 ms for ultra-reliable low-latency communications (URLLC) [2]. This is why the speed of the ML algorithm is crucial and why the process 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 emerging architectures with NFV and 5G implemented therein. It involves 6 different supervised ML algorithms: Bayes Net, NaiveBayes, J48, K-Nearest Neighbors (KNN), Decision Tree and AdaBoost, as they are the ones that are widely used in traditional computer networks, are proven to be reliable while simultaneously providing valid classification results, and are easy to implement in practice. We have used Weka [3] as a tool for classification.

The taxonomy used in this paper relies on 6 classes which are chosen based on our experience in traditional networks and remain in alignment with the network traffic expected within 5G radio, as well as 5G core networks: VoIP, encrypted VoIP, DNS, Management, SSH, HTTP and HTTPS traffic. It is our intention to highlight VoIP and encrypted VoIP classifications which are crucial for ensuring QoS capabilities of 5G networks, thus enabling smart connectivity and providing the ability to steer, secure and break out network traffic.

The NFV architecture is becoming a true 5G enabler, providing the ability to place initial workloads within the network and allowing them grow towards the edge, thus offering the basis needed for the expansion of IoT expected with the growth in 5G penetration.

Numerous previous papers have been devoted to the issue of ML algorithms used for performing packet inspection [4]–[7]. The novel experimental testbed and the method classifying network data based on the statistical parameters of packets and on packet flows only, without relying on source and destination addresses (both MAC and IP addresses), without any examination of the payload and without analysis of the communication ports, are the features that distinguish the approach we have adopted.

The volume of encrypted network traffic is growing fast. Significant numbers of services and applications are using encryption as a primary method of securing information. But this has made traffic classification a challenge. The solution that we propose is applicable in practice without compromising data privacy and integrity. It provides an insight into the performance of supervised ML algorithms and determines which one is best suited for NFV-based environments.

There are also many examples of ML algorithms used for deep packet inspection (DPI) in traditional networks [3], [8], [9]. Unlike the aforementioned works, we focus on virtualization and the NFV environment. In such a scenario, network packets are mostly moving in the east-west direction and are often encrypted, meaning that no classic DPI may be conducted. In the proposed approach, it is not important whether the payload is encrypted or not. Legal requirements related to performing DPI in a cloud environment (especially a public cloud) are satisfied as well, since the data carried within the payload is not compromised. We are using the statistical features of the network packets and the network flows only to create datasets that are later used for training and testing the ML algorithms.

During the testing phase, we are evaluating the efficiency of the algorithm from the point of view of its precision, but also from the point of view of its speed. Network traffic is sniffed directly inside an open vSwitch. We are not introducing any additional probes or SDN components to capture the traffic. We take into consideration all network traffic between the specific virtual elements making up the environment, but also traffic that is used in managing that environment (including that originating from controllers). Incoming and outgoing Internet traffic is dealt with as well. Such a scenario is realistic with majority of cloud solutions. In addition to its precision, the speed of an ML algorithm is even more important in many instances. If the time consumed to classify the data is adding significant latency to network traffic, and if it is consuming the resources (CPU time, memory usage) of the cloud, precision of the classification process is not as relevant.

In the remainder of the paper, we will go through the related work on the subject, briefly explained in Section 2. The experimental setup and the dataset creation procedure are explained in Section 3, while the results are analyzed in Section 4. Section 5 is devoted to the conclusion and our plans for future work.

2. Related Work

Many researches focus on DPI-related aspects and scenarios involving SDN components [10]–[12]. Others research security-related aspects of performing DPI [13], [14] by using SDN probes for sniffing network traffic and for processing data. This work may be distinguished by its NFV-based setup and targets to ensure complete isolation of the packet payload. Some authors consider the classification of network traffic in traditional networks [15], [16] without tackling the specifics of virtualization which is a very trendy solution and forms an important aspect of our work. Parsaei *et al.* [17] are using SDN to categorize traffic by application, using different variants of the neural network estimator. They are using data mining techniques based on different ML algorithms and propose a controller that could dynamically allocate bandwidth to network flows thus optimizing resource allocation. They achieve a classification accuracy rate of over 97%. Unlike in the work described herein, they use source and destination IPs, as well as the transport layer port for classification purposes. In [18], QoS in an SDN based network is researched with an emphasis placed on overcoming the limitations of traditional networking architectures. Different flow routing mechanisms are categorized there. In this research, we explore classification as a basic concept from which QoS may benefit significantly.

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

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 typical of different types of cellular networks (GSM, 3G, 4G). The framework was intended for developing future 5G self-organizing network (SON) applications. Several traffic forecasting-based applications are introduced as well. Five ML algorithms are used to classify traffic generated by mobile applications, with QoS implemented to enable bandwidth guarantees. The conclusion is that Decision Tree offers the best overall performance of all the algorithms tested. Our experiment is limited to the transport network layer, with the aim to classify traffic that is mostly exchanged along the east-west route, using ML algorithms, but also to evaluate the time needed to conclude the classification process, as it is crucial for the future 5G environments.

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

In [20], a machine learning-based classification of multiservice Internet traffic is used to evaluate the use of resources (CPU time and system memory). We are complementing this research, as we are evaluating the time needed by the ML algorithms to perform the classification.

Article [21] proposes a network traffic classification method based on a deep learning network structure. The experimental dataset is created from ten types of data, each of which 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 efficiency of classification performed by the deep learning network. Compared to this work, we are targeting different supervised ML algorithms, 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's speed may be a source of a functional problem in the environment.

The effect of attaching NFV elements to network traffic, especially in terms of an increase or decrease in the volume of traffic processed, is researched in [22]. The authors develop an algorithm that determines the flow path and then proposed a least-first-greatest-last routing.

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

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 the 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 allowing to classify various types of traffic. Viber and Skype are used to generate VoIP traffic, whereas scripts are used to open SSH management sessions for different hosts. Furthermore, a novel testbed is proposed in the context of 5G and to accommodate the usage of NFV elements within the virtualized environment, as expected in the real-life setup. 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 are used to perform further steps of the analysis.

3. Experimental Setup and Dataset Creation

To simulate the east-west traffic within a virtualized NFV-based network, the proposed experimental environment is based on Oracle VirtualBox [24] which is installed on a single physical host with an Ubuntu 18.04 Server. All components are connected with an Open vSwitch (OVS) [25], [26] that ensures network connectivity. The switch is connected to the Internet through the host in a bridge mode. All network packets flow through the OVS switch – this includes east-west traffic packets and north-south traffic packets, both sent to and originating from thw Internet. Traffic is captured directly on the OVS using Wireshark and tshark [27].

Mininet [28] is used as a network simulator. Two different installations on two separate virtual machines are used, each with a different network topology having 100 hosts, 20 switches and links between them and to the OVS. The hosts within the simulated networks have private IP addresses and are capable of communicating with each other. GRE tunneling is used to link the two simulated Mininet networks. Some of the hosts within Mininet have NAT-ed IP addresses and are able to communicate with the Internet. The Ryu Controller [29] is used to control the simulated Mininet networks. It is installed and configured on a separate virtual machine.

There are four other virtual machines connected to the OVS which are also used for traffic generation. Skype and Viber are installed thereon to simulate VoIP traffic. When initiated, VoIP needs access to the Internet, but later on peer-to-peer communications may be observed within the OVS, in a fully east-west direction. The script that initiates ssh sessions is enabled on the VMs. We have developed a Python script that automatically starts SSH sessions with the Mininet hosts as well. The SSH sessions were started in time intervals that are following Poisson distribution.

A distributed Internet traffic generator (D-ITG) [30] generates various types of TCP and UDP traffic among the hosts within the Mininet. Different scripts are used to generate traffic at packet level, replicating specific stochastic processes for both inter departure time (IDT) and packet size (PS) random variables.

Figure 1 shows an overview of the experimental setup, showing its components symbolically.

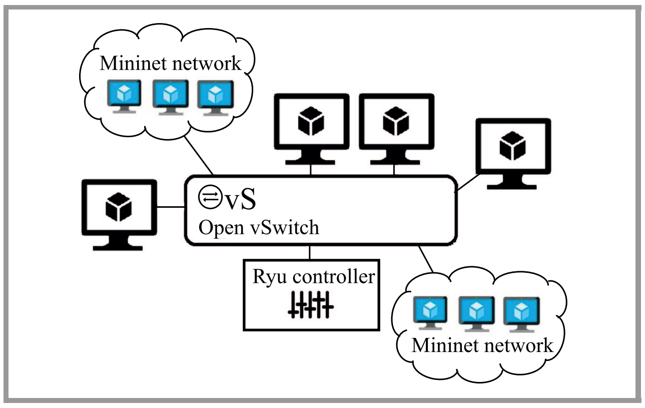

Fig. 1. Experimental environment.

We have performed 50 different experiments to generate various types of traffic (using D-ITG, Skype, Viber, custom scripts) and to analyze it. The experiments were conducted in time intervals varying from 4 to 20 minutes, with VoIP calls lasting from 10 s to 10 minutes, following Poisson distribution. One dataset per experiment was generated. Different D-ITG scripts for different traffic simulations were used in each of the experiments. The scripts used different Mininet hosts and different paths in each attempt. The average number of packets captured was 1.262.375 and the average number of flows was 4090. We have devised a specific classification of traffic, relying on commonly used classes, based on experience from the traditional networks. As it will be shown in the results, precision of the classification process was calculated as an overall figure, but also independently for each of the classes, in order to calculate the macro-average precision level in which the contribution of each class is treated equally (as the number of packets and flows varies for every class).

We used the following labels for the individual classes: DNS – for all traffic used for name resolution, NETMGMT – all traffic used for host 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. Based on the Wireshark pcap files generated, UDP and TCP packet flows, as well as the classes used for ML training and then for determining and confirming the level of precision, are identified using Argus [31]. Similarly to [5], we define a flow as a bidirectional 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 only. When observing flows within the OVS, one could notice that most of the traffic is of the east-west variety, is taking place inside the virtual layout and between the hosts, but flows from the management generated by the hypervisor and the Ryu controller could be detected as well. Because our focus was on the NFV-based environment, some of the flow features, such as the source and destination IP, MAC address, as well as the communication port that can vary inside the virtual environment, were not taken into consideration.

To train and to test the supervised ML algorithms, we have used Weka [3], [32]. 2/3 of each dataset were used for training, while 1/3 was used for testing each of the algorithms. As not all the attributes contribute to the classification equally, 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 ranked the attributes that are used for the algorithms, with the information gain of every attribute being evaluated thereafter. This approach prevents potential data leakage.
Based on experience from traditional networks and thanks to a careful observation of the datasets obtained, we have selected the attributes given in Table 1 as features that characterize the flows. The payload is not used due to the privacy of cloud environments and due to the use of different encryption methods that will make the payload irrelevant for classification purposes. The labels in the transport layer header (e.g. the port numbers) are not used as well, as they may be changed easily. A short explanation of each of the selected attributes is provided inside the table. The following section presents the results of the test involving the supervised ML algorithms and contains their analysis.

Table 1
Flow attributes

Abbreviation	**Feature**
proto	Transaction protocol
rate	Packets per second
srate	Source packets per second
drate	Destination packets per second
sintpkt	Source interpacket arrival time
dintpkt	Destination interpacket arrival time
sjit	Source jitter
djit	Destination jitter
mdoffset	Mean of the data offset Values of the packets in the flow
smeansz	Mean of the flow Packet size transmitted by the source
dmeansz	Mean of the flow packet Size transmitted by the destination
smaxsz	Max packet size for source
dmaxsz	Max packet size for destination
sminsz	Min packet size for source
dminsz	Min packet size for destination

4. Results and Analysis

We have conducted 50 experiments, creating 50 datasets. All the ML algorithms were tested on each dataset. The performance of each algorithm was defined as a combination of its precision and the time needed to perform the classification. Since time consumption is correlated to the

performance of the machine on which the analysis is conducted, all classification tasks were performed on the same machine, with all processes active thereon that may influence performance observed carefully. A mean value of 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 the number of instances that are correctly identified as belonging to a specific class,
- FP is the number of instances that are not correctly identified as belonging to a specific class,
- TN is the number of instances that are correctly identified as not belonging to a specific class,
- FN is number of instances that are not correctly identified as not belonging to a specific class.

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

$$Precision = \frac{TP}{TP+FP} \ . \tag{1}$$

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.

Table 2
Algorithm precision

No.	ML algorithm	Precision
1	AdaBoost	0.7440±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**

It can be seen that the Decision Tree algorithm has the best overall precision. It is followed by J48 and BayesNet. On the other hand, the AdaBoost algorithm has the worst overall performance with the lowest precision of 74.4%.
In order to perform a deeper analysis of the precision level, micro average precision was calculated – an indicator that aggregates the contribution of all classes and calculates the average metric, as given by Eq. 2. The results are presented in Table 3.

$$Precision\_MIC = \frac{TP_1+TP_2+...+TP_N}{TP_1+FP_1+TP_2+FP_2+...+TP_N+FP_N} \ . \tag{2}$$

Not all classes have same or similar number of packets and flows, and the data distribution is skewed. As the class distribution is unequal, the datasets are imbalanced. To avoid the problem of data balancing and to come to valid conclusions, we are calculating macro average precision, recall, the F1-score.

Table 3
Micro average precision of algorithms

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**

Macro average precision is the average of measure of each class. This means that every class will weigh the same in the macro average precision. Equation 3 is used to calculate macro average precision (Precision_MAC), where Pr1, Pr2, etc. denote the precision of the algorithm in relation to the individual classes.

$$Precision\_MAC = \frac{Pr_1+Pr_2+...+Pr_N}{Count(Pr)} \ . \tag{3}$$

The results are shown in Table 4, where the statistical standard deviation is calculated for the precision between classes.

Table 4
Macro average precision of algorithms

No.	ML algorithm	Macro average precision
1	AdaBoost	0.20335±0.3064
2	BayesNet	0.88990±0.1489
3	J48	0.98240±0.0148
4	KNN	0.82735±0.2202
5	NaiveBayes	0.78915±0.2048
6	**Decision Tree**	**0.98480±0.0107**

It becomes clear that the algorithms are not performing in the same manner with regard to all the classes. The Decision Tree algorithm has the highest macro average precision rate and the lowest standard deviation between classes, meaning that it classifies all classes similarly. J48 is very close to Decision Tree, with the precision rate of over 98%. On the other end of the scale, the AdaBoost algorithm shows a very low macro average precision rate with a high standard deviation, meaning that it performs poorly with regard to different classes. The K-Nearest Neighbor algorithm is underperforming as well, with its macro average precision rate equaling 82% only. After comparing these results with the standard weighted precision shown in Table 2, one may see that the algorithms have the same order, but the macro precision rate of the lower-end algorithms is worse, leading to the conclusion that AdaBoost and KNN offer different precision levels for different classes.
In order to evaluate the impact of the false negative classified instances, Recall is used as a model metric. It is the proportion between true positive instances and total actual instances:

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

Recall was used to calculate the F1-score of the ML algorithms tested in our experiments. It is a metric that balances the precision level 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:

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \,. \qquad (5)$$

Table 5 shows the F1-score values calculated for our experiments. The Decision Tree ML algorithm has the best F1-score, followed by J48, BayesNet, KNN, NaiveBayes and AdaBoost. The last algorithm has the F1-score of 23.2% only, with very high standard deviation.

Table 5
F1-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

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

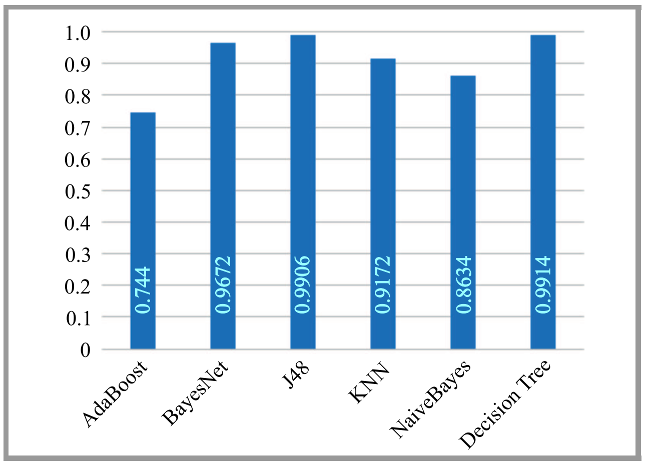

Fig. 2. Algorithm precision.

Precision of the algorithm is only one of the characteristics that determines its actual usability. The time needed to perform the classification is an important aspect as well. If the time needed to complete the classification is too long, the process will add latency to network communications, thus making the benefit of the classification too costly. This is important especially in protocols in which latency may degrade the quality of service, such as VoIP. Furthermore, this is also crucial in 5G scenarios, where latency is one of the major concerns. Consumption of the system's resources (CPU, memory, etc.) is another problem, as it increases if the algorithm operates as a slower pace. The two metrics (precision and time consumption) combined determine the overall performance of the algorithms.

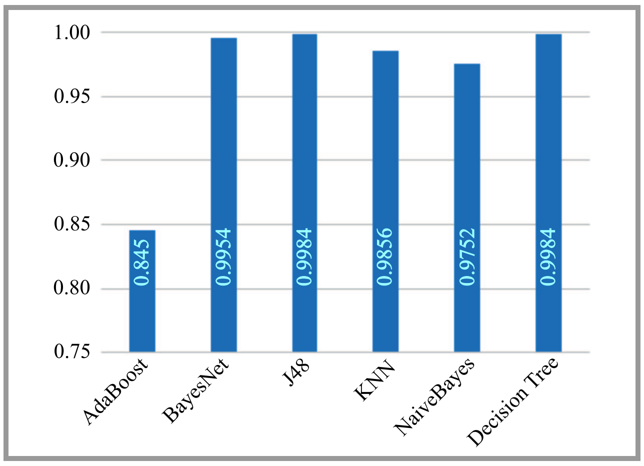

Fig. 3. Micro average precision.

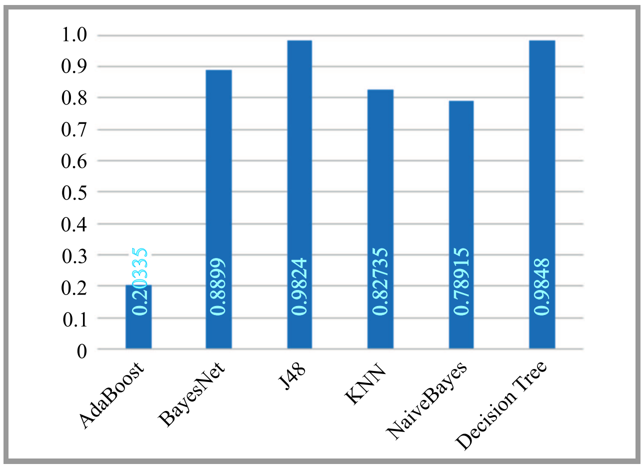

Fig. 4. Macro average precision.

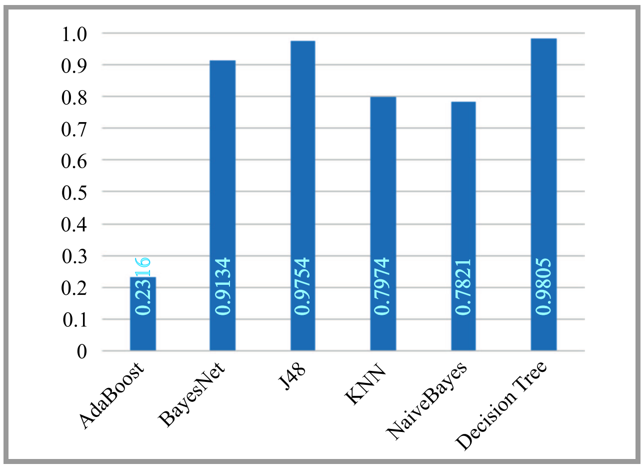

Fig. 5. F1-score.

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

Table 6 shows the average time needed by the algorithms to perform the classification procedure within the 6 chosen classes.

Table 6
Average time needed for classification

No.	ML algorithm	Average time [s]
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 concerning the average time required to perform the classification show that the AdaBoost algorithm is the fastest. Decision Tree and BayesNet algorithms are ranked second and third ex-equo, being 25% slower than AdaBoost. The result of J48 is satisfactory as well. NaiveBayes is almost 9 times slower than AdaBoost and more than 6 times slower than Decision Tree. The KNN algorithm is the slowest. Decision Tree and AdaBoost require only 5.9% of the time needed by KNN to perform the classification.
Figure 6 graphically represents the average time required by the algorithms to perform the classification.

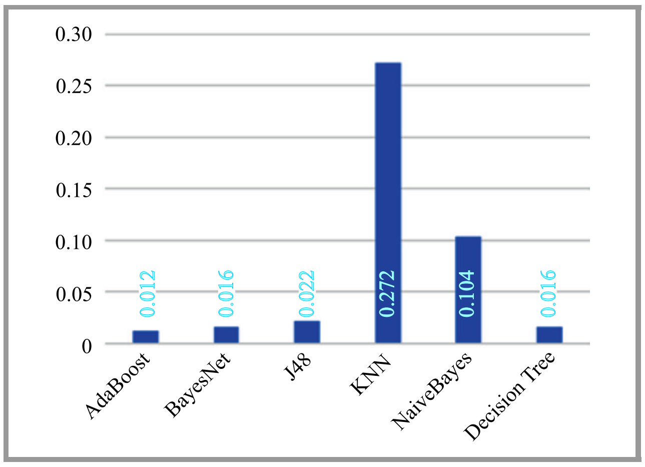

Fig. 6. Time required to perform the classification [s].

To summarize, when we take a look at both the precision and the time needed for classification, the Decision Tree supervised ML algorithm offers the best overall performance. Although AdaBoost is the fastest algorithm, its classification precision is poor and unsteady across different classes, which makes this algorithm unreliable for the scenario in question. J48 also offers a high level of precision that is evenly distributed among the classes, but it is slower than Decision Tree and BayesNet. Nevertheless, its speed similar to that of Decision Tree and BayesNet algorithms, which makes it a valid choice as well. BayesNet offers a high degree of precision, but macro average precision and F1-score values show that the distribution of its precision among the different classes is not as good as in the case of Decision Tree and J48.
NaiveBayes is in the middle of the scale, both in terms of precision and time. KNN, in turn, offers macro average precision of approximately 83% and F1-score of 80%, but it is by far the slowest algorithm, meaning that it is only useful in situations in which the time needed to perform the classification is of little importance.

5. Conclusion and Future Work

The main idea behind this paper was to present a method for creating datasets based only on the statistical characteristics of network traffic flows, and to test the performance of machine learning algorithms based on the created datasets. All those tasks were performed with the use of an experimental testbed with NFV architecture.
The efficiency of algorithms is examined taking into consideration their precision and the time required to perform the classification. Such an approach is important from the point of view of virtualization point of view, where mixed cloud scenarios are commonplace, but also from the point of view of the growing popularity of 5G, where network latency is crucial.
Our experimental testbed was used to perform multiple experiments and to collect network traffic data from which IP flows were extracted. The statistical features of the flows were used as attributes for the classification procedure. Because such attributes as source and destination IP, MAC addresses and communication ports may vary within 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 purposes.
The environment used did not rely on any network probes or SDN elements to collect the data, allowing not to affect the east-west traffic is any manner whatsoever. The traffic was fully intercepted within the virtual layer, where it resides naturally. Such an approach has an impact on resource consumption as well, minimizing additional latency that may be added to network packets by redirecting or by port replication used in the traditional DPI.
The results have shown that the Decision Tree algorithm offers the best overall performance, both from the point of view of classification precision and time consumption. 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 of approximately 80%, but they are slow. This applies, in particular, to KNN which is almost 20 times slower than Decision Tree and BayesNet. AdaBoost shows the worst performance with its precision varying considerably among the different classes. The same applies also to its macro average precision and F1-score.

The analysis presented in this paper may be relied upon in practice within multiple systems that are built on top of cloud environments. NFV elements are now an unavoidable part of such infrastructures. The 5G infrastructure relies on these types of systems, and connectivity with such systems is most likely to rely on 5G access technologies. In those examples, QoS, network and application security, data management, system and process monitoring and control all depend on a valid network traffic classification scheme that needs to be precise and fast, without consuming excessive amounts 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 intensity of the supervised ML algorithms, by introducing large numbers of classes and by reducing the classes. Another idea is to expand the experimental testbed to include multiple hosts and distributed switches, and to evaluate a network that is moving across multiple hosts.

References

[1] M. Chiosi *et al.*, "Network Functions Virtualisation", *Introductory White Paper*, 2015 [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf (accessed on 10.10.2020).

[2] M. Eiman, "Minimum Technical Performance Requirements for IMT-2020 radio interface(s). Presentation", 2018 [Online]. Available: https://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2020/Documents/S01-1_Requirements%20for%20IMT-2020_Rev.pdf (accessed on 10.10.2020).

[3] E. Frank, M. A. Hall, and I. H. Witten, *Data Mining: Practical Machine Learning Tools and Techniques, Fourth Edition.* San Francisco, CA, USA: Morgan Kaufmann, 2016, pp. 2464–2468 (ISBN: 9780128042915).

[4] J. Vergara-Reyes, M. C. Martinez-Ordonez, A. Ordonezy, and O. M. C. Rendon, "IP traffic classification in NFV: a benchmarking of supervised machine learning algorithms", in *IEEE Colombian Conf. on Commun. and Comput.*, Cartagena. Colombia, 2017 (DOI: 10.1109/ColComCon.2017.8088199).

[5] R. Alshammari and A. Nur Zincir-Heywood, "Identification of VoIP encrypted traffic using a machine learning approach", *J. of King Saud University – Computer and Informat. Sci. archive*, vol. 27, no. 1, pp. 77–92, 2015 (DOI: 10.1016/j.jksuci.2014.03.013).

[6] B. Ma, H. Zhang, Y. Guo, Z. Liu, and Y. Zeng, "A Summary of Traffic Identification Method Depended on Machine Learning", *Sensor Networks and Signal Process. (SNSP) 2018 Int. Conf.*, Xi'an, China, 2018, pp. 469–474 (DOI: 10.1109/SNSP.2018.00094).

[7] U. Trivedi and M. Patel, "A fully automated deep packet inspection verification system with machine learning", *IEEE Int. Conf. on Advanced Networks and Telecommun. Systems*, Bangalore, India, 2016 (DOI: 10.1109/ANTS.2016.7947802).

[8] S. Rezaei and X. Liu, "Deep Learning for Encrypted Traffic Classification: An overview", *IEEE Commun. Mag.*, vol. 57, no. 5, 2019, pp. 76–81 (DOI: 10.1109/MCOM.2019.1800819).

[9] M. Shafiq *et al.*, "Network traffic classification techniques and comparative analysis using machine learning algorithms", in *Proc. 2nd IEEE Int. Conf. on Computer and Commun. (ICCC)*, Chengdu, China, 2016, pp. 2451–2455 (DOI: 10.1109/CompComm.2016.7925139).

[10] U. Huang, P. Li, and S. Gu, "Traffic scheduling for deep packet inspection in software-defined networks", *Concurrency and Comput.: Practice and Experience*, 2017 (DOI: 10.1002/cpe.3967).

[11] M. Mousa, A. Bahaa-Eldin, and M. Sobh, "Software Defined Networking concepts and challenges", *11th Int. Conf. on Computer Engin. & Systems (ICCES)*, Cairo, Egypt, 2016, pp. 79–90 (DOI: 10.1109/ICCES.2016.7821979).

[12] L. Polčák *et al.*, "A High Level Policies in SDN", *Int. Conf. on E-Business and Telecommun.*, Colmar, France, 2016, pp. 39–57 (DOI: 10.1007/978-3-319-30222-5_2).

[13] J. Arevalo Herrera and J. E. Camargo, "A Survey on Machine Learning Applications for Software Defined Network Security", *Applied Cryptography and Network Security Workshops ACNS*, Bogotá, Colombia, vol. 11605, 2019, pp. 70–93 (DOI: 10.1007/978-3-030-29729-9_4).

[14] A. Chowdhary *et al.*, "SDFW: SDN-based Stateful Distributed Firewall", *Project: Secured and Resilient Networking*, 2018 (DOI: 10.13140/RG.2.2.11001.93281).

[15] S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection", *Int. Conf. on Smart Technol. and Management for Comput., Commun., Controls, Energy and Materials (ICSTM)*, Avadi, India, 2015, pp. 89–95 (DOI: 10.1109/ICSTM.2015.7225395).

[16] M. Shafiq *et al.*, "WeChat text and picture messages service flow traffic classification using machine learning technique", *IEEE 18th Int. Conf. on High Performance Comput. and Commun.; IEEE 14th Int. Conf. on Smart City; IEEE 2nd Int. Conf. on Data Sci. and Systems (HPCC/SmartCity/DSS)*, Sydney, NSW, Australia, 2016, pp. 58–62 (DOI: 10.1109/HPCC-SmartCity-DSS.2016.0019).

[17] M. Reza, M. J. Sobouti, S. Raouf, and R. Javidan, "Network traffic classification using machine learning techniques over software defined networks", *Int. J. of Adv. Computer Sci. and App.*, 2017 (DOI: 8.10.14569/IJACSA.2017.080729).

[18] M. Karakus and A. Durresi, "Quality of Service (QoS) in Software Defined Networking (SDN): A survey", *J. of Network and Computer App.*, 2016 (DOI: 80. 10.1016/j.jnca.2016.12.019).

[19] L. Le, D. Sinh, B. P. Lin, and L. Tung, "Applying Big Data, Machine Learning, and SDN/NFV to 5G traffic clustering, forecasting, and management", *4th IEEE Conf. on Network Softwarization and Workshops (NetSoft)*, Montreal, Canada, 2018 (DOI: 10.1109/NETSOFT.2018.8460129).

[20] S. Zander and G. Armitage, "Practical machine learning based multimedia traffic classification for distributed QOS management", *2011 IEEE 36th Conf. on Local Computer Networks*, Bonn, Germany, 2011, pp. 399–406 (DOI: 10.1109/LCN.2011.6115322).

[21] J. H. Shu *et al.*, "Network traffic classification based on deep learning", *First Int. Conf. on Advanced Algorithms and Control Engin.*, Pingtung, Taiwan, 2018 (DOI: 10.1088/1742-6596/1087/6/062021).

[22] W. Ma, C. Medina, and D. Pan, "Traffic-aware placement of NFV middleboxes", *IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, 2015, pp. 1–6 (DOI: 10.1109/GLOCOM.2015.7417851).

[23] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli, "Revealing Skype traffic: when randomness plays with you", *ACM SIGCOMM Computer Commun. Review*, vol. 37, no. 4, pp. 37–48, 2007 (DOI: 10.1145/1282427.1282386).

[24] Oracle VirtualBox [Online]. Available: https://www.virtualbox.org (accessed on 10.09.2020).

[25] M. V. Bernal, I. Cerrato, F. Risso, and D. Verbeiren, "Transparent optimization of inter-virtual network function communication in open vSwitch", *5th IEEE Int. Conf. on Cloud Netw. (Cloudnet)*, Pisa, Italy, 2016, pp. 76–82 (DOI: 10.1109/CloudNet.2016.26).

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

[27] Wireshark [Online]. Available: https://www.wireshark.org/ (accessed on 10.09.2020).

[28] Mininet: An instant virtual network on your laptop (or other PC) [Online]. Available: http://mininet.org (accessed on 12.09.2020).

[29] Ryu Framework [Online]. Available: http://osrg.github.io/ryu/ (accessed on 10.09.2020).

[30] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios", *Computer Networks (Elsevier)*, 2012, vol. 56, no. 15, pp. 3531–3547 (DOI: 10.1016/j.comnet.2012.02.019).

[31] Argus Quosient [Online]. Available: https://qosient.com/argus/ (accessed on 10.09.2020).

Gjorgji Ilievski received his M.Sc. in Computer Engineering in the field of Data Mining Technologies in 2012. His research interests include statistical analysis, cloud computing, computer networking, virtualization, LTE and 5G. He works in telecommunications industry at Makedonski Telekom AD, as a senior cyber security engineer.

https://orcid.org/0000-0003-2109-7027
E-mail: gjorgji.ilievski@telekom.mk
IT Department
Cyber Security Unit
Makedonski Telekom AD Skopje
Kej 13-ti Noemvri 6
Skopje, Macedonia

Pero Latkoski received his M.Sc. and Ph.D. degrees from the Faculty of Electrical Engineering and Information Technologies, Ss Cyril and Methodius University in Skopje, in 2006 and 2010, respectively. Currently, he holds the position of full professor at the same university's Institute of Telecommunications. His research interests include communication protocol engineering, software defined networking, and information theory.

https://orcid.org/0000-0002-8406-4057
E-mail: pero@feit.ukim.edu.mk
Faculty of Electrical Engineering and Information Technologies
Telecommunications Institute
Ss Cyril & Methodius University
Rugjer Boshkovikj 18
Skopje, Macedonia