




Data representation for CNN based internet traffic classification: a comparative study

Ola Salman¹  · Imad H. Elhadj¹ · Ayman Kayssi¹ · Ali Chehab¹

Received: 1 January 2020 / Revised: 6 July 2020 / Accepted: 28 July 2020 /
Published online: 19 August 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

It has been well established that the Internet of Things will bring an expansion in traffic volume and types. This will bring new challenges in terms of Quality of Service (QoS) and security, requiring innovative traffic management techniques. Traffic classification is a main network function that helps in managing both QoS and security. Different machine learning based methods have been applied for this aim. However, traditional machine learning methods rely on hand crafted features, limiting the model ability to learn. Deep Learning (DL), a branch of machine learning, is characterized by its representation learning ability. In this paper, we analyse two methods of data representation for DL-based classification: a raw packet-based representation and a quasi-raw flow-based representation. Different tests are performed to evaluate the robustness of these data representation methods. The tests include features' importance, model robustness, and anonymization tests. The results show that raw data representation suffers from traffic anonymization and the fact that many packet fields are data-dependent. On the other hand, the flow-based representation is sensitive to the number of packets used for classification and to traffic obfuscation.

Keywords Deep learning · Internet of things · Traffic classification · Data representation

1 Introduction

The Internet of Things includes a heterogeneous set of connected devices that run different types of applications. These devices and applications will generate different types of traffic,

✉ Ola Salman
oms15@mail.aub.edu

Imad H. Elhadj
ie05@mail.aub.edu

Ayman Kayssi
ayman@mail.aub.edu

Ali Chehab
chehab@mail.aub.edu

¹ American University of Beirut, Beirut 1107 2020, Lebanon

having different requirements in terms of Quality of Service (QoS) and security. Managing both QoS and security in this high-scale network calls for innovative network management techniques [59]. In this context, traffic classification is considered as an essential element for traffic engineering, security management, traffic trends analysis, and so on [15]. The ability to classify the traffic based on the different requirements in terms of bandwidth, latency, throughput, etc., enables the allocation of the corresponding resources for each type of traffic and thus, guarantee good QoS [4, 5, 7, 24]. On the other hand, traffic classification techniques can be used to detect abnormal traffic [7, 39, 67]. Furthermore, Intrusion Detection Systems (IDSs) are using machine learning to reveal the attack name/type [28].

Internet traffic consists of the flow of data between the different communication parties. The Internet Protocol (IP) network traffic dominates other Internet traffic types. In the IP network case, data is transmitted in the form of packets. An IP packet is a data unit consisting of two parts: the header and the payload. The header contains the main connection parameters and the payload contains the user data. A traffic flow is defined as being the ensemble of packets having the same source IP address, source port number, destination IP address, destination port number, and transport protocol. In fact, the traffic flows exhibit different communication patterns based on the traffic type. Thus, statistical features extracted at the flow level could reveal information about the corresponding traffic type (see Fig. 1). For example, real-time applications usually use small packet sizes and the inter-arrival time between packets is small. On the other hand, for file transfer applications, the packet sizes are large and the inter-arrival time between packets is high.

Accordingly, different traffic classification methods have been proposed in the last decade [50, 54] (see Fig. 2). First, port numbers were used to classify traffic, given that different application protocols were assigned known port numbers by the Internet Assigned Numbers Authority (IANA). However, today’s applications are using dynamic port numbers [3, 45, 63]. Additionally, traffic encapsulation hides the port numbers (e.g. mobile traffic is encapsulated in HyperText Transfer Protocol (HTTPS)). Later, Deep Packet Inspection (DPI) was proposed to classify traffic by examining the content of the packet’s payload.

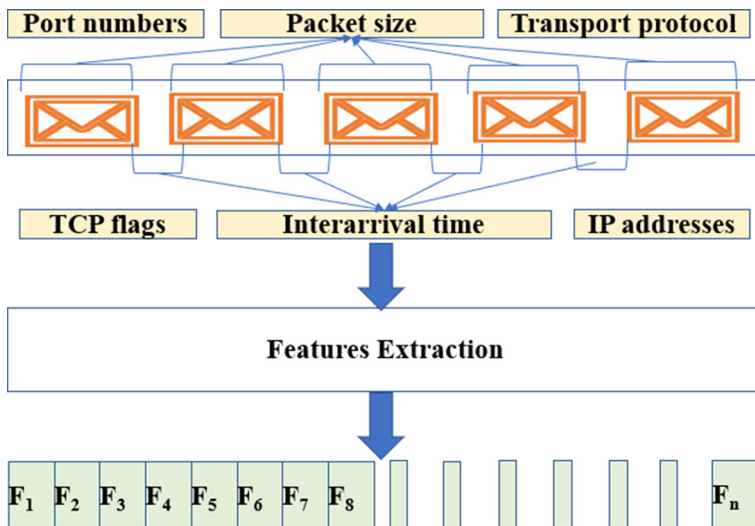


Fig. 1 Flow packets characteristics

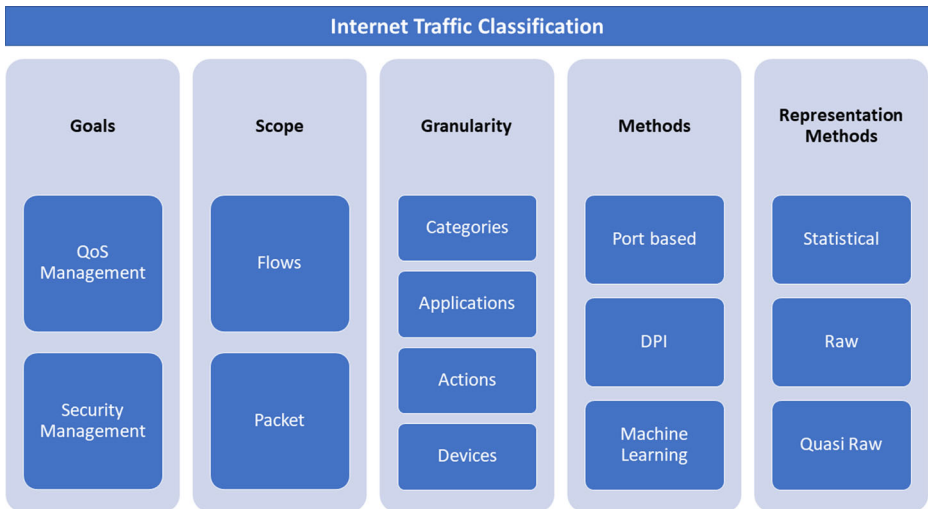


Fig. 2 Traffic classification goals, scope, methods, and data representation methods

However, this technique entails large processing overhead and fails in the case of encrypted traffic [51].

More recently, machine learning based methods were proposed to classify traffic into protocols, applications, categories, user actions, device types, etc. However, the proposed solutions have many limitations including hand-designed features, traffic obfuscation, in addition to model and data biases. Recently, Deep Learning (DL), has emerged as an efficient machine learning method to be applied on raw data, having the representation learning capability [6]. DL is expected to play an essential role in future applications due to its high data abstraction and model robustness [22]. Especially, Convolutional Neural Network (CNN) is a DL architecture that is initially used for image recognition. Being inspired by the visual cortex, CNN achieves very high accuracy when applied on raw images. Inspired by this, network traffic was represented in image format for CNN-based classification. Recent works have applied CNN for traffic classification, by transforming the first packet of a network flow to an $(n \times n)$ gray image [26, 33, 37, 64–66, 68]. Another work considered a Red Green Blue Alpha (RGBA) image-based representation of a network flow by extracting four features (size, inter-arrival time, protocol, direction) from the first $(n \times n)$ packets [52]. However, many questions arise when applying DL for traffic classification: how to represent the traffic to be passed to DL architectures? what are the consequences of passing raw traffic data on the classification robustness? and how DL compares to traditional machine learning methods with hand-crafted features?

The ML-based traffic classification is already an established domain. However, recently, the adversarial attacks using obfuscation techniques, are rising as tools to avoid classification or detection. In this paper, we shed the light on the robustness and security of using ML for classification and we study the robustness of the used features and how DL can counter this. This work is highlighting a very important challenge in traffic classification, which is the robustness of the selected features given the possibility of intended or unintended modification of traffic characteristics.

In this paper, we analyze and compare two data representations for CNN-based traffic classification: the packet representation (gray images), and the flow representation (RGBA

images). For this aim, we consider a hierarchical classification framework of the traffic into different categories, applications, user actions, and device types. The performed tests aim at evaluating, for each method, the model robustness, the features importance, and the features immunity towards anonymization. The results show that the performance of the packet data representation method is dependent on some dynamic connection parameters (i.e. Internet Protocol (IP), and Media Access Control (MAC) addresses). On the other hand, the flow data representation performance might be affected by traffic obfuscation techniques.

DL is emerging as a leading machine learning technique. Recently, the application of DL in the network domain witnessed an increased interest. Specifically, when representing traffic as images for traffic classification, CNN is considered the best suited DL architecture. In a previous work [52], considering the packet-based and flow-based representation methods, we showed that CNN gives the best results compared to other DL architectures such as Recurrent Neural Network (RNN), and Deep Neural Network (DNN). In addition, we showed that the ConvNet architecture outperforms other CNN architectures like GoogleNet, ResNet, AlexNet, etc.

To the best of our knowledge, this is the first work that investigates the comparison between two image-based traffic representations for CNN-based classification. In fact, the application of DL, especially CNN, in the traffic classification domain is in its early stages. Consequently, different solutions were proposed focusing on the classification accuracy, without considering the robustness of the employed features (data representations). By proposing new features importance tests, the comparative study in this paper will pave the way for future research in the traffic classification domain, by selecting features that are immune towards obfuscation. It should be noted that this paper's aim is not to find the optimal classifier with best classification accuracy. This paper is meant to be a comparative study and relative analysis is required. Knowing that the CNN accuracy is affected by the used parameters in the implementation, we adopted standard architecture, activation function, output function, and optimization method with a commonly-used learning rate for such applications. As such, the contributions of this paper can be summarized as follows:

- A comprehensive comparison between two data representation methods is conducted for CNN-based traffic classification. In addition, these methods were compared to a state-of-the-art method applying Random Forest (RF) with a subset of the Moore features [46].
- New tests were designed to evaluate the features robustness by anonymizing fields in case of the raw based classification or by anonymizing flow characteristics through padding and shaping for statistical features based classification. In addition, model robustness was evaluated by means of a proposed test based on hierarchical classification.
- The anonymization effect on both methods is studied.
- A new feature importance analysis method is proposed for both representations.
- For the first time, the effect of traffic obfuscation, including padding and shaping, is studied considering different representation and classification methods.

The rest of this paper is organized as follows: in Section 2, we review the traffic representation methods. In Section 3, we detail the data collection, the representation methods, the classification objectives, and the classification method. Section 4 presents the comparison criteria and methodology, along with the corresponding tests. Section 5 presents the experimental results. In Section 6, we discuss and analyse the obtained results. Finally, we conclude in Section 7.

2 Related work

Data representation is an essential part of any machine learning model. In this section, we review the data representation methods used in the traffic classification domain.

2.1 Behavioral based

The behavioral-based data representation aims at capturing the patterns of the interaction between different network elements (i.e. client, server, etc.) [8, 11, 16, 18, 31, 40, 42, 49]. In [29, 30], Karagiannis et al. proposed BLINC, a system that represents flows as “graphlets”. A graphlet is a communication representation using four connection parameters: source IP address, destination IP address, source port number, and destination port number. In [25], applications are classified based on a set of rules reflecting their behavioral profiles. In [8], a combination of both host-level and flow-level features was used to classify traffic, where the host-level features represent the host connections parameters. In [71], the “user-app bipartite” representation was proposed for user profiling. This method consists of extracting the interaction between the user and the server per application. In [44], graphlets and packet size-based features were employed to classify the Internet traffic. However, the behavioral data representation depends on the network, user, and time. Moreover, it exhibits high complexity and storage overhead, given that data has to be stored and processed for each user per network connection.

2.2 Vectors

One of the traditional traffic representation methods represents the network flows as vectors of statistical features. A comprehensive list of flow statistical features was proposed by Moore et al. in [46]. This list consists of 249 statistical features derived per flow based on the packets size, the packets inter-arrival time, the source IP address, the destination IP address, the source port number, the destination port number, the TCP flags, and the transport protocol. In [23], another vector-based representation employing hot encoding was proposed for the first n-bytes of a flow. However, in either case, the model performance is prone to degradation when applied on data collected from a different network environment.

2.3 Time-series

Time-series is a way to represent the network flow as a sequence of values. In [12], the network flows were represented by time-series for the bytes within incoming/outgoing packets, and the packet inter-arrival time. In [1], the authors represented the traffic flows by time-series of the data rate. However, this type of representation needs high number of packets to be able to extract distinctive time-series representation for different traffic types.

2.4 Word embedding

Given that certain protocols requests (e.g. HTTP request) contain useful information (e.g. host name), word embedding was applied to represent the traffic accordingly. In this context, FLOWR was proposed in [69, 70], by extracting the application signature, which consists of key-values extracted from the HTTP queries. In [27], the Longest Common Subsequence (LCS) algorithm was applied to extract common strings from the packet header. In [47], Word2Vec was applied to generate word-embedding for Domain Name Server

(DNS) requests, to classify the associated traffic flows. Net2Vec was proposed in [20] to classify the HTTP(S) traffic by considering the (IP address, host name) tuples from the network flows. In [21], multi-level signatures were extracted based on common substrings at the flow level, the content level, and the packet level. Moreover, Bag of Word (BoW)-based traffic representation was proposed in [48]. Numeric vectors were created to represent the web pages Uniform Resource Locators (URLs). In [73], ProWord was presented as being a framework for word-based protocol representation. ProWord consists of applying segmentation of the protocol request content to extract the keywords identifying the considered traffic. Moreover, in [38], Time To Live (TTL) and specific strings in the HTTP requests were used for mobile phones fingerprinting. However, many obfuscation techniques, including encryption and tunneling, affect the accuracy of such techniques.

2.5 2D matrices or images

Recently, DL has been applied in the communications and networking domains for traffic classification [17, 34, 74]. Essentially, DL was used for feature generation [56]. On the other hand, new data representation methods have been proposed to apply DL for traffic classification. In [32], Leroux et al. considered two spaces: packet size-inter-arrival time and burst time-burst size to represent the traffic flows in 2D histograms, considering the first 1024 packets of an application session. In [9], hot encoding was used to represent the HTTP packet header as $(m \times l)$ vectors, where $m = 95$ and $l = 1114$. In [10], kernel Hilbert space embedding was applied to represent the network by 6-channel images of statistical features along their marginal and conditional distributions.

DL consists of different types of architectures. CNN is one such type that is specialized for image applications. Consequently, to apply CNN for traffic classification, the representation of traffic as images was considered in the literature. In [26, 33, 37, 64–66, 68], a gray image representation was adopted by considering the first $(n \times n)$ bytes of traffic flows, as illustrated in (see Fig. 3a). In [52, 53], the authors proposed to extract 4 features (packet size, inter-arrival time, transport protocol, and direction) for the first $(n \times n)$ packets of a traffic flow, as illustrated in (see Fig. 3b) [36].

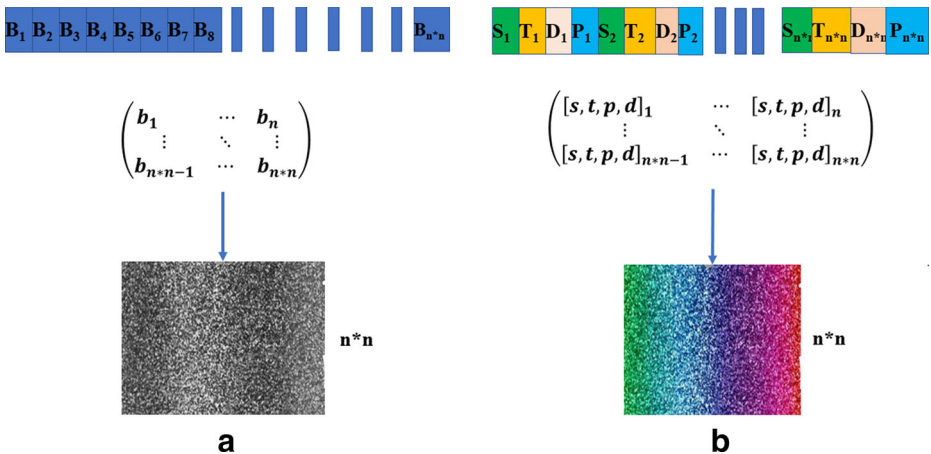


Fig. 3 Data representation: a gray images and b RGBA images

The comparison between the different data representations methods has been studied in [2, 35]. In [2], the comparison was performed between DL and Random Forest (RF) for mobile traffic classification. Different data representation methods were considered for DL classification including: the first N bytes of the payload, the first N bytes of a bi-flow, and a set of features extracted from the first 20 packets. For RF, a statistical set of 40 features including port number information was considered. The results show that DL outperforms RF when the applications are not overlapping, which means that the applications generate different types of traffic (voice call, mail, file transfer, etc.). In the opposite case, RF-based classification presented better results. Another comparison was performed in [35], between different data representation methods: 1) basic features [13, 55], 2) Moore features [46], 3) graph features [43], 4) joint features [14], and 5) service features [19]. The results showed that the combination of basic with in-flow behavior, distribution, and packet header features, outperforms the other representations. In fact, a key question could be asked about representing the traffic by statistical features or raw data when DL is considered for classification. None of the previous comparisons consider the comparison of raw traffic representation versus statistical or quasi-raw ones.

3 Classification framework

In this section, the classification framework is detailed. This includes the proposed hierarchical classification model, the classes definition, the data representation, and the classifier architecture.

3.1 Hierarchical classification

As shown in Fig. 4, our classification framework consists of four levels. At Level-1, the traffic is classified as one of four classes: interactive, bulk data transfer, streaming, and transactional. Below is the definition of these classes:

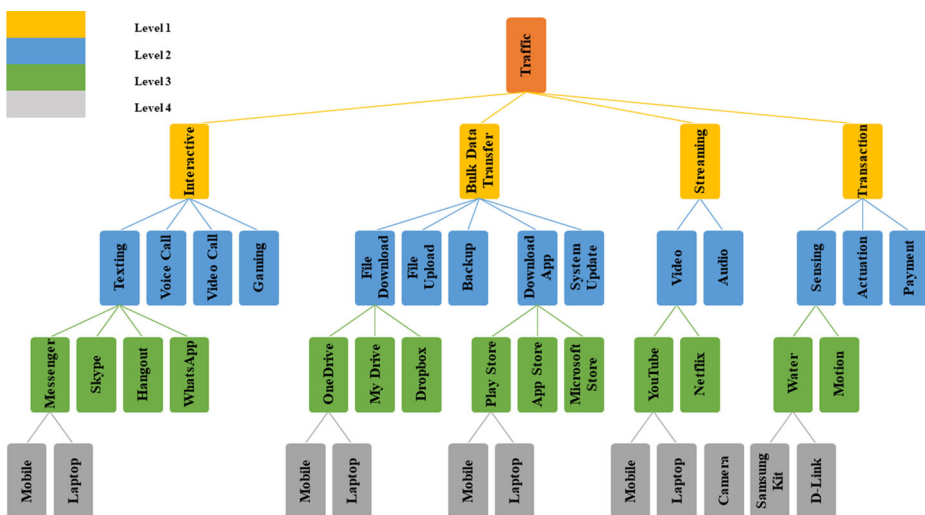


Fig. 4 Hierarchical Internet traffic classification

- *Interactive*: this class is generated by the applications that involve two or multiple users interacting with each other. In this type of traffic, time is of high importance; low latency and jitter are essential requirements.
- *Streaming*: in this type of traffic, time is important since a user is typically waiting for a media file (audio, video, etc.). However, the constraint in terms of jitter is less firm.
- *Bulk data transfer*: this type of traffic is very demanding in terms of bandwidth. It is characterized by the large size of the communicated data.
- *Transactional*: this type of traffic is characterized by a small size of communicated data. It requires high network availability and security.

At Level-2, the classes are related to specific actions of each Level-1 category. For example, for the *interactive* traffic, four classes are defined: *voice call*, *video call*, *texting*, and *gaming*. For *bulk data transfer*, five sub-classes are defined: *file download*, *file upload*, *backup*, *download app*, and *system update*. For the *streaming* traffic, two classes are considered: *video*, and *audio*. Finally, for the *transactional* traffic, three types of actions can be considered: *sensing*, *actuation*, and *payment*.

At Level-3, the classification aims at identifying the application name. Four applications are considered for generating the *texting*, *video call*, and *voice call* traffic, including: *Messenger*, *Skype*, *Hangout*, and *WhatsApp*. For gaming, we consider the *Ball Pool* application. Three applications are considered for the *file download* and *upload* traffic types, including: *One Drive*, *My Drive*, and *Dropbox*. The differentiation between *iOS*, *Android*, and *Windows* is considered for *system update* and *download app*. Moreover, *YouTube* is considered for *video streaming*. For *audio streaming*, *TuneIn radio* and *Anghami* are considered.

At Level-4, we aim at identifying the device type. In fact, different types of devices are considered. For the *interactive* and *bulk data transfer*, *mobile phones*, and *laptops* are used. Moreover, for *sensing* and *actuation*, a set of *sensors* and *actuators* are used to collect IoT traffic.

3.2 Data collection, pre-processing and representation

To train the classification model, data was collected from different types of devices and applications. To do so, we consider a set of IoT devices: D-Link Water Sensor, D-Link



Fig. 5 Data Collection Setup

Camera, D-Link Siren, D-Link Plug, and Samsung Home Kit. To collect traffic from the Wi-Fi connected devices (as shown in Fig. 5a), we configure a laptop (running Linux) as a router, connected to the Internet through the home router. The laptop, configured as a bridge, receives the traffic of the Wi-Fi connected devices at the WLAN interface and forwards it to the Ethernet interface, connected to the home router. This is enabled by adding a rule to the iptables. In this case, to collect traffic, Wire-shark is launched on the WLAN interface of the configured laptop. On the other hand, to collect traffic form the hub connected devices (as shown in Fig. 5b), the hub is connected to the configured laptop by Ethernet and the connection to the Internet is provided by the WLAN interface connected to the home router. In this case, a rule is added to the iptables to forward the incoming traffic from the Ethernet interface to the WLAN one and Wire-shark is launched on the Ethernet interface.

Table 1 summarizes the collected traffic per device and per application. The collection was performed for one hour for each device and application. In addition, we considered two online data-sets to analyze the effect of tunneling and anonymization. The first data-set is the VPN and non-VPN data-set [58], and the other one is a TOR and non-TOR data-set [62]. These data-sets consist of six classes each, including: chat, file transfer, mail, streaming, torrent, and Voice over IP (VoIP).

After collecting traffic, the PCAP files are pre-processed to extract the flows. First, the PCAP files are filtered to get only the TCP and UDP traffic. Then, we extracted the flows by aggregating the packets having the same source IP address, destination IP address, source port number, destination port number, and transport protocol (in either direction). In fact, these flows have different lengths. As DL is a data hungry method, we divide the flows into multiple sub-flows for data augmentation. It should be noted here that the division of the flows into sub-flows is done after dividing the data for training and testing, to avoid having the same flow in the training and testing data-sets, which prevents data peaking. Thus, each

Table 1 Collected data (number of samples)

Level-1		Level-2		Level-3				Level-4		
				Messenger	Whats App	Hangout	Skype	Mobile	Laptop	
Interactive	15020	Voice Call	6092	1368	128	1728	1302	684	684	
								864	864	
		Video Call	5804	1098	124	1896	1004	651	651	
								549	549	
		Texting	892	150	151	300	262	948	948	
								502	502	
		Gaming	104					75	75	
								150	150	
								131	131	
								52	52	
Bulk Data Transfer	37864			Dropbox	MyDrive	OneDrive		Mobile	Laptop	
		File up-load	4935	1112	2678	2702		556	556	
								1339	1339	
		File Down-load	14706	3036	2982	9586		1351	1351	
							1518	1518		
		System Update	322					1491	1491	
		Apps Down-load	4264					4793	4793	
								161	161	
								2132	2132	
Streaming	1918	Video	1542					Mobile	Laptop	Camera
		Audio	416					314	314	314
								2085	208	
								D-Link Plug	D-Link Siren	Samsung Smart Plug
Transaction	558	Actuation	543					181	181	181
								D-Link Water Sensor	Samsung Motion Sensor	Samsung Multi Purpose Sensor
		Sensing	183					61	61	61

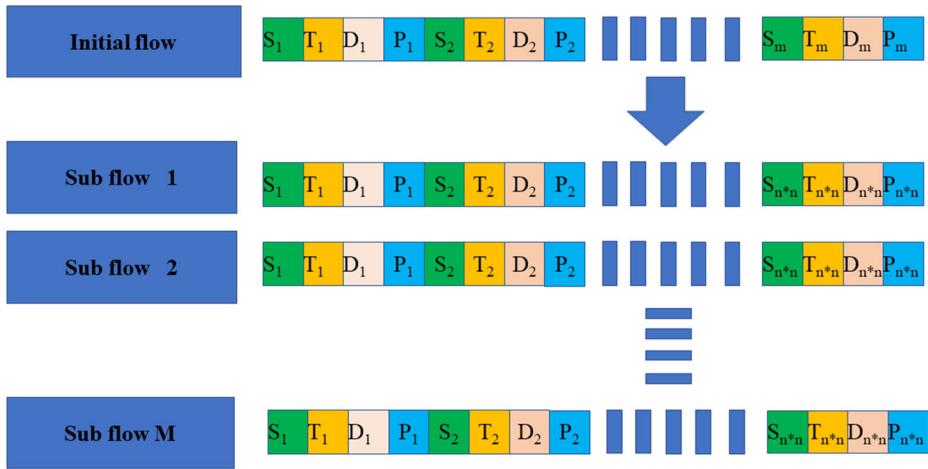


Fig. 6 Flow equalizer

flow is divided into M sub-flows (as shown in Fig. 6), where $M = \frac{m}{n \times n}$ and m is the initial flow length. Each sub-flow consists of $(n \times n)$ packets, from which the size, the inter-arrival time, the direction, and the transport protocol are extracted, resulting into an $(n \times n \times 4)$ features vector. It should be noted also that the extracted features are normalized within the range $[0, 1]$, using the min-max normalization method. To normalize the inter-arrival time, we set its maximum value to 1 second and then for each packet, if the inter-arrival time is greater than 1, the inter-arrival time is set to 1. For the packet size, the maximum is chosen to be 1,500 bytes (Maximum Transmission Unit (MTU)). For each packet, we check if its size is greater than 1,500, if yes, the packet size is set to 1. Then, the packet size is normalized by dividing it by 1,500. For the packet direction, the packets having the same direction of the first packet, their direction is set to 0, otherwise, it is set to 1. The protocol is set to 0 if it is a UDP packet, and to 1 if it is a TCP packet. Thus, the numbers included in

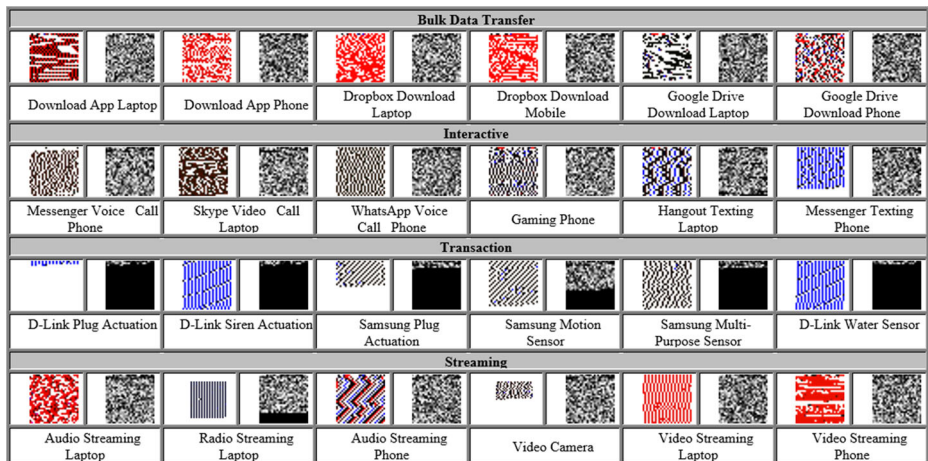


Fig. 7 Data visualization (at left: RGBA method, and at right: Gray method)

Table 1 represent the number of samples after sub-sampling with level-1 consisting of 67896 samples (Fig. 7).

Two data representation methods are considered for comparison. The first method, based on a previous work, considers the first packet of each flow as gray image, as shown in (see Fig. 3a). Thus, we refer to this method as the “Gray method” in the rest of this paper. The second method extracts four features of the first $(n \times n)$ packets of each flow, as shown in (see Fig. 3b). These features are: the packet size, the packet direction, the packet inter-arrival time, and the packet transport protocol. Each packet is represented by an RGBA entry in the obtained image. We refer to the second method as the “RGBA method” in the rest of this paper. A subset of the obtained images is shown in Fig. 7. It is obvious that, visually, the RGBA images present different patterns for the different types of traffic. However, for the gray images, the differentiation between the different classes is not obvious.

3.3 Classifier model

The aim of this study is to compare two data representation methods for CNN-based classification. CNN is a DL architecture that presents a specific architecture for image recognition. CNN consists mainly of four types of layers: convolution layer, pooling layer, dropout layer, and fully connected layer. The convolution layer applies a set of “sliding windows” across the input image. These sliding windows or filters detect the different primitive shapes or patterns. The pooling layer reduces the number of parameters by reducing the image size. It consists of specific operations applied on each feature map independently. One of the most used pooling functions is max pooling. The dropout layer consists of dropping parts of the input, with a defined probability, to avoid model over-fitting. The fully connected layer consists of a set of neurons connected to all previous layer neurons. The architecture, that we used, consists of 3×3 convolutional filters applied at stride 1. We used (2×2) sub-sampling (pooling) layers applied at stride 2. The whole architecture consists of three convolution layers, two pooling layers, two fully connected layers, and one dropout layer, as shown in Fig. 8.

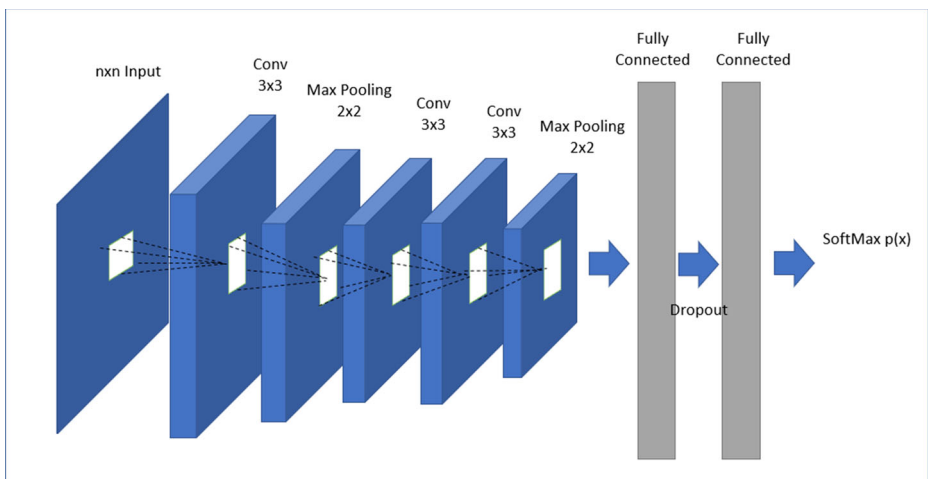


Fig. 8 The used CNN architecture

4 Comparison criteria and methodology

To compare the two data representation methods for machine learning based classification, different aspects must be considered:

- First, the **classification performance** of the representation method is evaluated at the different levels.
- Second, the **features importance** should be analyzed. For traditional machine learning, many methods exist to evaluate the features importance, including entropy based (mutual information), correlation based, and accuracy based (features ranking) methods. However, in the DL case, and more specifically for CNN, the features importance analysis is not tightly related to these measures.
- Third, the **model robustness** is key when comparing different data representation methods. The model robustness is related to its capability of classifying unseen data.
- Fourth, the **features robustness** is another factor that should be examined in the traffic classification domain. Anonymization is a technique to thwart classification by hiding data (encryption), connection metadata (IPs and port numbers), and some traffic characteristics (e.g. packet size, and packets inter-arrival time).

In the following, we present our proposed tests corresponding to the above listed criteria. It should be noted that a subset of Moore features, shown in Table 2, are considered with an RF classifier for comparison with a state-of-the-art method [41, 72].

4.1 Performance test

Algorithm 1 Performance test.

```

1: procedure PERFORMANCE TEST
2:   for  $node_i \in parent\_nodes$  do
3:     train model on train_datai
4:     test model on test_datai
5:   end for
6: end procedure

```

As shown in Algorithm 1, this test is performed for each parent node ($node_i$) at the different levels. A parent node is a node that has leafs. In fact, this type of hierarchical classification is called Local Per Parent Node (LPPN) classification, where a classifier is trained for each non-leaf node. In our case, the classifier is trained with the data corresponding to the direct sub-nodes types ($data_i$). Considering our data-set, at Level-1, the classifier is trained to classify traffic into four classes: *interactive*, *bulk data transfer*, *streaming*, and *transactional*. At Level-2, a classifier for each Level-1 category is trained to classify the traffic into sub-categories. For example, the *interactive* classifier is trained to classify traffic into four classes: *voice call*, *video call*, *texting*, and *gaming*. At Level-3, for each Level-2 class, a classifier is trained to differentiate traffic based on the different applications. For example, the voice call classifier is trained to classify traffic into four classes: *Messenger*, *Skype*, *WhatsApp*, and *Hangout*. At Level-4, a classifier is trained for each Level-3 class based on the different types of generating devices. For example, for the voice call Skype traffic, a classifier is trained to differentiate between *mobile* and *laptop* traffic.

Table 2 Subset of statistical Moore features

Feature	Description
<i>total_fwd_pkt</i>	Total packets in the forward direction
<i>total_fwd_bytes</i>	Total bytes in the forward direction
<i>total_bck_pkt</i>	Total packets in the backward direction
<i>total_bck_bytes</i>	Total bytes in the backward direction
<i>min_pkt_size_fwd</i>	Minimum packet size in the forward direction
<i>mean_pkt_size_fwd</i>	Mean packet size in the forward direction
<i>max_pkt_size_fwd</i>	Maximum packet size in the forward direction
<i>std_pkt_size_fwd</i>	Standard deviation packet size in the forward direction
<i>min_pkt_size_bck</i>	Minimum packet size in the backward direction
<i>mean_pkt_size_bck</i>	Mean packet size in the backward direction
<i>max_pkt_size_bck</i>	Maximum packet size in the backward direction
<i>std_pkt_size_bck</i>	Standard deviation packet size in the backward direction
<i>total_size</i>	Total flow size
<i>min_pkt_size</i>	Minimum packet size in either direction
<i>mean_pkt_size</i>	Mean size of packets in either direction
<i>max_pkt_size</i>	Maximum packet size in either direction
<i>std_pkt_size</i>	Standard deviation packet size in either direction
<i>min_iat_fwd</i>	Minimum inter-arrival time in the forward direction
<i>mean_iat_fwd</i>	Mean inter-arrival time in the forward direction
<i>max_iat_fwd</i>	Maximum inter-arrival time in the forward direction
<i>std_iat_fwd</i>	Standard deviation inter-arrival time in the forward direction
<i>min_iat_bck</i>	Minimum inter-arrival time in the backward direction
<i>mean_iat_bck</i>	Mean inter-arrival time in the backward direction
<i>max_iat_bck</i>	Maximum inter-arrival time in the backward direction
<i>std_iat_bck</i>	Standard deviation inter-arrival time in the backward direction
<i>total_time</i>	Total flow duration
<i>min_iat</i>	Minimum inter-arrival time in either direction
<i>mean_iat</i>	Mean inter-arrival time in either direction
<i>max_iat</i>	Maximum inter-arrival time in either direction
<i>std_iat</i>	Standard deviation inter-arrival time in either direction
<i>avg_pkt_fwd</i>	Average number of packets in the forward direction
<i>avg_bytes_fwd</i>	Average number of bytes in the forward direction
<i>avg_pkt_bck</i>	Average number of packets in the backward direction
<i>avg_bytes_bck</i>	Average number of bytes in the backward direction
<i>avg_iat_fwd</i>	Proportion of flow time in the forward direction to the total flow time
<i>avg_iat_bck</i>	Proportion of flow time in the backward direction to the total flow time

4.2 Features importance test

In this section, we present the features importance tests realized for the packet level and flow level representations. In this test, we consider our data-set, the VPN data-set, and the TOR data-set.

Algorithm 2 Anonymize packet fields in the training phase.

```

1: procedure ANONYMIZE FIELDS TRAINING
2:   while length_of_packet_fields > 0 do
3:     for field ∈ fields_to_anonymize do
4:       anonymize field in train_data
5:       anonymize field in test_data
6:     end for
7:     for field ∈ packet_fields do
8:       train model on train_data
9:       test model on test_data
10:      if current_test_accuracy < min_accuracy then
11:        save field
12:      end if
13:    end for
14:    add field to fields_to_anonymize
15:    remove field from packet_fields
16:  end while
17: end procedure

```

4.2.1 Packet level representation:

The Transmission Control Protocol/Internet Protocol (TCP/IP) packet consists of 25 fields as shown in Table 3.

For the packet level representation, and to analyze the importance of the packet fields, we run an experiment that anonymizes the fields sequentially. At each round, the field showing the minimum accuracy when anonymized is chosen to be anonymized at the next rounds. This is done until all packet fields are anonymized. The aim is to identify the most important fields that influence the classification accuracy. Two cases are considered; the case when the same feature is anonymized for training and testing, as shown in Algorithm 2, and the case where the anonymization is performed only on the testing data, as shown in Algorithm 3. We apply these tests on our data, the TOR data, and the VPN data.

Algorithm 3 Anonymize packet fields in the testing phase.

```

1: procedure ANONYMIZE FIELDS TESTING
2:   train model on train_data
3:   while length_of_packet_fields > 0 do
4:     for field ∈ fields_to_anonymize do
5:       anonymize field in test_data
6:     end for
7:     for field ∈ packet_fields do
8:       test model on test_data
9:       if current_test_accuracy < min_accuracy then
10:        save field
11:      end if
12:    end for
13:    add field to fields_to_anonymize
14:    remove field from packet_fields
15:  end while
16: end procedure

```

Table 3 Packet fields

1	Source MAC Address
2	Destination MAC Address
3	Type IP
4	Version
5	Diff Serv
6	Total Length
7	Identifier
8	Do not Fragment (DF)
9	Fragment Offset
10	Time To Live (TTL)
11	Transport Protocol
12	Header Checksum
13	Source IP Address
14	Destination IP Address
15	Source Port Number
16	Destination Port Number
17	Sequence Number
18	Acknowledgement Number
19	Packet Offset
20	Flags
21	Window Size
22	Checksum
23	Urgent Pointer
24	Options
25	Data

Algorithm 4 Flow level features importance test.

```

1: procedure TRAINING
2:   for  $feature \in flow\_features$  do
3:     train  $model$  on  $feature\_vector$ 
4:     test  $model$  on  $feature\_vector$ 
5:   end for
6: end procedure

```

4.2.2 Flow level representation:

For the flow level representation, we have four main features: packet size, inter-arrival time, protocol, and direction. Thus, to analyze the importance of each of these features, we performed the classification for each $feature_vector$, as shown in Algorithm 4, by representing it by an $(n \times n)$ gray image applied to two images sizes (28×28) and (4×4) .

4.3 Model robustness test

Based on the hierarchical architecture defined in section 2, we design the model robustness test. This test consists of removing one of the sub-classes traffic $data_j$ from the parent

traffic $data_i$ and train the classifier model on the obtained data $data_{i-j}$. Then, the trained model is applied on the removed traffic $data_j$, as presented in Algorithm 5. At Level-1, four classes are defined, including: *interactive*, *bulk data transfer*, *streaming*, and *transaction*. Consequently, the robustness test aims at removing one Level-2 class, at a time, and check if the Level-1 classifier is able to classify it correctly. For example, removing the *texting* class, the Level-1 classifier, trained on all the remaining data without *texting*, is tested to see if it can classify the *texting* traffic as *interactive*.

Algorithm 5 Model robustness test.

```

1: procedure TRAINING
2:   for  $i \in nb\_of\_parent\_nodes$  do
3:     for  $j \in nb\_sub\_classes$  do
4:        $data_{i-j} \leftarrow data_j$  from  $data_i$ 
5:       train model on  $data_{i-j}$ 
6:       test model on  $data_j$ 
7:     end for
8:   end for
9: end procedure

```

4.4 Features robustness test

This test investigates the features robustness towards traffic anonymization. To do so, we considered the TOR and VPN data-sets. We compared the Gray and RGBA methods, by training and testing a model on each of these data-sets consisting of six classes. In doing so, we aim at investigating:

1. The effect of traffic encapsulation by VPN, which consists of traffic encryption and meta-data modification such as the port numbers, IP addresses, etc.
2. The anonymization effect given that TOR affects also the flow statistical features such as the packet length, in addition to the traffic encapsulation effect.
3. The effect of mutation techniques that aim at anonymizing the main traffic characteristics such as packets sizes and/or inter-arrival times. These mutation techniques rely on traffic shaping and padding to manipulate the traffic characteristics in the aim to confuse the classifier.

This test consists of training the model on the original traffic and to test it on the anonymized traffic where three cases are considered: packets sizes are anonymized, packets inter-arrival-times are anonymized, and packets sizes and inter-arrival times are anonymized. What is meant by packet size anonymization is padding all the packet sizes to the MTU, and by inter-arrival time anonymization is setting the packets inter-arrival time to a fixed time interval (i.e. one second).

5 Experimental results

In this section, we present the comparison results. Note that the tests were performed on a Linux machine (Ubuntu 14.04 LTS) with 16 GB RAM and Intel core i7 processor. TFLearn [61] was used as a high-level API for the tensorflow [60] Python library for DL implementation. It should be noted that in all tests, cross validation was applied with 4 folds

and data was divided into 60% for training, 20% for validation, and 20% for testing. For the CNN classifier, the convolution layers were optimized using the Adam optimizer, and the Rectifier Linear Unit (ReLU) function is used for activation. Dropout probability was set to 0.5. The fully connected layer is also optimized using the Adam optimizer, and cross-entropy is used as output function. The learning rate is 0.001, the weights are initialized by the truncated normal distribution, and the biases are initialized to 0. The CNN model was trained for 100 epochs with a batch size equal to 5. The RF classifier was implemented using the scikit-learn python library [57], with grid search to optimize the number of trees with [1,100] as the range for search.

Let us define the metrics used in evaluating the classification performance.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100 \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$f1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

Where *TP* stands for “True Positive”, which is the number of correctly classified instances belonging to a class *C*. *TN* stands for “True Negative”, which is the number of correctly classified instances not belonging to class *C*. *FP* stands for “False Positive”, which is the number of wrongly classified instances belonging to class *C*. Finally, *FN* stands for “False Negative”, which is the number of wrongly classified instances not belonging to class *C*. In fact, the accuracy reflects the power of the classifier to correctly classify the observations. However, this metric is not very indicative when the cost of mis-classification is high or when the data is imbalanced. To solve this issue other metrics should be considered. Precision is one of these metrics that reflects the number of the correctly classified instances to the total instances classified pertaining to a certain class. Another metric is the recall, which reflects the number of correctly classified instances to the total number of instances of this class. Thus, the recall is indicative of the model performance when the data is imbalanced. Finally, f1-score is proportional to the precision and recall, reflecting the imbalanced data and the mis-classification issues. These metrics are computed over the 4 folds and the mean is considered as the final performance evaluation criteria.

5.1 Performance test results

To test the performance of the considered data representation methods, the ConvNet architecture was applied for multi-level classification. The accuracy results are shown in Figs. 9, 10, and 11. It is clear from the results that the RGBA(28x28) representation method achieves overall better results at the different classification levels. At Level-1, as shown in Fig. 9, RGBA(28x28) and RGBA(4x4) produce better results than the Gray method. RGBA(8x8) and RGBA(4x4) show better performance in terms of accuracy, precision, recall, and f1-score. In terms of accuracy, RGBA(28x28) achieves the highest value of 95.84%, followed by RGBA(4x4) with 93.49%, and then Gray with 92.18%. It can be noticed also that RGBA(4x4) shows better results than RGBA(28x28) in some cases. For example, at Level-3, RGBA(4x4) gives better accuracy (82.72%) than both the RGBA(28x28) and Gray methods, 79.59% and 70.4%, respectively.

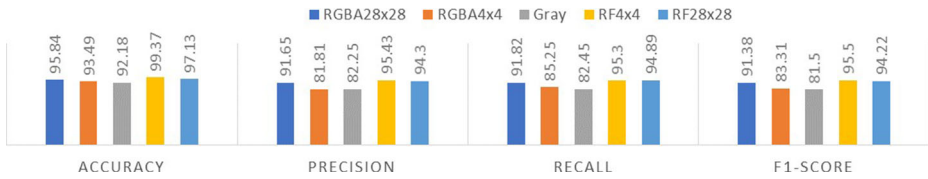


Fig. 9 Performance test results (Level-1)

To summarize, the Gray method failed to surpass the RGBA method at the different classification levels, even though it relies on important information in the packet header. This indicates that this method cannot be generalized on testing data due to the difference in the connection parameters between training and testing data with different flows from different devices. Furthermore, the results show that the RF method achieves better accuracy at the different levels when considering the first (28 × 28) packets, RF(28x28), or the first (4 × 4) packets, RF(4x4).

5.2 Features importance test results

5.2.1 Packet level representation:

For the Gray method, the features importance test results are illustrated in Fig. 12. Figure 12a displays the results of the features importance test when features are anonymized in the training and testing phases using our data. It can be noticed that source IP address, source MAC address, and the destination MAC address are the three most important fields. Anonymizing these fields in training and testing drops the accuracy by more than 10%. In the testing phase, the drop in accuracy is more pronounced and thus, as shown in (see Fig. 12b), anonymizing only the data drops the accuracy by 10%. Moreover, anonymizing data, source MAC address, and ACK number drops the accuracy noticeably to 10%. For the TOR data, the accuracy is already low compared to the flow level representation. In the first case, where anonymization is applied on the training and testing data, the destination port was the most important feature, as shown in (see Fig. 12c). However, in the second case, the anonymization of the source MAC and destination MAC addresses drops the accuracy noticeably to 15%, as shown in (see Fig. 12d). Finally, for the VPN data, anonymizing the

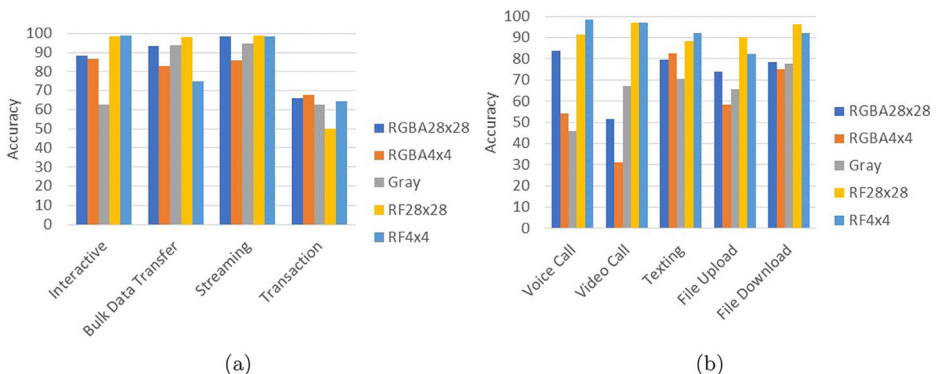


Fig. 10 Performance test results in terms of accuracy: a for Level-2 and b for Level-3

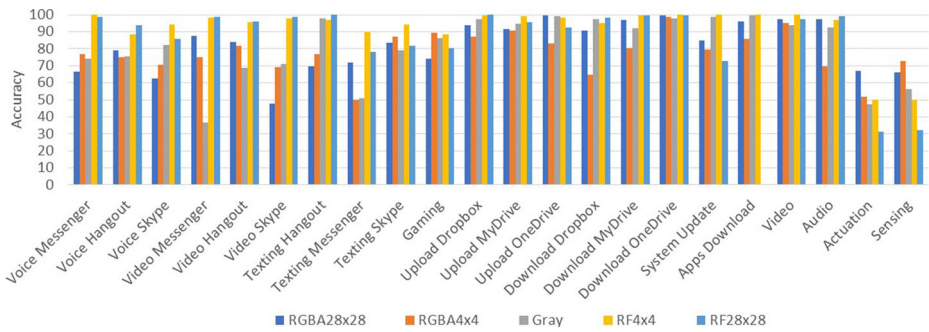


Fig. 11 Performance test results in terms of accuracy (Level-4)

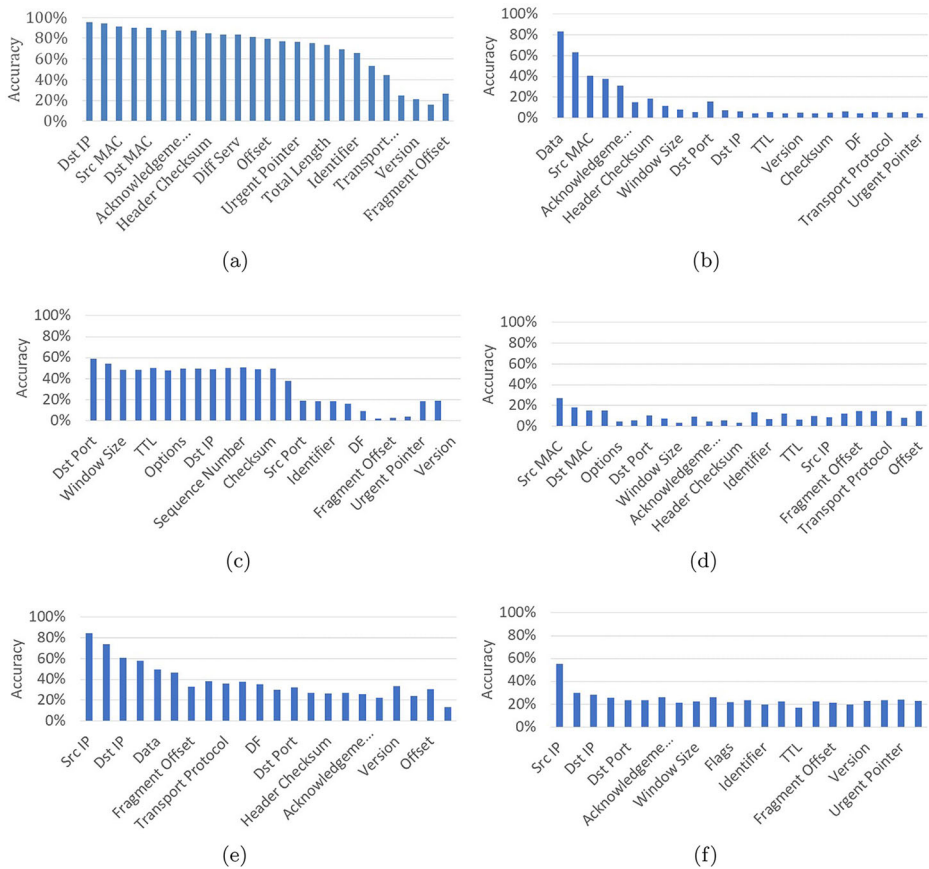


Fig. 12 Gray method features importance results in terms of accuracy: **a**, **c**, and **e** for anonymization at training and testing phases using our data, TOR, and VPN data-sets, respectively; **b**, **d**, and **f** for anonymization at the testing phase using our data, Tor, and VPN data-sets, respectively

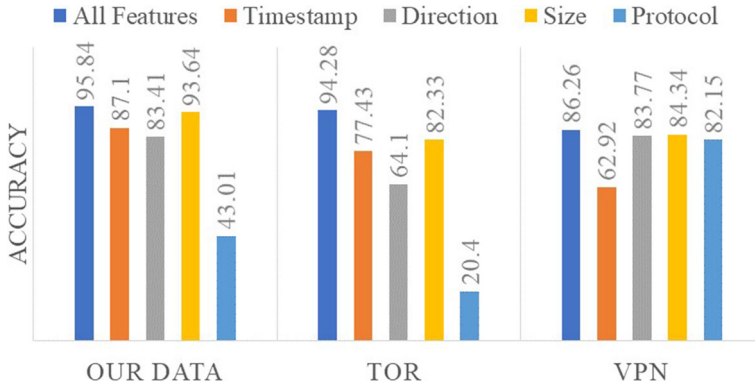


Fig. 13 RGBa(28x28) method features importance results in terms of accuracy

IP addresses drops the accuracy to 60%, in the first case. However, in the testing case, the accuracy drop was more pronounced for the same features.

To summarize, the results of this test show that the Gray method’s performance is data dependent. Thus, if the testing data is collected from different network environments, the model performance will decrease. Moreover, eliminating some fields in the training phase affects the learning power, given that the remaining packet data did not present any recognizable pattern.

5.2.2 Flow level representation:

For the RGBa method, The features importance test results are included in Figs. 13 and 14. For the (28x28) images, it can be noticed that packet size is the most important feature. This can be observed since the accuracy, when using the packet sizes of the first (28x28) packets of a flow, is the highest relative to other features. Similarly, for the RGBa(4x4) representation, the packet size is the most important feature for our data and the TOR data. However, for the VPN data, the packets’ inter-arrival time is the most important feature.

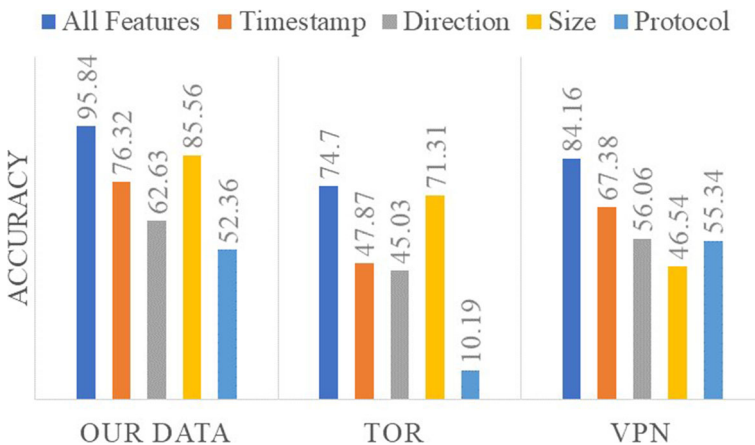


Fig. 14 RGBa(4x4) method features importance results in terms of accuracy

It can be noted that the RGBA method, including statistical features, enables the generalization of the trained model when the data is collected from different network setups. Moreover, the accuracy results obtained for VPN and TOR data show their immunity towards traffic anonymization. Moreover, the direction feature, which reflects a behavioral pattern of the traffic, can serve the classification when some mutation techniques, modifying the packet size or inter-arrival time, are applied.

5.3 Model robustness test results

The features robustness test results, shown in Fig. 15, illustrate the accuracy results of the robustness test for Level-1 classes. It is clear that RGBA(28x28) and sometimes RGBA(4x4) achieve better results than the Gray method. Similarly, in Fig. 16, the results of Level-2 robustness tests are shown. In these tests, for each Level-2 class, we remove one of its subclasses to see if the Level-2 classifier is able to classify it correctly. For example, for texting, we remove one of the texting applications (e.g. hangout) and we test the classifier trained on the interactive type of traffic (texting, voice call, video call, and gaming) to check if it will classify the hangout traffic correctly. The results show that RGBA is better than the Gray method in more than 70% of the cases. Moreover, if we consider the RF method, the results show that the RGBA method is more robust than the RF method with statistical features in most cases.

5.4 Features robustness test results

For TOR and VPN traffic, the results presented in Table 4 show that, for both types of traffic, the RGBA method gives better results than the gray one. It can be noted that the Gray method performance degrades noticeably when traffic is encrypted (78% accuracy on VPN data-set). Moreover, the RGBA method gives promising results on TOR traffic (94% accuracy). Moreover, it can be noticed that the RF method with statistical features gives better results than the CNN-based classification with flow-based or packet-based representation when considering the first (4x4) packets or (28x28) packets. However, as shown in Fig. 17, it is clear that the RF-based method performance degrades noticeably when the size and/or

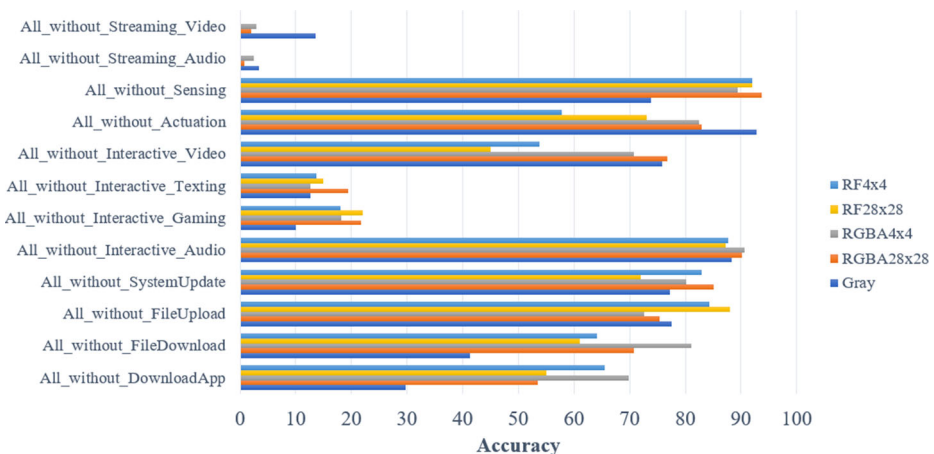


Fig. 15 Robustness test results in terms of accuracy (Level-1)

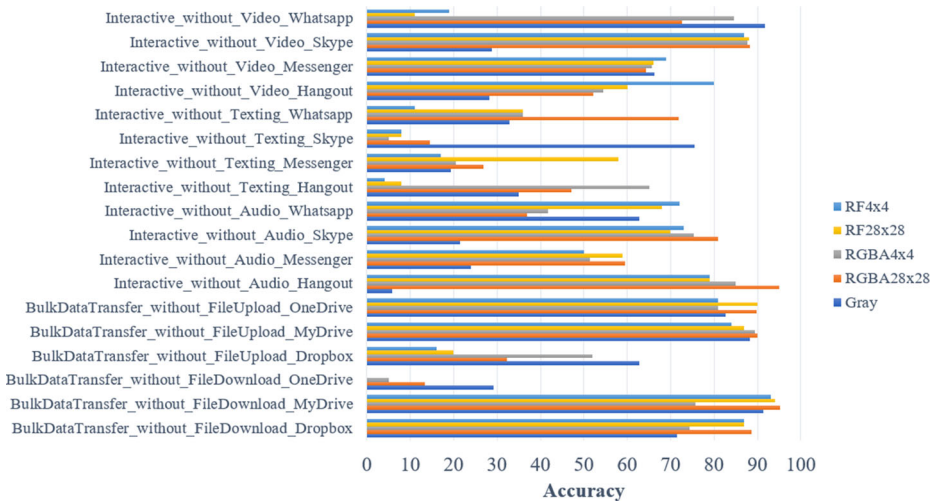


Fig. 16 Robustness test results in terms of accuracy (Level-2)

inter-arrival times of the packets are mutated. This indicates that the statistical features are prone to adversarial attacks making the statistical-based classification performance prone to degradation. The CNN-based classification, considering behavioral features (i.e. packets directions), can give better results when one or more channels (size and inter-arrival time) are mutated. Similarly, the anonymization of the packet sizes and/or inter-arrival times test is performed on our data-set at all levels. The results are shown in Fig. 18. It can be noticed that the drop in accuracy, for all levels, is more pronounced for the RF-based method than for the RGBA one. Thus, the RGBA method is more robust when one or more features are anonymized. This can be explained by the fact that the behavioral pattern of the packets, reflected by the direction map, can rescue the loss of information in other channels. Moreover, the anonymization can be compared to a noise added to an image and thus, in this case, image recognition is still possible using CNN.

6 Discussions

Image recognition is a well-established CNN application. Thus, applying CNN for traffic classification calls for new representation methods of the traffic as images. In this paper, we compared two state-of-the-art methods. The first considers the first packet of the flow as raw data ($n \times n$) image, and the second considers four features from the first ($n \times n$) packets of the flow to form an RGBA ($n \times n$) image. The results show the dependency of the raw packet data representation on some dynamic connection parameters (e.g. MAC, IP

Table 4 Accuracy Results (in%) with the TOR and VPN data-sets

	RGBA(28x28)	RGBA(4x4)	Gray	RF(28x28)	RF(4x4)
VPN	86.26%	84.16%	78.68%	93.96%	88.34%
TOR	94.28%	74.7%	64.78%	97.92%	88.13%

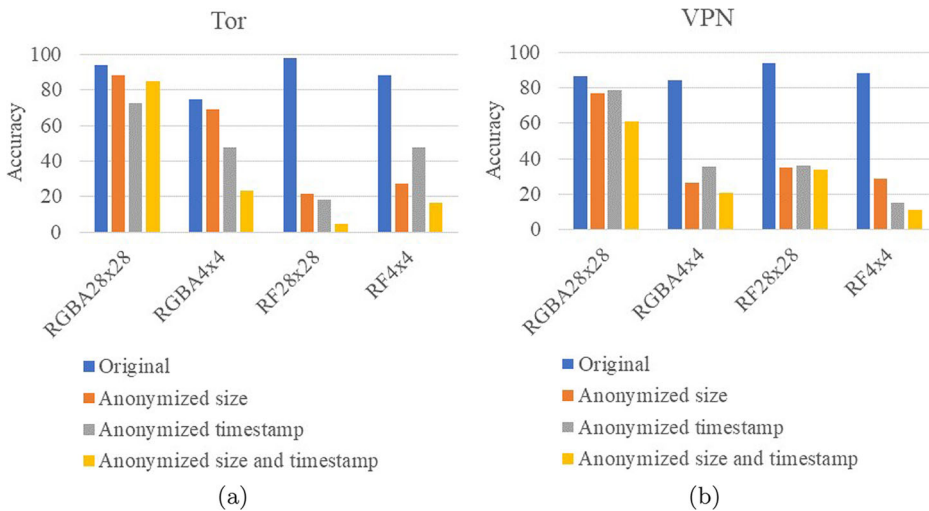


Fig. 17 Anonymization results in terms of accuracy considering the TOR and VPN data-sets

addresses), which hinders the generalization of the obtained model when applied on data collected from different network setups. On the other hand, the second method needs tuning for the number of packets needed for classification. In addition, relying on statistical flow features makes this method prone to accuracy drop in case of obfuscated traffic.

Considering a variety of classes, in a hierarchical type of classification, allows for the design of new tests that evaluate the similarity of the traffic images belonging to the same class. Thus, quantitatively (robustness test) and qualitatively (data visualization), the RGBA images present common patterns for traffic pertaining to the same class. This is clearly noticeable for the high-level classes (Levels 1 and 2). However, when it comes down to more granular classes, the accuracy and the visual differentiation is not obvious. On the other hand, the gray method does not provide visually discernable patterns, but the achieved accuracy at different levels are comparable to those obtained with the RGBA method. However, the features importance tests show that the anonymization of certain packet fields, which are network connection dependent, result in a significant change in the classification accuracy.

Finally, the anonymization test results show that the RGBA method provides better accuracy than the Gray one, when trained on tunneled (VPN) or anonymized traffic (TOR). This gives a clear indication that statistical features can lead to distinctive patterns when the packets content or metadata are anonymized. Consequently, this presents a new security challenge, where user privacy is the aim. In this case, privacy preserving methods such as mutation and morphing could be employed.

7 Conclusion

An essential benefit of DL over traditional machine learning methods is its representation learning capability. More specifically, CNN was shown to be very effective for image classification. Consequently, representing traffic as images has been considered in the last two years. One direction was to represent the first packet as a gray image and thus, traffic raw data is considered for classification. Another method consists of extracting four features

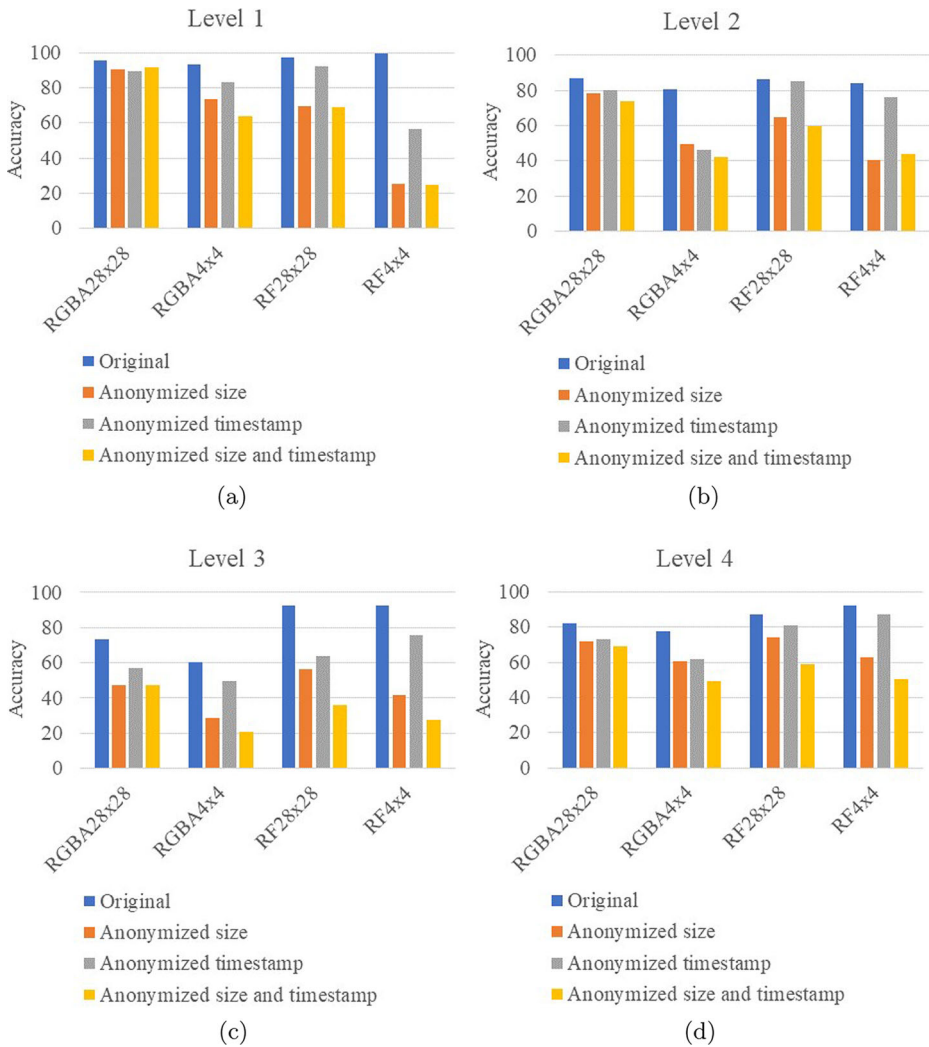


Fig. 18 Features robustness test results in terms of accuracy for all levels considering the RGBA(28x28), RGBA(4x4), RF(28x28), and RF(4x4): a) Level-1, b) Level-2, c) Level-3, and d) Level-4

from the first ($n \times n$) packets per flow. In this paper, we compared these two representation methods. Three types of tests were proposed, features importance test, model robustness, and anonymization test. The results showed that the features should be carefully selected such that they are not affected by the network environment or by possible obfuscation techniques. In such a case, the inter-correlation of the results obtained from different traffic characteristics (statistical: size and timestamp, and behavioral: direction) could help in detecting obfuscated traffic and thus, in avoiding misclassification.

Acknowledgments Research funded by the AUB University Research Board, the Lebanese National Council for Scientific Research, and TELUS Corp., Canada.

References

1. Acar A, Fereidooni H, Abera T, Sikder AK, Miettinen M, Aksu H, Conti M, Sadeghi AR, Uluagac AS (2018) Peek-a-boo: I see your smart home activities, even encrypted! arXiv:1808.02741
2. Aceto G, Ciuonzo D, Montieri A, Pescapé A (2019) Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Trans Netw Serv Manag* 16(2):445–458. <https://doi.org/10.1109/TNSM.2019.2899085>
3. Al Khater N, Overill RE (2015) Network traffic classification techniques and challenges. In: 2015 Tenth international conference on digital information management (ICDIM). pp 43–48. <https://doi.org/10.1109/ICDIM.2015.7381869>
4. Alizadeh H, Zúquete A (2016) Traffic classification for managing applications' networking profiles. *Secur Commun Netw* 9(14):2557–2575
5. Aureli D, Cianfrani A, Diamanti A, Vilchez JMS, Secci S (2020) Going beyond diffserv in ip traffic classification. In: IEEE/IFIP Network operations and management symposium (NOMS)
6. Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. *IEEE Trans Pattern Analysis Mach Intell* 35(8):1798–1828
7. Biersack E, Callegari C, Matijasevic M (2013) Data traffic monitoring and analysis: from measurement, classification, and anomaly detection to quality of experience. Springer, vol 7754
8. Cao J, Chen A, Widjaja I, Zhou N (2008) Online identification of applications using statistical behavior analysis. In: IEEE GLOBECOM 2008–2008 IEEE Global telecommunications conference. IEEE, pp 1–6
9. Chen Z, Yu B, Zhang Y, Zhang J, Xu J (2016) Automatic mobile application traffic identification by convolutional neural networks. In: 2016 IEEE Trustcom/bigdataSE/ISPA. pp 301–307. <https://doi.org/10.1109/TrustCom.2016.0077>
10. Chen Z, He K, Li J, Geng Y (2017) Seq2img: a sequence-to-image based approach towards ip traffic classification using convolutional neural networks. In: 2017 IEEE International conference on big data (big data). pp 1271–1276. <https://doi.org/10.1109/BigData.2017.8258054>
11. Chung JY, Choi Y, Park B, Hong JWK (2011) Measurement analysis of mobile traffic in enterprise networks. In: 2011 13Th asia-pacific network operations and management symposium. IEEE, pp 1–4
12. Conti M, Mancini LV, Spolaor R, Verde NV (2015) Analyzing android encrypted network traffic to identify user actions. *IEEE Trans Inf Forensic Secur* 11(1):114–125
13. Dai S, Tongaonkar A, Wang X, Nucci A, Song D (2013) Networkprofiler: Towards automatic fingerprinting of android apps. In: 2013 Proceedings IEEE INFOCOM, pp 809–817. <https://doi.org/10.1109/INFCOM.2013.6566868>
14. Dainotti A, Pescapé A, Kim H (2011) Traffic classification through joint distributions of packet-level statistics. In: 2011 IEEE Global telecommunications conference - GLOBECOM 2011, pp 1–6. <https://doi.org/10.1109/GLOCOM.2011.6134093>
15. Dainotti A, Pescapé A, Claffy KC (2012) Issues and future directions in traffic classification. *IEEE Netw* 26(1):35–40
16. Este A, Gringoli F, Salgarelli L (2009) Support vector machines for tcp traffic classification. *Comput Netw* 53(14):2476–2490
17. Fadlullah ZM, Tang F, Mao B, Kato N, Akashi O, Inoue T, Mizutani K (2017) State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Commun Surv Tutor* 19:2432–2455. [10.1109/COMST.2017.2707140](https://doi.org/10.1109/COMST.2017.2707140)
18. Filiposka S, Mishkovski I (2013) Smartphone user's traffic characteristics and modelling. *Trans Netw Commun* 1(1):14–40
19. Fu Y, Xiong H, Lu X, Yang J, Chen C (2016) Service usage classification with encrypted internet traffic in mobile messaging apps. *IEEE Trans Mob Comput* 15(11):2851–2864. <https://doi.org/10.1109/TMC.2016.2516020>
20. Gonzalez R, Manco F, Garcia-Duran A, Mendes J, Huici F, Niccolini S, Niepert M (2017) Net2vec: Deep learning for the network. arXiv:1705.03881
21. Goo YH, Shim KS, Lee SK, Kim MS (2016) Payload signature structure for accurate application traffic classification. In: 2016 18Th asia-pacific network operations and management symposium (APNOMS), . IEEE, pp 1–4
22. Greenspan H, Van Ginneken B, Summers RM (2016) Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique. *IEEE Trans Med Imaging* 35(5):1153–1159
23. Haffner P, Sen S, Spatscheck O, Wang D (2005) Acas: automated construction of application signatures. In: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data. ACM, pp 197–202
24. Headquarters A Qos: Classification configuration guide, cisco ios xe release 3s
25. Hu Y, Chiu DM, Lui JC (2008) Application identification based on network behavioral profiles. In: 2008 16Th international workshop on quality of service. IEEE, pp 219–228

26. Huang H, Deng H, Chen J, Han L, Wang W (2018) Automatic multi-task learning system for abnormal network traffic detection. *Int J Emerging Technol Learn* 13(4)
27. Hur M, Kim MS (2012) Towards smart phone traffic classification. In: 2012 14Th asia-pacific network operations and management symposium (APNOMS). IEEE, pp 1–4
28. Jaiganesh V, Mangayarkarasi S, Sumathi P (2013) Intrusion detection systems: a survey and analysis of classification techniques. *Int J Adv Res Comput Commun Eng* 2(4):1629–1635
29. Karagiannis T, Papagiannaki K, Faloutsos M (2005) Blinc: multilevel traffic classification in the dark. In: ACM SIGCOMM Computer communication review, vol 35. ACM, pp 229–240
30. Karagiannis T, Papagiannaki K, Taft N, Faloutsos M (2007) Profiling the end host. In: International conference on passive and active network measurement. Springer, pp 186–196
31. Lee SW, Park JS, Lee HS, Kim MS (2011) A study on smart-phone traffic analysis. In: 2011 13Th asia-pacific network operations and management symposium. IEEE, pp 1–7
32. Leroux S, Bohez S, Maenhaut P, Meheus N, Simoens P, Dhoedt B (2018) Fingerprinting encrypted network traffic types using machine learning. In: NOMS 2018 - 2018 IEEE/IFIP Network operations and management symposium, pp 1–5 <https://doi.org/10.1109/NOMS.2018.8406218>
33. Li Z, Qin Z, Huang K, Yang X, Ye S (2017) Intrusion detection using convolutional neural networks for representation learning. In: International conference on neural information processing. pp. 858–866. Springer
34. Liu Y, Zhang S, Ding B, Li X, Wang Y (2018) A cascade forest approach to application classification of mobile traces. In: 2018 IEEE Wireless communications and networking conference (WCNC). IEEE. pp 1–6
35. Liu Z, Wang R, Japkowicz N, Cai Y, Tang D, Cai X (2019) Mobile app traffic flow feature extraction and selection for improving classification robustness. *J Netw Comput Appl* 125:190–208. <https://doi.org/10.1016/j.jnca.2018.10.018> <http://www.sciencedirect.com/science/article/pii/S1084804518303400>
36. Lopez-Martin M, Carro B, Sanchez-Esguevillas A, Lloret J (2017) Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access* 5:18042–18050. <https://doi.org/10.1109/ACCESS.2017.2747560>
37. Lotfollahi M, Siavoshani MJ, Zade RSH, Saberian M (2017) Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Comput*:1–14
38. Maier G, Schneider F, Feldmann A (2010) A first look at mobile hand-held device traffic. In: International conference on passive and active network measurement. Springer, pp 161–170
39. Marnierides AK, Schaeffer-Filho A, Mauthe A (2014) Traffic anomaly diagnosis in internet backbone networks: a survey. *Comput Netw* 73:224–243
40. Meiss M, Menczer F, Vespignani A (2011) Properties and evolution of internet traffic networks from anonymized flow data. *ACM Trans Internet Technol (TOIT)* 10(4):15
41. Michael AKJ, Valla E, Neggatu NS, Moore A (2017) Network traffic classification via neural networks. Technical report. University of Cambridge, Computer Laboratory
42. Mitevski B, Filiposka S (2013) Smartphone traffic review. In: International conference on ICT innovations, Springer. pp 291–301
43. Mongkolluksamee S, Visoottiviseth V, Fukuda K (2015) Enhancing the performance of mobile traffic identification with communication patterns. In: 2015 IEEE 39Th annual computer software and applications conference, vol 2 <https://doi.org/10.1109/COMPASAC.2015.50>, pp 336–345
44. Mongkolluksamee S, Visoottiviseth V, Fukuda K (2016) Combining communication patterns & traffic patterns to enhance mobile traffic identification performance. *J Inf Process* 24(2):247–254
45. Moore A, Papagiannaki K (2005) Toward the accurate identification of network applications. In: International workshop on passive and active network measurement. Springer, pp 41–54
46. Moore A, Zuev D, Crogan M (2013) Discriminators for use in flow-based classification. Technical report
47. Murgia A, Ghidini G, Emmons SP, Bellavista P (2016) Lightweight internet traffic classification: a subject-based solution with word embeddings. In: 2016 IEEE International conference on smart computing (SMARTCOMP), pp 1–8. <https://doi.org/10.1109/SMARTCOMP.2016.7501703>
48. Nowak J, Korytkowski M, Nowicki R, Scherer R, Siwocha A (2018) Random forests for profiling computer network users. In: International conference on artificial intelligence and soft computing. Springer, pp 734–739
49. Okabe T, Kitamura T, Shizuno T (2006) Statistical traffic identification method based on flow-level behavior for fair voip service. In: 1St IEEE workshop on voIP management and security, pp 35–40. <https://doi.org/10.1109/VOIPMS.2006.1638120>
50. Pacheco F, Exposito E, Gineste M, Baudoin C, Aguilar J (2018) Towards the deployment of machine learning solutions in network traffic classification: a systematic survey. *IEEE Commun Surv Tutorials* 21(2):1988–2014

51. Parckekani A, Naghadeh SN, Shah-Mansouri V (2020) Classification of traffic using neural networks by rejecting: a novel approach in classifying vpn traffic. arXiv:2001.03665
52. Salman O, Elhadj I. H, Chehab A, Kayssi A (2018) A multi-level internet traffic classifier using deep learning. In: 2018 9Th international conference on the network of the future (NOF), pp 68–75
53. Salman O, Elhadj IH, Chehab A, Kayssi A (2019) A machine learning based framework for iot device identification and abnormal traffic detection. *Trans Emerg Telecommun Technol* 0(0):e3743
54. Salman O, Elhadj IH, Kayssi A, Chehab A (2020) A review on machine learning–based approaches for internet traffic classification. *Ann Telecommun*:1–38
55. Schmidt B, Al-Fuqaha A, Gupta A, Kountanis D (2017) Optimizing an artificial immune system algorithm in support of flow-based internet traffic classification. *Appl Soft Comput* 54:1 – 22. <https://doi.org/10.1016/j.asoc.2017.01.016>, <http://www.sciencedirect.com/science/article/pii/S1568494617300285>
56. Shi H, Li H, Zhang D, Cheng C, Cao X (2018) An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification. *Comput Netw* 132:81 – 98. <https://doi.org/10.1016/j.comnet.2018.01.007>, <http://www.sciencedirect.com/science/article/pii/S1389128618300082>
57. scikit-learn: machine learning in python — scikit-learn 0.22 documentation. <https://scikit-learn.org/stable/>, (Accessed on 01/01/2020)
58. Vpn-nonvpn dataset (iscxvpn2016). <https://www.unb.ca/cic/datasets/vpn.html>
59. Tahaei H, Afifi F, Asemi A, Zaki F, Anuar NB (2020) The rise of traffic classification in iot networks: A survey. *J Netw Comput Appl*:102538
60. Tensorflow. <https://www.tensorflow.org/>
61. Tflern — tensorflow deep learning library. <http://tflern.org/>
62. Tor-nontor dataset (iscxtor2016). <https://www.unb.ca/cic/datasets/tor.html>
63. Tongaonkar A, Keralapura R, Nucci A (2012) Challenges in network application identification. In: Presented as part of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats. USENIX, San Jose. <https://www.usenix.org/challenges-network-application-identification-10k-tongaonkar>
64. Wang Z (2015) The applications of deep learning on traffic identification. BlackHat, USA, pp 24
65. Wang W, Zhu M, Wang J, Zeng X, Yang Z (2017) End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In: 2017 IEEE International conference on intelligence and security informatics (ISI). pp 43–48. <https://doi.org/10.1109/ISI.2017.8004872>
66. Wang W, Sheng Y, Wang J, Zeng X, Ye X, Huang Y, Zhu M (2018) Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* 6:1792–1806
67. Wang W, Zhang X, Shi W, Lian S, Feng D (2011) Network traffic monitoring, analysis and anomaly detection [guest editorial]. *IEEE Netw* 25(3):6–7
68. Wang W, Zhu M, Zeng X, Ye X, Sheng Y (2017) Malware traffic classification using convolutional neural network for representation learning. In: 2017 International conference on information networking (ICOIN). IEEE, pp 712–717
69. Xu Q, Andrews T, Liao Y, Miskovic S, Mao ZM, Baldi M, Nucci A (2014) Flowr: a self-learning system for classifying mobile application traffic. *ACM SIGMETRICS Perform Eval Rev* 42(1):569–570
70. Xu Q, Liao Y, Miskovic S, Mao ZM, Baldi M, Nucci A, Andrews T (2015) Automatic generation of mobile app signatures from traffic observations. In: 2015 IEEE Conference on computer communications (INFOCOM). IEEE, pp 1481–1489
71. Yu K, Liu Y, Qing L, Wang B, Cheng Y (2018) Positive and unlabeled learning for user behavior analysis based on mobile internet traffic data. *IEEE Access* 6:37568–37580
72. Zhang J, Chen X, Xiang Y, Zhou W, Wu J (2014) Robust network traffic classification. *IEEE/ACM Trans Netw* 23(4):1257–1270
73. Zhang Z, Zhang Z, Lee PP, Liu Y, Xie G (2014) Toward unsupervised protocol feature word extraction. *IEEE Journal on Selected Areas in Communications* 32(10):1894–1906
74. Zhang C, Patras P, Haddadi H (2019) Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials*