

AMERICAN UNIVERSITY OF BEIRUT

Machine Learning for Internet Traffic
Classification

by
OLA SALMAN

A dissertation
submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy of Engineering
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
February 2020

AMERICAN UNIVERSITY OF BEIRUT

Machine Learning for Internet Traffic Classification

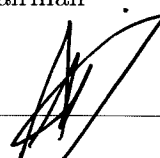
by
Ola Salman

Approved by:


Dr. Ayman Kayssi, Professor
Electrical and Computer Engineering


Chairman

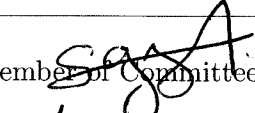
Dr. Imad H. Elhadj, Professor
Electrical and Computer Engineering


Advisor

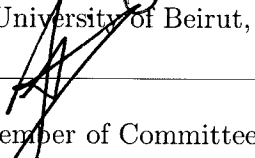
Dr. Ali Chehab, Professor
Electrical and Computer Engineering


Member of Committee

Dr. Georges Sakr, Professor
Electrical and Computer Engineering, Saint Joseph University of Beirut, Lebanon


Member of Committee

Dr. Ghassan AlRegib, Professor
Electrical and Computer Engineering, Georgia Tech, USA


Member of Committee

Date of dissertation defense: February 10, 2020

AMERICAN UNIVERSITY OF BEIRUT

DISSERTATION RELEASE FORM

Student Name: Salman _____ Ola _____ Mohamad _____
Last First Middle


Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One ___ year from the date of submission of my dissertation .**

Two ___ years from the date of submission of my dissertation .

Three ___ years from the date of submission of my dissertation .



Signature

20, Feb, 2020

Date

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Imad Elhajj for his continuous support of my PhD study and related research. His guidance helped me in all the time of research and writing of this thesis. I would like also to thank Prof. Ali Chehab and Prof. Ayman Kayssi for their continuous support during the last five years.

I would like to thank the rest of my thesis committee: Prof. Georges Sakr and Prof. Ghassan ElRegib, for their insightful comments and encouragement.

I would like also to thank AUB University Research Board, the Lebanese National Council for Scientific Research, and TELUS Corp., Canada for funding this research. Especially, I would like to thank Mr. Marc Kneppers from TELUS for his support.

Last but not the least, I would like to thank my friends and my family: my parents, my brothers and sister for their continuous support throughout my life.

An Abstract of the Dissertation of

Ola Salman for Doctor of Philosophy
Major: Electrical and Computer Engineering

Title: Machine Learning for Internet Traffic Classification

Traffic classification acquired the interest of the Internet community early on. Different approaches have been proposed to classify Internet traffic to manage both security and Quality of Service (QoS). However, traditional classification approaches consisting of modifying the Transmission Control Protocol/Internet Protocol (TCP/IP) scheme have not been adopted due to their complex management. In addition, port-based methods and deep packet inspection have limitations in dealing with new traffic characteristics (e.g., dynamic port allocation, tunneling, encryption). Conversely, Machine Learning (ML) solutions effectively classify traffic down to the device type and specific user action. Different ML based methods have been applied for this aim. However, traditional ML methods rely on hand crafted features, limiting the model ability to learn. Deep learning (DL), a branch of ML, is characterized by its representation learning ability. on the other hand, intrusion detection systems (IDSes) have been proposed to detect and/or prevent security attacks by inspecting and detecting attacks patterns. However, traditional IDSes are rule based and present high management complexity. Alternatively, ML-based IDSes have emerged to overcome the management complexity issue. However, many of the proposed solutions are based on hand crafted features and considered datasets are outdated. In this thesis, we propose a new data representation method to apply DL for traffic classification and intrusion detection. For traffic classification, a hierarchical classification framework is proposed to classify the traffic based on different granularity levels. The defined classes reflect the different QoS and security needs. We analyse two methods of data representation for DL-based classification: a raw packet representation and a flow-based representation. Different tests are performed to evaluate the robustness of the considered data representation methods. These 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. Moreover, the statistical features are prone to modification if the main flow characteristics (packets sizes and inter-arrival times) are manipulated. In this context, another research direction is emerging in the aim to anonymize Internet traffic and thwart classification to maintain user privacy. Traffic mutation is one such obfuscation technique, that consists of modifying the flow's statistical characteristics to mislead the traffic classifier. In fact, this same technique can also be used to hide normal traffic characteristics for the sake of privacy. However, the concern is its use by attackers to bypass intrusion detection systems by modifying the attack traffic characteristics. To tackle this problem, we propose an unsupervised DL-based model to detect mutated traffic. This model is based on generative DL architectures, namely Autoencoders (AE) and Generative Adversarial Network (GAN). This model consists of a denoising AE to de-anonymize the mutated traffic and a discriminator to detect it. The implementation results show that the traffic can be denoised when different mutation techniques are applied with a low reconstruction error and high detection rate. Another obfuscation technique that aims at imitating traffic characteristics to make one traffic look like another one. In this thesis, we propose also a Generative Adversarial DL-based method to detect morphed traffic. The proposed method consists of a combination of Variational Autoencoder (VAE) and GAN. The VAE aims at inferring the probability distribution of the original traffic and morphing it, while the GAN's discriminator is trained to detect the fake (morphed) traffic. The testing results show that we can generate fake traffic that is very similar to the original one, and that the discriminator is able to detect the morphed traffic, with a high detection rate. Deciding on the number of packets that should be considered per flow is not a straightforward task. While considering a subset of packets or a window of time per flow was considered in previous work for real-time classification without defining a stopping criteria, in our case, we treat the flows as time series and we aim to define a criteria guaranteeing good accuracy and low response time. To do so, we consider the mutual information between the flows features and the class labels, while increasing n . Different than the traditional mutual information based features reduction methods, in our case, the traffic nature (streaming data) calls for new features selection and reduction method. In this context, a confidence measure is proposed based on the training accuracy and the mutual information between the data and the class vector. The obtained results show that our proposed measure results in choosing n that meets the balance between real time traffic classification, high accuracy, and less training overhead. Finally, we propose a classification model where a consecutive set of classifiers of different dimensions is considered as an ensemble classifier. The implementation results show that the ensemble method is able to enhance the accuracy of the individual

classifiers to reach better classification performance at early stages.

Contents

Acknowledgements	v
Abstract	vi
1 Introduction	1
1.1 Contributions	3
1.2 List of Publications	4
1.3 Organization	5
2 Background and Related Work	7
2.1 Internet Traffic Data Challenges	7
2.1.1 Data Size	9
2.1.2 Data Labeling	10
2.1.3 Data Timeliness	10
2.2 Data Representation	11
2.2.1 Behavioral-based	11
2.2.2 Vectors	12
2.2.3 Time series	14
2.2.4 Word embeddings	14
2.2.5 Images	15
2.3 Machine Learning Methods	16
2.3.1 Supervised	17
2.3.2 Unsupervised	21
2.3.3 Semi-Supervised	25
2.4 Traffic Classification Classes	28
2.4.1 Protocols Classification	28
2.4.2 Applications Classification	30
2.4.3 Actions Classification	31
2.4.4 Categories Classification	31
2.4.5 Devices Classification	33
2.5 Thwarting Internet Traffic Classification	35
2.5.1 Encryption	35
2.5.2 Steganography	36

2.5.3	Tunneling	37
2.5.4	Anonymization	38
2.5.5	Mutation	38
2.5.6	Morphing	40
2.5.7	Physical Layer Obfuscation	41
2.6	Key Findings, Limitations, and Recommendations	42
2.6.1	Key findings	42
2.6.2	Limitations	43
2.7	Conclusion	45
3	Data Representation	46
3.1	Classification Framework	46
3.1.1	Hierarchical Classification	47
3.1.2	Data Collection, Preprocessing and Representation	48
3.1.3	Classifier Model	50
3.2	Comparison Criteria and Methodology	51
3.2.1	Performance Test	54
3.2.2	Features Importance Test	55
3.2.3	Model Robustness Test	57
3.2.4	Features Robustness Test	58
3.3	Experimentation Results	59
3.3.1	Performance Test Results	59
3.3.2	Features Importance Test Results	60
3.3.3	Model Robustness Test Results	62
3.3.4	Features Robustness Test Results	64
3.4	Discussions	65
3.5	Conclusion	67
4	Detecting Mutation	69
4.1	Proposed Abnormal Traffic Detection	69
4.1.1	Attack Model	70
4.1.2	Deep Learning Model	71
4.1.3	Proposed Model Workflow	72
4.1.4	Data Representation	73
4.2	Implementation	73
4.2.1	Data Collection and Preprocessing	74
4.2.2	Model Implementation and Training	75
4.2.3	Evaluation and Results	75
4.3	Discussions	77
4.4	Conclusion	78

5	Detecting Morphing	80
5.1	Proposed Morphing Detection Solution	80
5.1.1	Attacker-Defender Model	81
5.1.2	Generative Deep Learning Model	83
5.1.3	Data Representation	85
5.2	Experimental Analysis	86
5.2.1	Data Collection	86
5.2.2	Model Implementation	88
5.2.3	Testing Results	88
5.3	Discussions	89
5.4	Conclusion	90
6	Real-time Traffic Classifier	91
6.1	Data Visualization	91
6.2	Confidence Measure	92
6.3	Ensemble Classifier Model	96
6.4	Experimentation Results	97
6.4.1	Confidence Measure Results	97
6.4.2	Ensemble Classifier Results	97
6.4.3	Optimal n Results	97
6.5	Formal Description and Discussions	106
6.5.1	Problem Description	106
6.5.2	Proposed Solution	106
6.6	Conclusion	110
7	Conclusion	111

List of Figures

1.1	Proposed classification framework	3
2.1	Internet traffic classification process	8
2.2	Data representation method	13
2.3	ML method approaches	17
2.4	Trends of supervised ML methods used for traffic classification . .	18
2.5	Trends of unsupervised ML methods used for traffic classification	22
2.6	Trends of unsupervised ML methods used for traffic classification	27
2.7	Internet Traffic classification objectives	28
2.8	Internet traffic obfuscation techniques	36
3.1	Hierarchical Internet Traffic Classification	47
3.2	Data Collection Setup	48
3.3	Data Representation: (a) Gray images and (b) RGBA images . .	51
3.4	Data visualization (at left: RGBA method, and at right: Gray method)	52
3.5	The used CNN architecture.	54
3.6	Performance test results (Level 1)	60
3.7	Performance test results in terms of accuracy: (a) for Level 2 and (b) for Level 3	60
3.8	Performance test results in terms of accuracy (Level 4)	61
3.9	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.	62
3.10	RGBA method features importance results in terms of accuracy .	63
3.11	Robustness test results in terms of accuracy (Level 1)	63
3.12	Robustness test results in terms of accuracy (Level 2)	64
3.13	Anonymization results in terms of accuracy considering the TOR and VPN data-sets	65
3.14	Features robustness test results in terms of accuracy for all levels considering the RGBA28x28, RGBA4x4, RF28x28, and RF4x4: a) level 1, b) level 2, c) level 3, and d) level 4.	66

4.1	Attack model	70
4.2	Proposed model	72
4.3	Proposed scheme workflow	73
4.4	Data Representation	74
4.5	Visualized images representing network traffic	76
5.1	Attacker-Defender Model	81
5.2	Generative Adversarial Deep Learning Model	82
5.3	4x4 obtained images	86
5.4	8x8 obtained images	86
5.5	28x28 obtained images	87
6.1	t-SNE visualization of Level 1 data for different values of n: (a) n = 2, (b) n = 4, (c) n = 6, (d) n = 8, (e) n = 10, (f) n = 12, (g) n = 14, (h) n = 16, (i) n = 18, (j) n = 20, (k) n = 22, (l) n = 24.	93
6.2	Image Representation	94
6.3	Classifier Model	96
6.4	Confidence Measure Results	99

List of Tables

2.1	Previous survey summary	8
2.2	Internet traffic data providers	9
2.3	Comparison of ML methods used for Internet traffic classification	26
2.4	ML based Internet traffic classification objectives	29
2.5	Traffic obfuscation techniques	34
2.6	Obfuscation techniques	42
3.1	Collected Data (Number of Flows)	50
3.2	Feature Set for statistical classification	53
3.3	Packet fields	56
3.4	Accuracy Results (in%) with the TOR and VPN data-sets	65
4.1	Testing evaluation results	77
4.2	Classification Results	78
5.1	Summary of notations	81
5.2	Classification Results	87
5.3	Morphing Loss	87
5.4	Detection Rate	87
6.1	Ensemble Accuracy Results	98
6.2	Best Choice of n	101
6.3	Effect of choosing optimal n based on the MI measure	102
6.4	Effect of choosing optimal n based on the training accuracy	103
6.5	Effect of choosing optimal n based on the proposed confidence measure	104
6.6	Enhancement in terms of time	105
6.7	Voice Results	106

Chapter 1

Introduction

From the first application of the Internet (i.e., electronic mail in 1972 [1]) to present day, Internet traffic has undergone a tremendous evolution. With the emergence of the Internet of Things (IoT), there is a shift in the types of connected devices and supported applications. Future large-scale networks present increased management complexity in terms of Quality of Service (QoS) and security [2]. The goal of Internet traffic classification is to facilitate network management. Traffic classification has a key role in QoS provisioning, network planning, intrusion detection, network monitoring, traffic trends analysis, and firewalling. First, packet marking was proposed to differentiate traffic based on its QoS class. Examples of packet fields used for packet marking are Type of Service (ToS), Differentiated Services Code Point (DSCP), and Explicit Congestion Notification (ECN). Indeed, several protocols have been proposed for traffic differentiation including Differentiated Services (DiffServ), Integrated Services (IntServ), and Multiprotocol Label Switching (MPLS). However, these protocols were not widely deployed and adopted due to their complexity and compatibility issues. A more practical solution is to classify the traffic without modifying the Transmission Control Protocol/Internet Protocol (TCP/IP) mechanism. In this context, Deep Packet Inspection (DPI) was proposed for classification based on the packet header and application signature. However, this method burdens the network with its high processing overhead. Furthermore, DPI cannot deal with encrypted traffic [3, 4]. Alternatively, port-based classification was proposed, given that well-known port numbers have been assigned by the Internet Assigned Numbers Authority (IANA) for certain application and protocols. Though, nowadays, the applications are using dynamic port allocation. Additionally, some network services, like tunneling and anonymization, hide the port numbers information [1, 5]. Furthermore, in the mobile domain, most of the applications traffic is tunneled through Hyper Text Transfer Protocol Secure (HTTPS) traffic. In this context, Machine Learning (ML) based methods were proposed for traffic classification and intrusion detection. Many supervised and unsupervised methods have been employed accordingly. While supervised methods are devoted to classifying the

(attack) traffic based on known class labels, the unsupervised ones allow the detection of unknown traffic. The traditional ML methods rely on well-structured and hand-designed features. These features are extracted using statistical traffic measures (e.g. maximum, minimum, standard deviation, etc. of the packet sizes, and packets inter-arrival times). However, these features are hand-crafted and the defined classes do not reflect the network needs in terms of QoS and security. In this thesis, we rely on Deep Learning to be applied on quasi-raw traffic features. In addition, we define a hierarchical classification framework to classify traffic based on its required QoS and security levels. However, recently, the security of ML based methods have been questioned. New adversarial attacks have been employed to manipulate the input data in the aim of forgery or misclassification. In the traffic classification domain, many obfuscation techniques have been proposed towards hiding or modifying the flow characteristics to confuse the classifier and prevent any detection. From these obfuscation techniques, two well-known techniques are considered in this thesis: mutation and morphing. Even though in many cases, the recovery of the original traffic is impossible, the detection of manipulated traffic is essential to prevent misclassification. In this context, adversarial DL networks have been proposed to generate and detect adversarial attacks. In this thesis, the combination of denoising autoencoder and Generative Adversarial Network (GAN)'s discriminator are used to detect mutated traffic and denoise it. Moreover, the combination between variational autoencoder and GAN's discriminator are employed for generating morphed traffic and detecting it. Detecting mutation and morphing is not enough to avoid misclassification. In our case, the choice of n affects the classification accuracy. Choosing large value of n does not always guarantee better accuracy in addition to real time application constraints. Consequently, the choice of n should meet the compromise between real-time classification constraints and classification performance. Based on that, we propose a model confidence measure that reflects the confidence in the model to give good testing accuracy. Moreover, giving the time series type of traffic, we propose an ensemble method of the successive classifiers to enhance the individual classification results.

As shown in Figure 1.1, our proposed classification framework consists of several modules:

- **The first module** aims at extracting quasi-raw statistical features from the first $n \times n$ packets of a flow. At this stage, the flow is transformed into an image to be passed to a CNN classifier.
- **The second module** aims at detecting mutated traffic. While statistical features can be used for mutation detection, our data representation has the benefit of mutated traffic recovery by applying image denoising techniques.
- **The third module** has the role of morphing detection. In this case, statistical features are useless for detection. However, our data representation

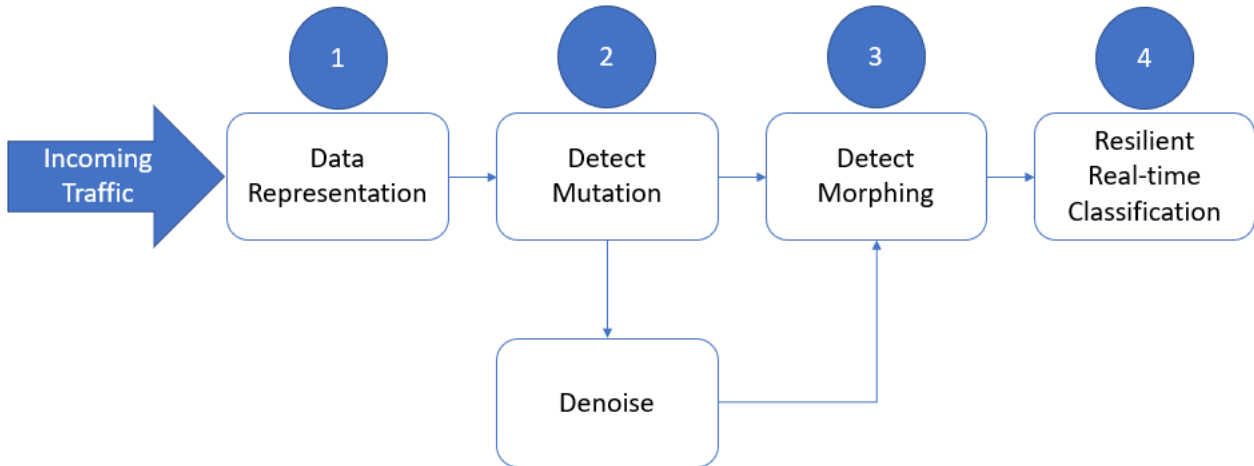


Figure 1.1: Proposed classification framework

can be used for detection. While traditional morphing techniques rely on analysing the probability distributions of packets sizes and inter-arrival time in a complex manner, GAN can be naturally used to infer the probability distributions of these features by applying VAE with our data representation.

- Finally, **the fourth module** aims at building a confidence in the model considering the traffic as time series data to optimize the choice of n meeting the compromise between the accuracy, training overhead, and testing response time. In this case, a confidence measure is proposed based on the training accuracy and the mutual information between the packets features and the class vector. Moreover, to enhance the accuracy at the chosen n , we propose an ensemble method of the successive classifiers by taking the average of the classes probabilities to get the ensemble decision.

1.1 Contributions

The contributions of this thesis can be summarized as follows:

1. Proposing a new data representation method to transform traffic flows into images and apply CNN for classification.
2. Conducting a comparative study between raw and quasi-raw representation for DL-based traffic classification.
3. Proposing a DL-based architecture for defending traffic mutation, by recovering and detecting mutated traffic.

4. Defending traffic morphing by employing generative DL-based model to generate and detect fake traffic.
5. Proposing a new confidence measure based on the training accuracy and the mutual information.
6. Optimizing the number of packets sufficiently required for accurate classification based on a proposed confidence measure.
7. Proposing an ensemble based classification method by aggregating successive classification results for successive values of n to enhance the classification accuracy.

1.2 List of Publications

My thesis work has resulted in a number of publications as listed:

1. Hussein, O. Salman, S. Abdallah, I. H. Elhajj, A. Chehab, A. Kayssi, SDN & Edge Computing: Key Enablers towards the 5G Evolution, Institution of Engineering and Technology (IET), Titled: Access, Fronthaul and Backhaul Networks for 5G & Beyond, chapter 19, London, UK, 2017.
2. O. Salman, I. H. Elhajj, A. Chehab and A. Kayssi, A machine learning based framework for IoT device identification and abnormal traffic detection, Trans Emerging Tel Tech. 2019; e3743.
3. O. Salman, I. H. Elhajj, A. Chehab and A. Kayssi, IoT survey: An SDN and fog computing perspective, Computer Networks, 2018, 143, 221-246.
4. O. Salman, I. Elhajj, A. Kayssi and A. Chehab, "An architecture for the Internet of Things with decentralized data and centralized control," 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, 2015, pp. 1-8.
5. O. Salman, I. Elhajj, A. Kayssi and A. Chehab, "Edge computing enabling the Internet of Things," 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, 2015, pp. 603-608.
6. O. Salman, I. H. Elhajj, A. Kayssi and A. Chehab, "SDN controllers: A comparative study," 2016 18th Mediterranean Electrotechnical Conference (MELECON), Lemosos, 2016, pp. 1-6.
7. O. Salman, S. Abdallah, I. H. Elhajj, A. Chehab and A. Kayssi, "Identity-based authentication scheme for the Internet of Things," 2016 IEEE Symposium on Computers and Communication (ISCC), Messina, 2016, pp. 1109-1111.

8. O. Salman, R. Morcel, O. Al Zoubi, I. Elhajj, A. Kayssi and A. Chehab, "Analysis of topology-based routing protocols for VANETs in different environments," 2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), Beirut, 2016, pp. 27-31.
9. O. Salman, M. Awad, F. Saab, I. Elhajj, A. Chehab and A. Kayssi, "A game-theoretic approach to resource allocation in the cloud," 2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET), Beirut, 2016, pp. 132-137.
10. O. Salman, I. H. Elhajj, A. Kayssi and A. Chehab, "Software Defined IoT Security Framework," 2017 The Fourth International Conference on Software Defined Systems (SDS-2017), Valencia, May 8-11, 2017.
11. O. Salman, A. Kayssi, A. Chehab, and I. H. Elhajj, "Multi-Level Security for the 5G/IoT Ubiquitous Network," 2017 The 2nd International Conference on Fog and Mobile Edge Computing (FMEC 2017), Valencia, May 8-11, 2017.
12. O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, "QoS Guarantee over Hybrid SDN/non-SDN Networks," The 8th International Conference on Network of the Future (NoF2017), London, Nov. 22-24, 2017.
13. O. Salman, L. Chaddad, I. H. Elhajj, A. Chehab and A. Kayssi, "Pushing intelligence to the network edge," 2018 Fifth International Conference on Software Defined Systems (SDS), Barcelona, 2018, pp. 87-92.
14. O. Salman, I. H. Elhajj, A. Chehab and A. Kayssi, "A Multi-level Internet Traffic Classifier Using Deep Learning," 2018 9th International Conference on the Network of the Future (NOF), Poznan, 2018, pp. 68-75.
15. A. Hussein, O. Salman, A. Chehab, I. H. Elhajj and A. Kayssi, Machine Learning for Network Resiliency and Consistency, 2019 Sixth International Conference on Software Defined Systems (SDS), Rome, 2019.
16. O. Salman, I. H. Elhajj, A. Chehab and A. Kayssi, " Denoising Adversarial Autoencoder for Obfuscated Traffic Detection and Recovery," 2019 2nd International Conference on the 2nd IFIP International Conference on Machine Learning for Networking (MLN'2019), Paris, 3-5 December 2019.

1.3 Organization

This thesis consists of 5 chapters, besides the introduction chapter. These chapters are organized as follows:

- Chapter 2 presents a comprehensive literature review on the traffic classification and obfuscation techniques. In this chapter, we review the different methods, goals and proposed classification methods. In addition, we present the traffic obfuscation techniques that can affect the accuracy of the classification methods. In addition, in this chapter, we include the main concepts needed for understanding the rest of the chapters.
- In chapter 3, we present our proposed data representation methods. In addition, we conducted a comparative study between our method and previous work.
- Chapter 4 presents our proposed method to detect traffic mutation considering several mutation techniques.
- In Chapter 5, we consider generative adversarial DL-based architecture to generate and detect morphed traffic.
- Chapter 6 presents the proposed confidence measure to define the sufficient number of packets $n \times n$ for achieving good accuracy while minimizing the response time. Moreover, in this chapter, we present our ensemble classifier model that aims at enhancing the accuracy of individual classifiers at different values of n .
- Finally, we conclude in chapter 7, opening the floor for future research perspectives.

Chapter 2

Background and Related Work

Classification using Machine Learning (ML) methods have been proposed to overcome DPI and port-based limitations [6, 7, 8]. ML methods have shown their effectiveness to classify even encrypted traffic [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33].

In the literature, several works compare different ML classification methods. However, some of these publications do not consider new ML methods used for traffic classification (e.g., deep learning, DL) or new traffic classification approaches (e.g., action-based classification). Table 2.1 summarizes previous surveys on Internet traffic classification. The work done in [34, 35, 36] does not include new ML methods for traffic classification or new Internet applications and traffic types. In [37], the authors only consider payload-based classification. In [15], the methods proposed for encrypted traffic classification are reviewed. The most recent work surveying ML-based traffic classification methods is in [38]. However, none of the existing surveys consider the main traffic classification limitation represented by traffic obfuscation techniques.

In this chapter, we review ML-based Internet traffic classification including the applied ML methods, available public data, different data representations (i.e., feature extraction), and classification objectives. Importantly, we also describe the obfuscation methods proposed to thwart the Internet traffic classification.

The ML-based classification process consists mainly of four phases, as illustrated in Figure 2.1: data collection, data preprocessing and features extraction (data representation), model training, and performance evaluation based on the classification objectives.

2.1 Internet Traffic Data Challenges

ML learning consists of two phases: training and testing (or validation). In the training phase, the classifier is fed with part of the collected data with the aim of recognizing differentiating patterns. Cross-validation is used during this phase,

Table 2.1: Previous survey summary

Reference	Year	Summary
[36]	2008	Review of the different ML methods used for traffic classification
[35]	2009	Review of the different ML methods used for traffic classification and review of the different techniques used
[34]	2012	Review of the main challenges and future research directions
[37]	2014	Review of the payload-based traffic classification
[15]	2014	Review of the techniques used to classify encrypted traffic
[38]	2018	A systematic review of ML-based traffic classification with its different stages: data collection, features extraction, ML method selection, and model evaluation

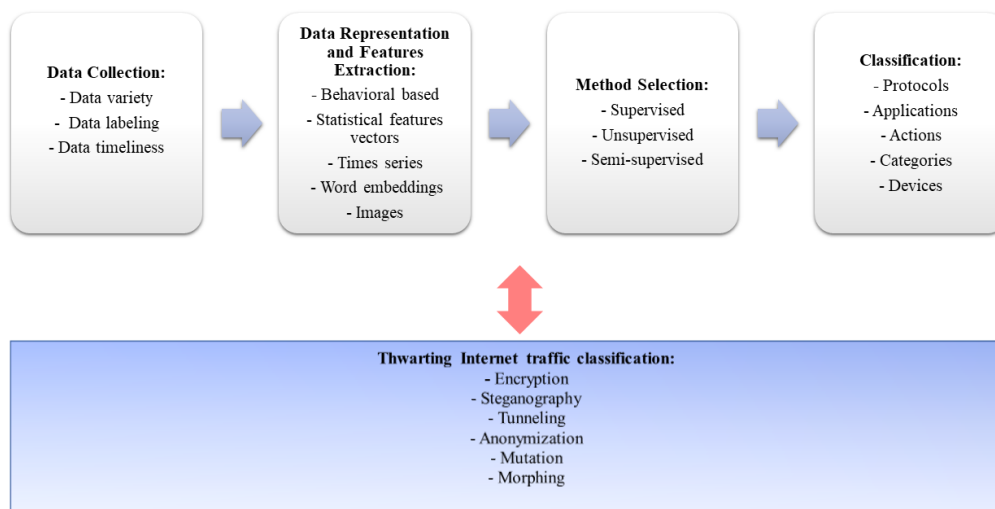


Figure 2.1: Internet traffic classification process

and model performance is evaluated at each fold to optimize the model parameters. In the testing phase, the model is tested on unseen data. Therefore, if the model performance is not as good as expected, re-training the model with a different set of parameters might enhance its performance. However, sometimes model performance remains unsatisfactory, even after a grid search over all the parameter values. In this case, there is a need to re-collect additional representative data. Indeed, data are the essential part of any ML classification problem.

Table 2.2: Internet traffic data providers

Source	Dataset	Year	Type	Used In	Description
Canadian Institute for Cybersecurity (CIC)	ISCX VPN-non VPN [39] ISCX Tor-nonTor [40]	2016	Labeled	[31, 41, 42] (3 papers)	These datasets consist of 7 categories: Web Browsing, Email, Chat, Streaming, File Transfer, Voice over IP (VoIP), and Peer to Peer (P2P), collected by researchers at the University of New Brunswick (UNB)
University of Cambridge Computer Laboratory Research Group	Moore Dataset [43]	2003	Labeled	[1, 22, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58] (18 papers)	This data was captured at the University of Cambridge campus, where 1000 users are connected to the Internet by a gigabit Ethernet link. It consists of 7 classes: attack, web, bulk, mail, service, P2P, and database
Research group at the University of Waikato Computer Science Department (WAND)	Auckland (I, II, III, and IV) [59] University of Auckland	I (July 1999), II (November 1999- June 2000), III (August 2000), IV (February 2001 - April 2001)	Unlabeled	[60, 61, 62, 63, 64, 65] (6 papers)	This research group has set up a bi-directional measurement system on the OC3c ATM access link that connects the University of Auckland to the Internet
Measurement & Operations Analysis Team, (NLANR)			Unlabeled	[22, 66, 67, 68, 69, 70] (6 papers)	NLANR is an NSF-funded research group at the San Diego Supercomputer Center
Center for Applied Internet Data Analysis (CAIDA)			Unlabeled	[71, 72, 73, 74] (4 papers)	Founded in 1997, CAIDA conducts network research and builds research infrastructure to support large-scale data collection
MAWI Working Group Traffic Archive	WIDE [75]	1999-2019	Unlabeled	[12, 22, 74, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85] (14 papers)	WIDE is an updated dataset collected at the US-Japan Trans-Pacific backbone link. It consists of several traces that range in time from 1999-2019

In [86], the authors conducted a validation analysis of training and testing Internet traffic datasets. Their aim was to show how model performance is affected when the testing and training datasets are collected at different geographical locations. In the following, we discuss different issues encountered when classifying Internet traffic data using ML methods.

2.1.1 Data Size

Imbalanced data are one of the main issues encountered by most ML methods. Most specifically, in the network domain, some protocols/applications traffic dominate over other types [87]. Data balancing is proposed to overcome biasing encountered when training a model with imbalanced data [88]. In this context, resampling techniques have been employed to balance the data by under- or over-sampling, which consist of augmenting the size of the minority classes or decreasing the size of the majority classes, respectively. Compared to the original data size, under-sampling produces a smaller data size and over-sampling results in a

larger data size. However, under-sampling is barely used, given that: 1) there is no large labeled Internet traffic, 2) and the larger the data, the better is the model performance. Instead, the over-sampling technique, which consists of randomly replicating the samples of the class presenting a smaller number of instances, is widely adopted. However, data balancing might affect the accuracy of the trained models when applied to real data. Another issue encountered, at the data collection phase, is *packet sampling*. Packet sampling consists of capturing the packets at fixed intervals [89]. In this case, not all the flow packets are captured due to the large data size and real-time processing limitations when dealing with the Internet backbone traffic. Consequently, this might affect the classification performance given that the statistical features extracted from the full packets flows are different from those extracted from sampled packets flows. In this case, it is important to consider sampling at the training and testing phases.

2.1.2 Data Labeling

Data labeling is another issue in the Internet domain, where controlling the applications traffic is not a straightforward task. Consequently, the collection of labeled Internet traffic requires specific setup in a private network (e.g., lab), to capture the traffic and filter it based on specific parameters (e.g., time interval, IP addresses, Media Access Control (MAC) addresses, and port numbers). Consequently, the burden of collecting and labeling Internet traffic pushes the researchers to find a way to label the widely available unlabeled datasets (e.g., CAIDA datasets). Two methods are mainly employed to label Internet traffic: DPI and port-based classification. Many DPI tools have been developed for this aim, such as: PACE [90], OpenDPI [91], L7-filter [92], nDPI [93], Libprotoident [94], and NBAR [88]. These tools have been used to validate ML classification results and label unlabeled Internet traffic [95, 96, 97]; however, they cannot be applied to label encrypted traffic. Similarly, port-based classification tools such as CoralReef [98] have been also developed and used to label the unlabeled Internet traffic datasets. These approaches have also been combined with the statistical based ML approaches to ensure better performance [4, 99]. However, none of these labeling approaches ensure 100% correct labels, so the ground truth will not be highly accurate. In this case, collecting labeled data remains necessary.

2.1.3 Data Timeliness

As shown in Table 2.2, the Moore dataset was extensively used in the reviewed papers. This dataset consists of preprocessed data with 249 extracted features and was captured in 2003. Since then, Internet traffic has changed noticeably with the emergence of IoT. *New traffic trends* have emerged with new applications, new kinds of traffic, and new types of connected devices. According to the last

Cisco VNI forecast [100], video traffic will dominate the Internet traffic by 2021. In this context, new datasets, including new applications traffic, are needed. In fact, the traffic characteristics depend on the device specifications (OS, memory capacity, processing power, and supported protocols). Consequently, considering traffic generated by new kinds of devices (IoT sensors, robots, drones, smart cars, etc.) is key to be able to identify new traffic types (e.g., sensing and actuation traffic) along with their generating devices. To summarize, collecting representative data is key for ML classification. In this context, data variety, accurate data labeling, and updated classes definition are essential to ensure ML classification effectiveness. After collecting data, a very important phase comes up, which is features extraction or data representation. In the next section, we review the different data representation methods proposed in the traffic classification domain.

2.2 Data Representation

Data representation is an essential part of any ML classification process. To recognize the discriminating patterns, the ML model needs to be based on well-structured input. Traditional ML methods require hand-crafted features. Indeed, after collecting data, a preprocessing phase follows to extract the features that will be inputted to the ML model. Typically, traffic classification is assessed per network flow, where a flow is defined as the ensemble of packets having the same connection parameters, including the source and destination IP addresses, the source and destination port numbers, and the transport protocol. Note that a bidirectional flow is defined as the set of packets having the common 5-tuples connection parameters in either direction. Two main flow features, with their various statistical values (max, min, standard deviation, quartile, etc.), have been considered extensively in the literature: the packet interarrival time and packet size [28, 101, 102, 103, 104, 105]. Based on the reviewed work, we can categorize the data representation methods employed in the traffic classification domain, as follows: behavioral-based, vectors, time series, word embeddings, and images.

2.2.1 Behavioral-based

In this category, traffic is the intercommunication patterns between the flow source and destination [71, 106, 107, 108, 109, 110, 111, 112, 113, 114]. Karagiannis et al. in [107] proposed a multi-level behavioral analysis system, called BLINC, based on the interaction between hosts, protocol usage, and per-flow average packet size. The BLINC system represents the flow data as graphlets that describe the normal usage patterns of network applications. These structures are then used in conjunction with the host information to predict the application/protocol name. A graphlet, as shown in Figure 2.2-a, consists of the

communication patterns defined by the 4-tuple: source IP address, destination IP address, source port number, and destination port number. Even though port numbers are used, the classification is independent of their exact values what matters is the associated pattern.

More recently, a method to represent the users behavior and analyze the dynamic characteristics of the host behaviors was proposed [108]. Hu et al. extracted the features that correlate with the target application from the traffic traces [115]. For each application, they built a profile (ensemble of rules) that defines the application behavior. Cao et al. presented an approach that combines both host- and flow-level identifications to classify applications traffic by statistical behavioral analysis [109]. In [116], the user-app bipartite method is proposed. User-app bipartite, as shown in Figure 2.2-b, aims at illustrating the communication between mobile users and application servers. Two types of nodes are defined: the user and server nodes. The user node is characterized by the phone number client section (i.e., the first three digits of the phone number), mobile phone brand, and phone model. The server node is characterized by the supported application categories and sub-categories. The edge is characterized by the flow start and end times, packet size in bytes, direction (upstream, and downstream), and application category label (determined by the server node). In Figure 2.2-b, the color refers to the application category label. Because the user might use different applications, one user node may have more than one colored edge at the same time. The server node might have multiple colored edges at the same time because a server might host different applications. In [77], graphlets (macroscopic traffic characteristics) and packet size-based features (microscopic traffic characteristics) are employed. However, the behavioral based flow representation presents memory and processing overhead.

2.2.2 Vectors

A typical method to represent Internet traffic is by extracting flow-based statistical features. These features are calculated by extracting packet-related information, as shown in Figure 2.2-c. Moore presented a comprehensive list consisting of a total of 249 statistical features computed per network flow [117]. This comprehensive set includes different statistics (max, min, standard deviation, mean, etc.) of the different flow packets attributes: packets interarrival time, packet size, burst/flow volume, in addition to the IP and port numbers and TCP flags. These statistics are computed for in and out flow directions. Another vector-based representation is the discrete byte encoding proposed in [118]. In this method, the first n -bytes of a flow are encoded by a feature vector v with 256 elements. The components of v are initialized to 0. Then for each byte in the input stream, the $(i*256 + c(i))$ i th component is set to 1, where i is the position of a byte with value $c[i]$ in the extracted flow. The resulted binary vector v is input to the ML classifier. This data representation type is widely adopted because it is suitable

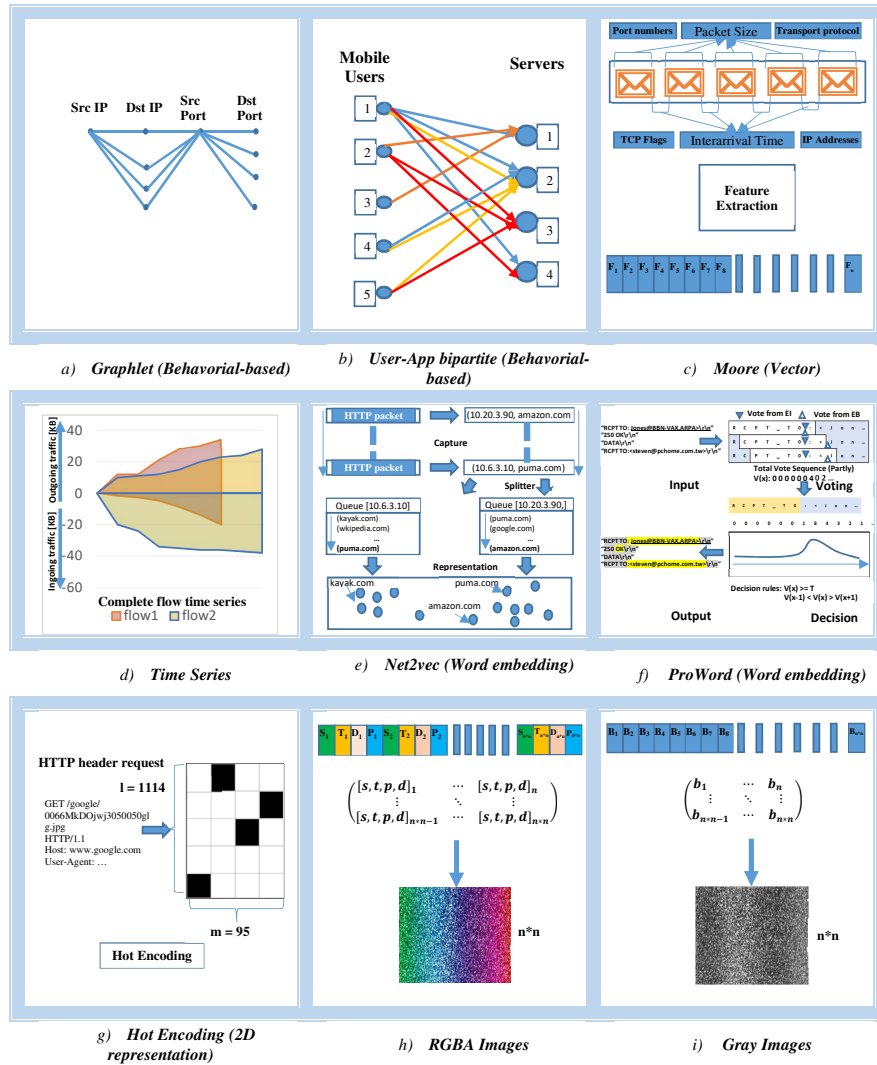


Figure 2.2: Data representation method

for most existing ML methods.

2.2.3 Time series

Another way to represent Internet traffic is through time series. Conti et al. represented network flows by generating time series of the communicated packets/bytes [119]. Three time series are generated for each flow: 1) for the bytes transmitted by incoming packets, 2) for the bytes transmitted by outgoing packets, and 3) for the bytes transmitted by both incoming and outgoing packets. Additionally, a time series for the packet interarrival time is considered. The obtained shapes, as illustrated in Figure 2.2-d, are used for classifying user actions. Acar et al. proposed another time series-based traffic representation by considering the traffic data rate [120]. In fact, the following statistics are computed for the obtained time series: absolute energy, length, mean, median, skewness, entropy, standard deviation, variance, continuous wavelet transform coefficients, Fast Fourier Transform coefficients, and coefficients of polynomial fitted to time series. These computed statistics are then passed to the classifier. This data representation is usually applied for Markov Model-based methods.

2.2.4 Word embeddings

Because HTTP protocol requests contain valuable information about the destination host, and inspired by the text recognition applications, word embeddings have been applied to represent Internet traffic. In this context, FLOWR was proposed in [24, 121] for extracting the app signature from the HTTP header. The proposed method relies on key-values exchanged in HTTP queries (e.g., hostname). By training the model by pre-computed key-value pairs for a known application, the model can induce the key-values of new applications by means of regression. Another method relying on grouping the HTTP User-Agent fields is proposed in [122]. The common strings for certain applications are extracted using the LCS algorithm. Then the classification of unknown applications can be performed using the extracted header signature. By employing the Word2vec unsupervised learning techniques, used mainly to generate word embeddings from a word-based dataset, the work in [123] considers the domain names in the Domain Name Server (DNS) requests to classify Internet traffic flows associated with these DNS queries. The results reveal a relationship between the domain names and the type of traffic directed towards them. In a similar way, Net2vec is proposed in [124] to classify the HTTP(s) requests. The captured HTTP(s) requests are transformed into tuples of the form (IP address, hostname) representing the requested server. The aim is to assign the user to certain product categories as the case in web-based recommending systems which are based on the websites visited by the user. However, net2vec is run on the network level. A queue of the visited domain servers is constructed for each user by the splitter, and then this user is associated with a certain user category (e.g., Education, Sports, and Health & Fitness). In fact, each domain name is represented by a numerical data

point to be passed to the classifier.

Goo et al. proposed a signature extraction method consisting of three levels [125]. The first level consists of extracting common substrings from the traffic payload forming the flow signature. Similarly, at level two, common substrings from the packet content are extracted to form the content signature. At level three, the packet signature is extracted.

Bag-of-Words (BoW) is a popular technique used in Natural-Language Processing (NLP) for representing words as numeric vectors. In this context, a BoW-based traffic representation is proposed in [126]. Numeric vectors are created for representing the web pages Uniform Resource Locators (URLs) visited by the user in one session. For URLs, special characters (i.e., :, /, ?, etc.) are treated as a space between words; therefore, a URL is represented as a sequence of words from the dictionary.

ProWord is proposed for extracting a set of words from the application request to identify the type of traffic flow in [127]. For extracting words from text, boundaries are identified through segmentation, a well-known statistical method used in NLP. Specifically, the Voting Experts (VE) algorithm, is used to identify possible word boundaries using entropy. ProWord uses a ranking algorithm that maps different dimensions of protocol feature heuristics (e.g., frequency, occurrence location, and length). ProWord consists of four phases: input processing, segmentation voting, decision making, and output consisting of the key strings of the protocol request header. The different obtained strings will be inputted to a neural network classifier to extract the representative protocol features. Moreover, the identification of the mobile devices model (iPhone, Android, etc.) is performed based on TTL and specific strings in the HTTP requests [128].

2.2.5 Images

Recently, DL has shown its power in enabling many computer vision applications and especially in image recognition. Consequently, representing traffic as images was considered to apply DL for traffic classification. DL methods have the representation learning ability and do not require hand-crafted features as is the case for the other traditional ML methods. Leroux et al. proposed representing flows by 2D histograms [26]. The two considered spaces are: packet size-interarrival time and burst time-burst size. The histogram reflects the relationship between the considered features. Each histogram represents a window containing the first 1024 packets of an application session. The HTTP packet header is transformed by hot encoding into $m \times l$ vectors, where $m = 95$ and $l = 1114$ [129]. Chen et al. apply the reproducing kernel Hilbert space embedding to represent the flow statistical features into 6 channels images [130]. Each feature is represented by its static (marginal probability) and dynamic (conditional probability) information. After applying convolution, a 28-feature vector representing the raw flow features is passed to the fully connected layer. Representing the traffic raw data

as grayscale images was proposed in [21, 31, 42, 131, 132, 133, 134]. This method consists of transforming the first $n \times n$ bytes of each flow or session into a grayscale image of size $n \times n$, as illustrated in Figure 2.2-e [31]. As shown in this figure, the first $n \times n$ bytes ($[B_1, B_2, \dots, B_n]$) form an $n \times n$ matrix. In [135], Salman et al. proposed extracting four features (packet size (s), interarrival time (T), direction (D), transport protocol(P)) for the first $n \times n$ packets of a traffic flow. Then, each flow is represented by an RGBA image, as illustrated in Figure 2.2-f, to apply a Convolutional Neural Network (CNN). In this case, each packets features represent the RGBA components of a pixel in the obtained image.

A comparison of different data representations applied to DL for mobile traffic classification was performed in [136]. Three sets of features are considered for comparison: 1) the first N bytes of the payload, 2) the first N bytes of a bi-flow, 3) and a set of features extracted from the first 20 packets (20*6 features). In addition, a statistical set of 40 features including port number information is considered to compare with a baseline method employing Random Forest (RF). The results show that DL outperforms RF for classifying different apps. However, when it comes to overlapping apps, using classification port-based information in the RF-based method leads to better accuracy and precision. In [137], another comparison was conducted between different data representation methods: 1) basic features [138, 139], 2) Moore features [117], 3) graph features [140], 4) joint features [105], 5) and service features [32]. The results show that the proposed hybrid features set, which is a combination of basic feature sets with in-flow behavior, distribution, and packet header features, presents the best results. After extracting and selecting the features that help differentiate between classes, selecting the ML method is an important stage in the ML classification process. In the next section, we review the different ML methods used in the traffic classification domain.

2.3 Machine Learning Methods

In this section, we review the different types of ML algorithms used for traffic classification. A classification problem is a decision-making problem, in which the aim is to decide if data corresponds to a specific class given different hypotheses. Consequently, the classification problem can be formulized as an optimization problem, in which the aim is to choose a class label (hypothesis) that minimizes a certain cost function. In this context, the classification methods can be categorized into two types: parametric and non-parametric. For the parametric category, the aim is to define a function f , such that $Y = f(X)$, where Y is the output (label) and X is the features vector. Examples of parametric classification methods are Nave Bayes, linear regression, etc. However, non-parametric methods aim to minimize the classification error without inferring the mapping between the input and output. Furthermore, the ML methods are based on the

training approach, as illustrated in Figure 2.3. In the following, we review the ML methods used for traffic classification, including supervised, unsupervised, and semi-supervised. Note that a new training method is emerging in the ML domain, which is reinforcement learning. However, based on the reviewed work, reinforcement learning has not been applied in the traffic classification domain, given the lack of the reward notion, which is an essential part of this type of learning. However, in the future, the application of reinforcement learning might be employed based on an innovative business model including a feedback about the user satisfaction or Quality of Experience (QoE).

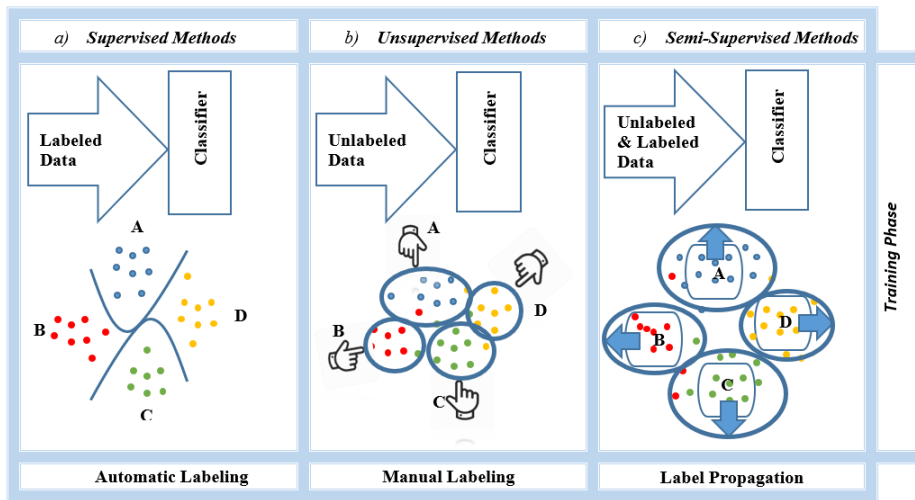


Figure 2.3: ML method approaches

2.3.1 Supervised

Supervised ML methods rely on labeled data, as illustrated in Figure 2.3-a. During the training phase, the classifier infers the rules (e.g., decision tree) or the model parameters (e.g., support vector machine, SVM) to optimize the cost function. After training the model, at the testing phase, the classifier automatically assigns the testing data to one of the learned classes. Supervised learning is usually preceded by a features selection/reduction phase. This phase aims at selecting the relevant features to the classification by computing the mutual information or covariance between the features and class labels. Different supervised methods have been considered in the literature to classify Internet traffic. Figure 2.4 illustrates the evolution of the supervised methods used in the traffic classification domain. Bayesian-based methods have been constantly considered over the years. Moreover, the graph shows that DL was recently applied (in 2017 and 2018) for traffic classification.

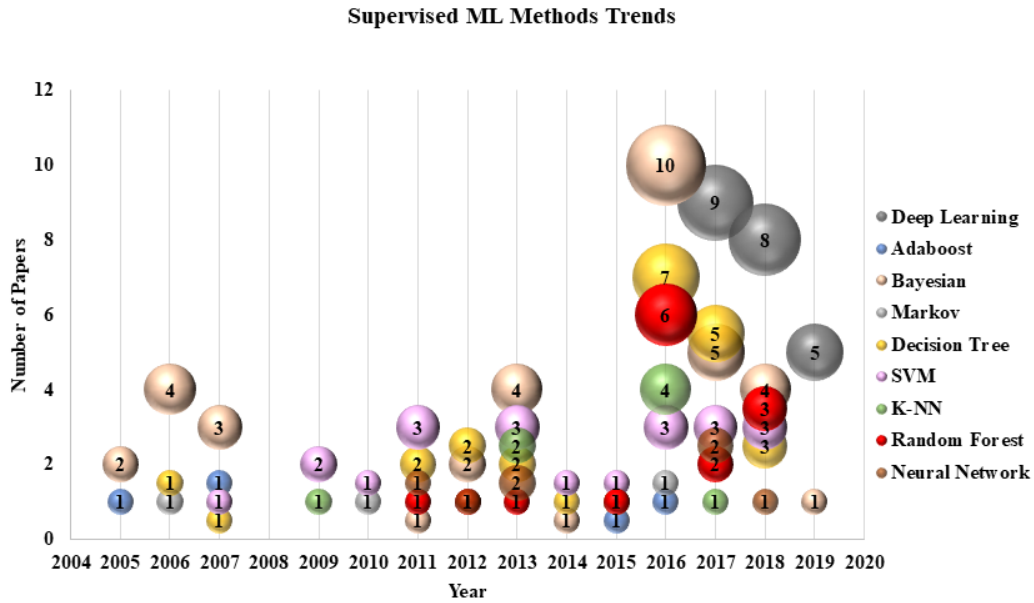


Figure 2.4: Trends of supervised ML methods used for traffic classification

Decision Tree

The **Decision Tree (DT)** classification method is a rule-based method. It mainly consists of answering a sequence of questions at the tree edges towards a leaf representing the predicted label. DT-based methods are fast and can be applied for large datasets. However, they are prone to overfitting. DT-based methods have been considered to classify traffic [141, 142]. Given a set of flow features, to build a decision tree, the feature of high differentiation power is chosen to be at the root of this tree. The feature importance can be measured by different means such as Gini index, entropy, mutual information, etc. The tree depth defines the maximum number of branches to reach a leaf. Limiting the maximum depth of the DT helps minimize the overfitting issue. C4.5, an algorithm used for generating DTs, is one of the best performing methods used in traffic classification domain. The DT-based methods give the best results in 11 out of 17 comparisons.

Bayesian

Nave Bayes (NB) is another ML method that was used for traffic classification [44]. Nave Bayes is a probabilistic method that relies on the Bayes theorem stating that: $P(Y/X)=(P(X/Y)P(Y))/p(X)$, where Y is the output (class label), and X is the input (features vector). Nave Bayes is the simplest method in the Bayesian methods family. Other Bayesian-based methods exist to model more complex situations, where there is dependence between the different features, the

output and input, along with different constraints related to their probability density distributions. In this context, different Bayesian-based methods have been proposed for traffic classification [80, 81]. In addition, Bayesian methods have been combined with other ML methods such as neural networks to classify traffic [48]. Furthermore, a cascade of naive Bayes classifiers was proposed in [143] for fine-grained traffic classification.

K-Nearest Neighbor

K-Nearest Neighbor (K-NN) is a non-parametric ML method [50]. As a lazy algorithm, K-NN does not include a training phase. However, the classification time is dependent on data size. At the classification phase, the algorithm consists of measuring the distance between the testing instance with all the labeled instances. Consequently, the testing instance is assigned to the class of its K nearest neighbors. In case the k nearest neighbors do not belong to the same class, the majority rule is applied. Typically, the distance between two data instances is the (Euclidean) distance between their features vectors.

Support Vector Machine

SVM, a well-known ML method, has been widely used for traffic classification [1, 46, 47, 52, 55, 71, 79, 144, 145, 146]. SVM consists of solving for the hyperplane that optimally separates the points representing the instances from two different classes. This is the simplest case in which the feature vectors can be represented in a two-dimensional space. However, the problem becomes more challenging when there are multi-dimensional feature vectors with multiple classes. In this case, kernel-based methods are considered to build multi-SVM models. Accordingly, the high computational complexity is the main drawback. Other limitations of the SVM method are the high training time when the training data size is large and the high re-training time when new training data are added. In this context, incremented SVM has been proposed [55] to decrease the time needed to retrain a model on new data. SVM also presents many advantages, such as its high generalization performance and its applicability to different types of classification problems. However, SVM might not offer the best performance when compared with other models. In this context, K-NN was compared to SVM in [105], and the results show that K-NN achieves better results in terms of classification accuracy. Despite the effectiveness of SVM, its accuracy is very sensitive to the data scale, used features, and model parameters because it is a sparse classification technique [47].

Neural Network

Neural Network (NN) is a well-established ML method inspired by the human neural system. The primary component of a neural network is the neuron. During

the training phase, the neural network algorithm aims at finding the function $Y = f(X) = W * X$, where Y is the output, X is the input, and W is the weights matrix. The minimization of the difference between the expected output and the actual one, results in calculating the weight matrix W and thus determining the mapping function f . NN was applied by Moore et al. to classify traffic [48, 147]. In [148], an improved back propagation NN method was proposed for traffic classification.

Deep Learning

DL is a branch of ML that has been applied to different fields including computer vision, speech recognition, and natural language processing. A DL model consists of a NN with several layers, as signified by the deep in its name. DL recently emerged in the communications domain and more specifically in the traffic classification domain [25, 31, 131, 130, 132, 149, 150, 151, 152, 153, 154, 155]. In fact, there are different types of DL architectures: CNN is a type of DL architecture, designed specifically for image-based applications. From these architectures, one can mention: LeNet-5 [156], AlexNet [157], ConvNet [158], and GoogleNet [159]. **Residual Neural Network (ResNet)** is a CNN architecture with skip connections to avoid the vanishing gradients problem. Moreover, **Recurrent Neural Networks (RNNs)** are DL architectures capable of handling sequential inputs. Accordingly, different DL architectures have been proposed to classify traffic. Wang in [133] used the Stacked Auto Encoder (SAE) DL architecture to classify traffic, considering the first 1000 bytes of each flow. Wang et al. [31, 134] propose to apply CNN for traffic classification. Wang et al. employed CNN for malware detection using the Le-Net5 architecture [134]. CNN is compared to the C4.5 method. Other results show that CNN outperforms C4.5 [31]. One study proposes a scheme called Deep Packet for traffic categorization and application identification [21]. This work is similar to that done in [31]. However, the authors in [21] work at the packet level. In [160], the authors proposed a CNN model that takes time series-based features as input. A combination of CNN and Recurrent Neural Network (RNN) layers is proposed in [161] to classify traffic by considering statistical based features. A cascade forest is proposed in [150] to classify the mobile traces into different applications. The work in [135] compared different DL architectures: LeNet-5 [156], AlexNet [157], ConvNet [158], GoogleNet [159], Residual Neural Network (ResNet) [162], RNN [163], and Deep Neural Network (DNN) [156].

Hidden Markov Model

The **Markov Model** consists of a chain of states where the transition probability determines the probability of transitioning from one state to another. Each state is linked to a certain output. In the case where the states are unobservable, the

Markov model is called hidden (HMM). The HMM has been applied to traffic classification by presenting traffic flows as time series [164, 165, 166].

Ensemble Learning

Ensemble ML methods consist of training sequentially or in parallel a set of (weak) classifiers and then aggregating the results by voting or averaging. Two main methods are employed for ensemble learning: aggregation (bootstrapping) and boosting. Different ensemble methods have been proposed in the traffic classification domain [54, 82, 167].

Random Forest (RF) is an ensemble method belonging to the bootstrapping ensemble type. It reduces bias and variance by training several DTs on different data and feature subsets. RF is one of the methods that presented high performance in traffic classification [84, 168]. **Adaboost** is an ensemble method belonging to the boosting ensemble type. It consists of sequentially training a set of weak classifiers to enhance the classification accuracy by penalizing the misclassified instances. In [169], an Adaboost model is proposed to classify traffic. In addition, transfer learning was applied to alleviate the scarcity of labeled data.

A comparison of supervised learning methods: NB, Bayesian Network, RF, DT, NB Tree and, Multilayer Perceptron (MLP), is performed in [60], with RF and DT achieving the best results. Another comparison of supervised learning methods is done in [170], and the findings show that C4.5 presents the best results. NN and C4.5 are compared in [171]. The authors state that traffic classification is not a complex task that needs the application of NN; rather, DT-based methods are more suitable in this case. A comparison between the C4.5 decision tree, SVM, NB, and RF is performed in [17]. In [172], the comparison considered clean and unclean data, where unclean data includes background traffic. The results show that C4.5, BN, RF and K-NN present better performance on clean network traffic than NB. However, on unclean data, NB presents better performance than the other algorithms. In [173], a comparison of 11 methods is performed, where RF and C4.5 yield the best results.

2.3.2 Unsupervised

Unsupervised learning is a class of ML methods applied mainly to unlabeled data, as illustrated in Figure 2.3-b. In this case, the classifier extracts the patterns from the training data without being able to evaluate classification result correctness. However, these methods could detect unknown classes. Given the abundance of unlabeled Internet traffic, different unsupervised ML methods have been applied [174]. Figure 2.5 illustrates the unsupervised learning methods trends in the traffic classification domain and shows that K-Means and clustering are the most employed methods. Moreover, the graph shows that DL was recently applied (in 2017 and 2018) for unsupervised traffic classification. In the following,

we detailed the different unsupervised ML approaches employed in the traffic classification domain.

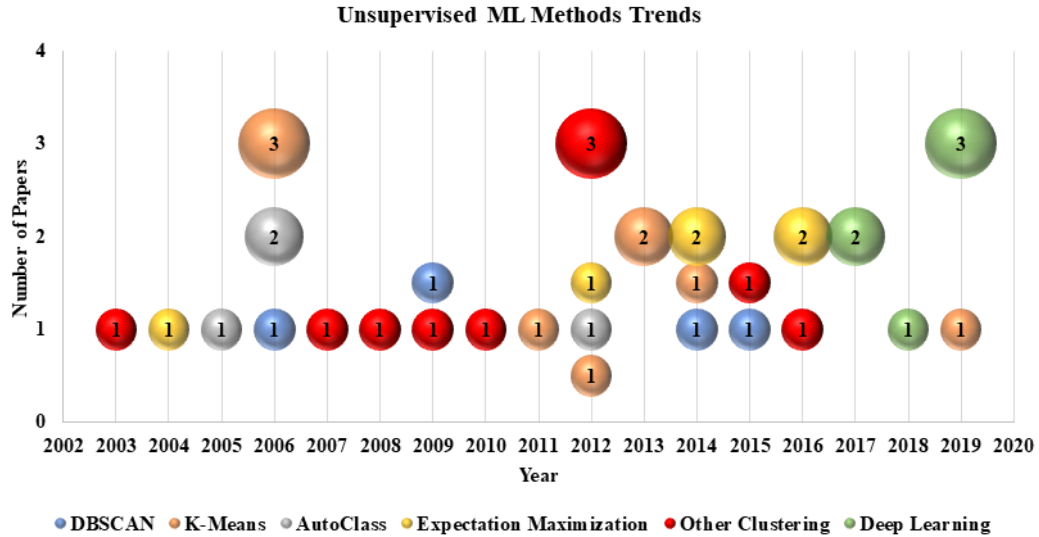


Figure 2.5: Trends of unsupervised ML methods used for traffic classification

Clustering

Hierarchical clustering is one of the unsupervised ML methods, consisting of grouping near data points into clusters. Two main approaches are adopted for clustering: bottom-up (agglomerative), and top-down (divisive). The bottom-up approach considers each instance as a cluster and then merges the nearest ones until convergence. However, the top-down approach consists of starting by one cluster for all the data instances and then dividing it until convergence [66]. In [175], an agglomerative hierarchical clustering method is proposed for traffic classification into different applications protocols. In [176], an unsupervised classification of host behaviors is proposed by considering connection patterns. Erman et al. differentiate between web and Peer-to-Peer (P2P) traffic by clustering their generated traffic [177]. Moreover, in [178], an entropy-based clustering method was proposed for traffic classification into different application types.

Density-based spatial clustering of applications with noise (DBSCAN) is another clustering technique that groups points based on their density with the ability of detecting noise, by isolating the points lying alone in low-density regions [179]. A comparison of the DBSCAN, K-means, and AutoClass (clustering) methods in [62] shows that DBSCAN can detect noise flows, while the other methods cannot. OLDBSC, a modified DBSCAN, is proposed in [180].

K-means is another unsupervised classification technique in which initially, k means are randomly created, then, the data points are clustered into k clusters

based on the points distance to these means. Afterward, the means are selected to be the centroid of the clusters, and the clustering process is repeated until convergence. This method was applied in the traffic classification domain [62, 181, 182]. An improved ***K-means*** clustering method is proposed in [183]. Given that the K-means concept is very similar to the K-NN approach, the authors in [184] proposed a classification of Internet traffic flows into QoS classes by unsupervised K-NN clustering. Similarly, RF was also applied for clustering in [185].

Deep Learning

As in the case of supervised learning, DL presents different architectures for unsupervised learning. In this context, generative algorithms such as ***Autoencoders (AE)***, are DL methods applied for unsupervised learning. As its name indicates, AE aims at encoding the training data by extracting a compressed representation. At the decoding phase, the decoder regenerates the initial image using the extracted code. The comparison between the input and output permits evaluation of the efficiency of the extracted code. Recently, this method has been applied to unsupervised traffic classification [184, 185].

- **Generative Adversarial Network:** Introduced by Goodfellow et al. in [186], GAN consists of two parts: the generator (G) and the discriminator (D). From a game theoretic perspective, GAN can be interpreted as a zero-sum or min-max game between the generator and the discriminator. The generator tries to learn the input data representation to generate data samples very similar to the real ones. The discriminator tries to maximize the probability of distinguishing between fake and real input. The GAN objective function can be presented as follows: $V(G, D) = E_{x \sim p_{data}(x)}(\log(D(x)) + E_{z \sim p_z(z)}(\log(1 - D(G(z))))$ where x is the input data, $p_{data}(x)$ is the data distribution, $D(x)$ is the discriminator output, $p_z(z)$ is the fake data distribution, z is a sample from $p_z(z)$, and $G(z)$ is the generator output. The generator aims at minimizing the probability of fake data detection by the discriminator, which means that the G objective is to find $\min_G (E_{z \sim p(z)}(\log(1 - D(G(z))))$. The discriminator aims at maximizing the probability of detecting real data as real and fake data as fake, which means that the D objective is to find $\max_D (E_{x \sim p_{data}(x)}(\log(D(x))) + E_{z \sim p(z)}(\log(1 - D(G(z))))$. Thus, the GAN objective is to find $\min_G \max_D (V(G, D))$. Primarily, GAN is applied for synthetic data generation. GAN has been applied also for image anomaly detection [187, 188, 189]. In fact, the adversarial learning permits the discriminator to detect abnormal input data. Furthermore, the generative learning permits the generator to learn the real data representation, which makes GAN suitable for image denoising [190, 191].
- **Autoencoders:** Being a generative model, AE is a type of DL networks that is specialized in extracting the input data representation. The AE consists

of two parts: an encoder and a decoder. The encoder extracts a compressed data code by estimating a function f , in such a way $z = f(x)$, where x is the input data, and z is the extracted representation or latent variable. The decoder aims at reconstructing the input data by relying on the extracted representation. In other terms, the decoder tries to infer the inverse function g , in such a way that $y = g(f(x)) = g(z)$. The AE objective function can be represented at the minimization of the difference between $g(f(x))$ and x . In other terms, the AE aims at minimizing the reconstruction error. A type of AEs is the probabilistic autoencoder, which aims to infer the distribution of x , $p_\theta(x)$ by means of another distribution $q_\phi(z/x)$ (probability of the latent variable z knowing the input x). A well-known type of probabilistic AEs is the Variational Autoencoder (VAE), which imposes a prior restriction on $p(z)$ to be a normal distribution. In this case, the problem reduces to maximizing the Evidence Lower Bound (ELBO) or maximizing the KullbackLeibler (KL) divergence between $q_\phi(z)$ and $p_{\theta(z/x)}$, represented by $KL(q_\phi(z/x)||p(z))$. Having the ability to extract the real data distribution, VAEs have been applied for anomaly detection. Indeed, when the reconstruction error is large, anomaly is detected in the input data. Borrowing the adversarial concept from GAN, Adversarial AEs (AAE) were introduced by Makhzani et al. in [192]. Similar to the GAN, AAE includes a discriminator that tries to differentiate between the data sampled from the latent variable prior $p(z)$ and the real data. In this case, the discriminator aims to minimize $L_{dis} = -1/N \sum_{i=0}^{N-1} \log(d_x(z_i)) + \sum_{j=N}^{2N} \log(1 - d_x(z_j))$, and the generator tries to minimize $L_{prior} = 1/N \sum_{i=0}^{N-1} (\log(1 - d_x(z_i)))$, where N is the number of samples, and $d_x(z_i)$ is the discriminator output of the latent space variable. In this case, if the total loss function is optimized, $q_\phi(z/x)$ will be very similar to $p(z)$, or in other terms, $KL(q_\phi(z/x)||p(z))$ will be minimized, and thus the log likelihood of the original data distribution will be maximized. AAEs were applied also for anomaly detection in images [193, 194, 195, 196]. Furthermore, a recent work has considered to add the denoising function to the AAEs for image denoising [197]. In this case, the corrupted data \tilde{x} is considered as input and two methods were proposed for model representation. The first consists of matching $\tilde{q}_\phi(z/x)$ to $p(z)$, and the second consists of matching $q_\phi(z/\tilde{x})$ to $p(z)$. However, in our work, we use a sparse AE that aims to minimize the Mean Square Error (MSE) between the reconstructed data and original data. In addition, applying the adversarial concept, we choose to train a discriminator to detect abnormal traffic when the reconstruction error is high. Thus, unlike previous work, the generator part of GAN is omitted [198].

Expectation Maximization

Expectation maximization (EM) is a statistical method that aims at finding the maximum likelihood estimate of a parameter. In the ML domain, expectation maximization can be used to induce the probability distribution of the output. Consequently, the extracted model can be applied to predict the output of any input.

EM was applied for unsupervised traffic classification in [68, 199]. **AutoClass**, an EM-based method, is applied with an unsupervised Bayesian classifier in [70]. Moreover, **Gaussian Mixture Model (GMM)** was also employed to classify traffic based on observed data probability distributions [200, 201].

In [202, 203], it was shown that supervised classification performs better than unsupervised classification of the training data; however, the unsupervised classification is more robust when applied to unseen data. In this case, a smaller decrease in accuracy is perceived compared to the supervised case. Another comparison between supervised and unsupervised methods is conducted in [16], concluding that C4.5 (supervised method), presents better performance than the Multi-Objective Genetic Algorithm (**MOGA**) (unsupervised method) for encrypted traffic, when the training and test data are collected from different networks. However, MOGA presents better performance, if the training and test data are from the same network, since the unsupervised method does not suffer from the overfitting problem.

2.3.3 Semi-Supervised

While supervised learning methods are effective when fine granularity is needed, and unsupervised methods have the ability of detecting unknown classes, the semi-supervised approach offers the best of both methods. Having a plethora of unlabeled data, semi-supervised methods were envisioned to label unlabeled traffic by means of limited labeled data. This process is often called label propagation or label induction, as illustrated in Figure 2.3-c. New applications and new types of traffic are emerging constantly. Therefore, this zero-day traffic identification requires semi-supervised ML methods [204, 205, 206, 207]. In Figure 2.6, the timeline illustrates the evolution of the proposed semi-supervised methods for traffic classification.

Erman et al. were the first to propose a semi-supervised method consisting of clustering the labeled and unlabeled data [216]. Here, the clusters are mapped to the application types based on the labeled instances contained in each cluster; however, to detect unknown traffic, the clusters that do not contain labeled instances or those that contain labeled instances of multiple classes are left unlabeled. Subsequently, Zhang et al. proposed another traffic classification method based on label propagation with **Bag-of-Flows (BoF)** [209]. The authors com-

Table 2.3: Comparison of ML methods used for Internet traffic classification

References	Year	ML Category	Compared Method	Results
[118]	2005	Supervised	NB, Adaboost, Regularized Maximum Entropy	Adaboost
[62]	2006	Unsupervised	DBSCAN, K-means, AutoClass	AutoClass
[63]	2006	Supervised, Unsupervised	NB and AutoClass	AutoClass
[67]	2006	Supervised	Bayesian Network, C4.5, NB, NB Tree	C4.5
[208]	2007	Supervised	C4.5, AdaBoost+C4.5, Logitboost, JRip, NB Tree, Bayesian Neural Network	C4.5
[17]	2011	Supervised	C4.5, SVM, NB, and RF	C4.5
[16]	2011	Supervised. Unsupervised. Semi-Supervised	Supervised: C4.5, unsupervised: K-means, semi-supervised: MOGA	MOGA (semi-supervised)
[203]	2012	Supervised, Unsupervised, Semi-Supervised	Supervised: SVM, Logistic Regression, NB, NB simple, RF, MLP, C4.5, Bayes Net; unsupervised: EM, K-Means, Cobweb hierarchic clustering, Shared Nearest Neighbor Clustering, Autoclass, Constrained clustering; semi-supervised: SVM + constrained clustering	SVM + constrained clustering (semi-supervised)
[209]	2013	Supervised, Semi-Supervised	Supervised: NB, C4.5, Bayes Network, kNN; semi-supervised: Proposed label propagation, AutoClass	Proposed label propagation (semi-supervised)
[210]	2013	Supervised, Unsupervised	Supervised: C4.5, k-NN, SVM, NB, NN, Bayes Nets; unsupervised: K-means with BoW	K-means with BoW (Unsupervised)
[202]	2014	Supervised, Unsupervised	Supervised: J48, NB, SVM; unsupervised: K-means, EM, DBSCAN	Features dependent
[13]	2015	Supervised	C4.5, Adaboost, Genetic Programming	C5.0
[81]	2016	Supervised	NBD, NBTree, BayesNet, HNB (Bayesian) and IBL, IBK, KStar (Lazy)	HNB (Bayesian), Kstar (Lazy)
[172]	2016	Supervised	C4.5, BN, RF, SVM, K-NN	C4.5
[211]	2016	Supervised, Semi-Supervised	Supervised: J48/C4.5 decision tree, k-NN, Best-first decision tree, Regression tree representative, Two-phased, Sequential minimal optimization, DT and NB, Bayesian Network, NB; semi-supervised: K-means + C5.0	K-means + C5.0 (Semi-Supervised)
[212]	2016	Unsupervised Semi-Supervised	Unsupervised: Fussy clustering; semi-supervised: label propagation (Zhang et al.) and the proposed method	Semi-Supervised proposed method
[28]	2016	Supervised	RF, SVM	RF
[64]	2016	Supervised	NB, Bayesian Network, AdaBoost, Bagging, OneR, PART, k-NN, J48, NB Tree, RF	Features dependent
[213]	2016	Supervised	RF, stochastic gradient boosting, extreme gradient boosting	RF
[60]	2017	Supervised	Nave Bayes, Bayesian Network, RF, DT, NB Tree, MLP	RF and DT
[33]	2017	Supervised	K-NN, SVM, Nave Bayes, WFNP (C4.5 based)	WFNP (C4.5 based)
[45]	2017	Supervised	ELM, GA-ELM, GA-WK-ELM	GA-WK-ELM
[49]	2017	Semi-Supervised	Ermans method, Zhang et al. method, proposed SSTCS	SSTCS
[167]	2017	Supervised	DT, RF	RF
[130]	2017	Supervised	NB, DT SVM, MLP, CNN	CNN
[31]	2017	Supervised	CNN, C4.5	CNN
[80]	2018	Supervised	TAN, ATAN, t-Cherry, LDLMCS	LDLMCS
[150]	2018	Supervised	Cascade forest (DL), RF, SVM, Bayesian Network, C4.5	Cascade forest (DL)
[170]	2018	Supervised	ANN, SVM, C4.5	C4.5
[173]	2018	Supervised	Bayesian Network, NB, SMO, Adaboost, Bagging, OneR, PART, Hoeffding, C4.5, RF, Random tree	RF and C4.5
[214]	2019	Semi-Supervised	Probabilistic graphical model [215], Offline/real-time semi-supervised classification [216], Bipartite graph-based maximization [217], Various-Widths Clustering [218], Semtra [214]	Semtra

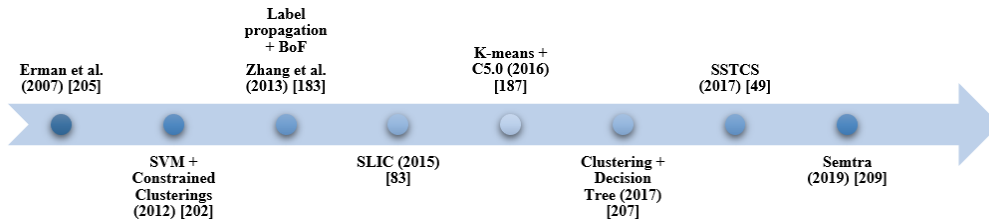


Figure 2.6: Trends of unsupervised ML methods used for traffic classification

pared their proposed method to Ermans method. The results showed that label propagation performs better than Erman’s semi-supervised method. In [212], the authors combine the unsupervised and supervised methods to perform semi-supervised classification. At the first stage, k-means is used for clustering Internet traffic. Then, the learned clusters are passed to a supervised decision tree-based classifier to label them based on few labeled instances [219]. Similarly, in [49], k-means is used to cluster the unlabeled data, and the supervised K-NN method is applied for labeling the unlabeled data. Likewise, a semi-supervised SVM-based method is proposed in [220].

SLIC [83], a Self-Learning Intelligent Classifier, is a classification system that self-learns by inducting the labels of unlabeled data from a small set of labeled data. Upon labeling new data, it retrains the classification model. A comparison with the *BoF-based k-NN* semi-supervised method on two real-world traffic traces shows the effectiveness of the proposed SLIC method. Another semi-supervised method using a Gaussian mixture model is proposed in [221]. More recently, Semtra, a self-training flow labeling method, was proposed in [214]. A comparison with state-of-the-art semi-supervised methods (probabilistic graphical model [215], offline/real-time semi-supervised classification [216], bipartite graph-based maximization [217], and k-NN various-widths clustering [218]) shows that the proposed method presents an improvement in terms of classification accuracy.

Comparisons between different ML-based methods are summarized in Table 2.3. It can be noticed that the supervised methods are more considered than other types of ML methods. Moreover, C4.5 seems to be the ML method that gives the best results in most of the cases. Another insight is that DT is the best suitable supervised ML method for traffic classification when statistical features are considered. After selecting the ML method, classifier accuracy is evaluated based on the defined classes. In this context, classified output is dependent on the defined classes, which vary in the literature. In the next section, the different traffic classifications are reviewed.

2.4 Traffic Classification Classes

In this section, we review the different objectives behind classifying Internet traffic, as illustrated in Figure 2.7, which shows the evolution in the Internet classification objectives. Some work considers classifying traffic based on application protocols. In this case, most of the reviewed work considers port number information. Another direction is to classify traffic into different categories. A new trend recently emerged in the classification domain, especially in the mobile network domain. It consists of classifying traffic based on the application name. Other approaches focused on classifying the actions carried out in a given application. With the emergence of IoT, there is a new classification direction consisting of identifying the IoT devices generating the traffic. In the following, we review the work related to each type of classification, as summarized in Table 2.4.

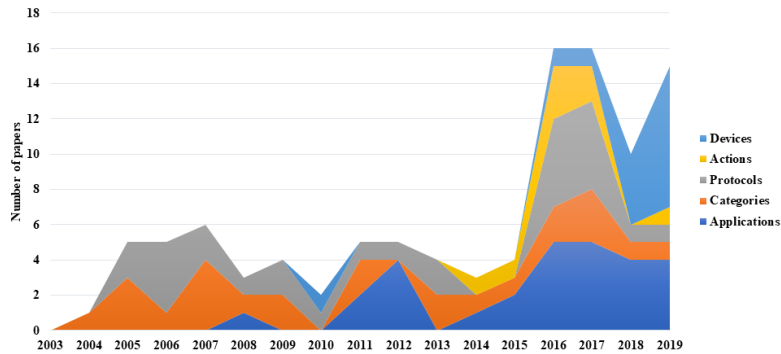


Figure 2.7: Internet Traffic classification objectives

2.4.1 Protocols Classification

Protocols classification is a traditional traffic classification approach that relies mainly on port number information; however, different researchers have considered traffic protocols differentiation based on statistical features. Bernaille et al. considered the classification of traffic into different application protocols: Post Office Protocol (POP3), Simple Mail Transfer Protocol (SMTP), Secure Shell (SSH), HTTPS, POP3S, HTTP, File Transfer Protocol (FTP), Edonkey, and Kazaa, using the start-of-flow [180, 182]. Crotti et al. introduced the protocol fingerprinting notion based mainly on statistical traffic characteristics (packet

Table 2.4: ML based Internet traffic classification objectives

Objectives	References	Description
Protocols	[14, 16, 33, 45, 60, 62, 64, 67, 69, 71, 76, 80, 81, 118, 181, 203, 209, 210, 219, 222, 223, 115, 224, 225, 226, 227]	The objective is to classify Internet traffic into different protocols such as: HTTP, FTP, etc.
Categories	[44, 46, 51, 53, 61, 74, 79, 107, 139, 144, 208, 220, 228, 229, 230, 231, 232, 233, 234, 235]	The aim is to categorize Internet traffic into classes reflecting the different traffic category, for example: VoIP, Video streaming, Web browsing, etc.
Applications	[13, 17, 18, 24, 28, 41, 103, 121, 129, 130, 150, 211, 213, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246]	The aim is to identify the name of the application generating the considered Internet traffic.
Actions	[19, 20, 27, 119, 175, 247, 248]	Knowing the application, the aim is to know the exact action performed by the user, for example: share, comment, like, etc.
Devices	[108, 120, 128, 249, 250, 251, 252]	The goal is to identify device models/types.

size and interarrival time) to classify traffic based on the used protocols: POP3, HTTP, and SMTP [222]. In [253], an active learning framework is proposed to extract features from unknown protocols using a small number of labeled instances. Williams et al. [67] compared five widely utilized ML algorithms to classify traffic into different protocols: FTP, Telnet, SMTP, DNS, and HTTP. Este et al. propose classifying traffic into different protocols: SMTP, HTTP, MSN, POP3, FTP, and BitTorrent using a one-vs-all classifier for each application protocol [71]. In [76], the authors apply a stream-based technique to deal with the dynamic Internet traffic nature. Zhanyi Wang [133] considered the first 1000 bytes of the payload of each TCP session to classify traffic into different application protocols. The Stacked Auto Encoder (SAE) DL architecture is used, and the results show that the accuracy can reach 99% for certain protocols.

Only known classes can be learned when applying supervised classification. However, Internet traffic is very dynamic, and there is need to detect new traffic classes. Accordingly, unsupervised ML methods have been employed to detect

unknown data and adapt to network dynamicity. To address the problem of unknown protocols, the authors in [210] propose a novel unsupervised approach based on clustering traffic flows according to their application protocol. A BoW model, to represent the content of the obtained clusters, is constructed using flow statistical features. Moreover, the Latent Semantic Analysis (LSA) is applied to aggregate similar traffic clusters based on their payload content. Erman et al. [62, 63] apply unsupervised ML algorithms to classify traffic into four classes: HTTP, P2P, POP3, and SMTP.

In [205, 219], a combination of unsupervised and supervised methods to classify traffic into different application protocols is proposed. Sun et al. combined signature-and statistical based approaches to classify traffic based on the used protocols [223]. First, they identified SSL/TLS traffic, then they applied statistical based classification to know the application protocol. The main limitation for this type of classification is traffic tunneling. In fact, mobile network traffic is mostly carried over HTTPS traffic. Moreover, traffic classification into protocols does not reflect the specifics of the classified traffic; therefore, this classification mainly serves protocol trends analysis and traffic abnormality detection.

2.4.2 Applications Classification

This type of classification aims at identifying the specific application that generate the considered flow of packets. This is challenging given that the same type of traffic (e.g., video call traffic) can be generated by two different applications (e.g., messenger, WhatsApp, etc.). Consequently, knowing the exact application requires extracting features specific to the application implementation and behavior. In this context, an identification of popular end-user applications (e.g., Facebook, Twitter, and Skype) was proposed [41]. The UNB ISCX network traffic dataset and an internal dataset consisting of well-known applications were used for testing. Lotfollahi et al. classified traffic at the packet level [21]. To do so, they used the IP header of the first packet and the first 1480 bytes of this packets payload (in total 1500 bytes per flow). Additionally, the authors choose to classify VPN vs. non-VPN traffic first and then identify the application name. AppScanner is proposed in [28], where 110 apps were chosen at random from the top free apps in the Google Play Store in July 2015. In [29], an extension of AppScanner is presented. In [254], 160 applications are considered for classification using a three-layer classifier. Wang et al. proposed an app identification scheme based on the side-channel information [23]. Frame size and interarrival time, with their derivate statistics, are used as classification features.

This type of classification serves QoS management and traffic trends analysis, as well as network security functions (e.g., firewall). However, the applications are becoming more and more similar implementing the same options (chatting, video call, voice call, etc.). Therefore, knowing only the application name might not

infer the exact QoS or security requirements.

2.4.3 Actions Classification

This type of classification aims at knowing the action performed by the user within the application (e.g., video call, voice call, comment, like, etc.). In [119, 175], Conti et al. classify traffic based on user actions. For each application, they considered a set of actions, for example, the following actions are considered for Facebook: send message, post user status, open user profile, open message, status button, post on wall, and open Facebook. In [27], the authors propose a way to inspect user specific actions through IP packet header analysis. The main used features are the interarrival time, send/receive packet sizes, and count ratios. K-Means is used for behavioral models extraction, and an SVM is applied for specific user activities within mobile application detection. In [20], an online mobile application traffic analyzer is proposed for classifying mobile applications traffic into different types of actions. Three experiments are performed considering three applications: WeChat, WhatsApp, and Facebook. The authors in [211] proposed a two-phase method to classify flow-based traffic based on the user actions. In the unsupervised phase, approximately 6.8 million bidirectional flows for all applications were collected and clustered into 12 unique flow classes. In the supervised phase, they used four different feature sets of NetFlow attributes from the derived flow classes. The C5.0 ML classifier was used. An average prediction accuracy of 92.37% was achieved with one of the feature sets comprising 14 NetFlow attributes, and this increased to 96.67% with adaptive boosting. However, this type of classification presents privacy concerns, since it reveals user activities. Moreover, many applications share common functionalities, so classification accuracy might be affected when considering traffic containing the same actions from different applications.

2.4.4 Categories Classification

The aim of this type of classification is to divide traffic into categories presenting different QoS or security requirements. In fact, the different traffic categories might be generated by the same application and carried over the same protocols. In this context, Roughan et al. define four categorical traffic classes: interactive, streaming, bulk data transfer, and transaction [61, 255]. They linked each class to certain application protocols: interactive to telnet, streaming to Realmedia, bulk data transfer to FTP, Kazaa, and transaction to HTTPS and DNS. In [6], Qin et al. investigate the possibility of using ML algorithms to classify traffic into different QoS-based classes: bulk data transfer, interactive, services, and multimedia. In [208], Wei Li et al. proposed a real-time classification approach. The considered data consists of a set of TCP flows representing essential Internet ap-

applications classes: web-browsing, mail, bulk (FTP), attack, P2P, database, multimedia services, and interactive. In [230], Jiang et al. investigate the possibility of achieving good classification accuracy using only the NetFlow records. Karagianis et al. [107] studied multi-level traffic behavior such as analyzing interactions between hosts, protocol usage, and per-flow average packet size. Their aim was to classify traffic into attack, web, games, chat, P2P, DNS, FTP, streaming, and mail. In their recent work [108], they presented an interesting investigation to profile users activity and behaviors and to analyze the dynamic characteristics of the host behaviors. Moore and Zuev in [44] presented a statistical approach to classify traffic into different types of services: bulk, database, interactive, email, services, WWW, P2P, attack, games, and multimedia. A Nave Bayes classifier combined with kernel estimation and a correlation-based filtering algorithm was used to classify traffic flows in an offline mode. In [46], Zhu et al. applied SVM to identify 7 classes of applications: www, mail, FTP-control, FTP-pasv, attack, P2P, database, FTP-data, multimedia, services, and multi-class. In [229], Wang et al. classify traffic into QoS-based classes (voice: GoogleVoice; video conference: Skype, GoogleTalk; Streaming: USstream, Sopcast; bulk data transfer: FTP, Mega; interactive data: SSH, Telnet; and best-effort trafrc: default class). The presented framework consists of two parts: detection of large flows at the software-defined switches level and QoS labeling done at the controller level. In [231], an unsupervised ML method is proposed to classify traffic into QoS-based classes. The classification is performed in two phases. In the first off-line phase, unsupervised Self-Organizing Map (SOM) and K-means clustering algorithms are used. In the second classification phase, the K-NN classifier is used for supervised data classification. In [234], the down-/upstream video streaming rates are considered to classify video traffic into broadcast video, Web video trade style video, barter style video, and interactive video. In [256], classification of video content based on the byte code distribution is proposed. Two classes of videos are considered: action and romance. Another work considers the classification of multimedia traffic into five categories based on their criticality level: non-critical, low critical, some critical, critical, and very critical [257]. In [31], Wang et al. consider categorical classes for VPN and non-VPN data using the ISCX VPN-non-VPN dataset [39]. These classes are: chat, file transfer, mail, streaming, torrent, and VoIP.

The importance of this type of classification lies in its independence of the protocols and port-based information. Consequently, it is possible to classify the traffic based on the statistical characteristics even if some obfuscation techniques are employed. Moreover, this type of classification is key to manage QoS and in some cases, might be used for abnormal traffic detection.

2.4.5 Devices Classification

With the IoT emergence, a new type of classification appeared to identify IoT devices based on their generated traffic. This type of classification can be also called device fingerprinting. In [249], Miettinen et al. proposed a method to identify the type of a device based on the packets generated during the setup phase. A set of 26 devices is considered. The presented method consists of extracting a fingerprint composed of a number of unique packets from the setup phase. Classification is done in two steps. First, the device fingerprint is classified using the trained classifiers. In a second phase, if a device is classified as belonging to several classes, the Damerau-Levenshtein edit distance is computed to identify the most probable class. In [258], the authors propose a self-learning identification approach to overcome the one-time identification issue. The new approach consists of extracting a signature based on the device type without knowing the exact device model. In [259], a technique to detect unauthorized IoT devices is proposed. By collecting data from 17 devices pertaining to 8 types of IoT devices, the authors build a multi-class classifier to detect unauthorized devices. In [260], Siby et al. consider the physical layer communication patterns to identify devices connected to a certain network. In [261], the identification of 9 IoT devices was presented by using a total of 180 traffic features, based on the packet size and interarrival time for three types of directions (client to server, server to client, and bidirectional). In [251], a multi-stage classification method is presented. In the first stage, classification is done using the BoW method applied to the port numbers, domain names, and cipher suites. At the second stage, an RF classifier is used with statistical features including the flow volume, flow rate, flow duration, sleep time, DNS interval, and NTP interval. In [262], Lei et al. propose an automatic classification of new IoT devices based on network traffic flows. A Long Short-Term Memory with Convolutional Neural Network (LSTM-CNN) cascade model is used after preprocessing the data and extracting the feature. In [263], physical layer fingerprinting is proposed by leaking the physical channel information, including signal power, attenuation, interference, etc. A DL-based IoT authentication framework is proposed in [264]. The authors apply LSTM to detect imperfections in the transmitters signals of low-power devices. The proposed method differentiates between legitimate and illegitimate devices, including high-power devices that try to imitate legitimate low-power devices. User profiling, by monitoring IoT devices network activity in a smart home, is proposed in [120, 252]. However, devices might generate different types of traffic, so knowing the device type does not necessarily imply specific traffic characteristics.

Due to variable traffic classification goals, different approaches have been proposed. Classifying the traffic based on the different protocols, applications, categories, actions, or devices might serve many network applications including traffic analysis, QoS management, intrusion detection, and device authentications,

among others. Consequently, having a hierarchical classification framework with dynamic granularity is key to meet the diverse network management requirements. In fact, the correlation between the results of the different classification outputs decreases the probability of error. For example, knowing the type of traffic along with the types of devices generating traffic makes it possible to link the different classification results and ensure improved classification accuracy (e.g., a streaming traffic is not expected to be generated by a printer). However, traffic classification encounters a real challenge when traffic obfuscation techniques are employed. These techniques aim at modifying traffic characteristics to preserve user privacy or avoid censorship. Being aware of these techniques helps design classifiers in a way that avoids performance degradation when obfuscation is used. To avoid misclassification, the classifier might use rigid features that cannot be modified by obfuscation or it can simply detect the obfuscated traffic by means of unsupervised learning. In the next section, we review the obfuscation techniques used to thwart traffic classification.

Table 2.5: Traffic obfuscation techniques

Technique	Year	Approach	Summary
Infranet [265]	2002	Morphing	Modulate sequence of HTTP requests to send data
eMule [266]	2007	Encryption	Encrypted stream socket
Dust [267]	2011	Encryption	Using stream cipher
CensorSpoofers [268]	2012	Morphing	HTTP traffic is morphed to VoIP traffic
StegoTorus [269]	2012	Morphing	TOR plugin: hide TOR traffic to appear like Skype traffic
SkypeMorph [270]	2012	Morphing	TOR plugin: hide TOR traffic to appear like Skype traffic
BuFLO [271]	2012	Mutation	Traffic shaper that outputs fixed size packets at fixed times
Obfs2 [272]	2012	Encryption	TOR plugin: Full encryption of messages
Flash Proxies [273]	2013	Tunneling	Plaintext tunnel using web sockets through browser-based proxies
ScrambleSuit [274]	2013	Mutation	TOR plugin: Uniform key exchange, full encryption of messages
Freewave [275]	2013	Morphing	Transfer data over VoIP connections
Obfs3 [276]	2013	Encryption	TOR plugin: add flow signature
FTE [277]	2013	Morphing	Format-Transforming Encryption
Meek [278]	2014	Tunneling	Utilizing different domain names
Blindspot [279]	2014	Steganography	Hiding traffic into social network data
Deepflow [280]	2014	Steganography	Hiding Internet traffic in P2P traffic
Facet [281]	2014	Steganography	Hiding video streaming traffic into Skype video call
Obfs4 [282]	2014	Encryption	TOR plugin: providing protocol obfuscation based on SSH or TLS
psOBJ [283]	2014	Mutation	Inserting pseudo-objects with different number and different sizes
coOBJ [284]	2014	Mutation	Combined objects are inserted into the HTTP traffic
CPSMorph [285]	2014	Morphing	Mimicking the delay time distribution
Muffler [286]	2015	Morphing	Extracting Super distribution to which streams can be morphed
SkypeLine [287]	2016	Morphing	Direct-Sequence Spread Spectrum (DSSS) based steganography to hide information in VoIP communication
CovertCast [288]	2016	Steganography	Hiding the content of web pages into images
DeltaShaper [289]	2017	Steganography	Hiding data in the form of images into Skype video call
Liberate [290]	2017	Mutation	A suite of classifier-evasion techniques: packet insertion, payload splitting, packet reordering, classification flushing
Tamaraw [291]	2014	Morphing	Different time intervals for uplink and downlink packets in the open-world setting

2.5 Thwarting Internet Traffic Classification

Known as traffic watermarking techniques or traffic obfuscation techniques, this section reviews the techniques proposed in the aim to thwart traffic classification [292, 293, 294]. In this case, traffic classification is considered as an attack that compromises the user privacy. Conversely, traffic classification thwarting techniques can be employed maliciously by attackers to perform their attacks without being detected by the ML-based Intrusion Detection System (IDS). The debate continues among the defenders of the legality of classification and thwarting. In this chapter, we focus on the technical aspect of both techniques.

In either case, a deep understanding of the anonymization techniques helps in designing a robust classifier. In [238], Iwai et al. propose an adaptive identification based on flow statistical features. In the evaluation part, the proposed method was tested against existing classification-thwarting approaches like Tamara [291], Pad to the Maximum Transmission Unit (MTU) [271], Direct Target Sampling (DTS), and Traffic Morphing (TM) [295].

While for traffic classification, we reviewed the ML-based methods, for traffic obfuscation, we review all the methods that affect the traffic statistical characteristics. Note that a new direction is emerging in the ML domain, which is the adversarial learning, which can be employed for traffic obfuscation in the future. In the following, we categorized the classification-thwarting techniques into six types: encryption, steganography, tunneling, anonymization, mutation, morphing, and physical layer obfuscation. We classify the reviewed work in Table 2.5.

2.5.1 Encryption

Some Internet applications rely on private information that needs to be protected against theft, disclosure, copying, use, or modification. Consequently, traffic encryption was proposed to hide the private information carried over the Internet, as illustrated in Figure 2.8-a. To this end, several application-layer security protocols have been proposed: HTTPS, FTPS, etc. Concerning traffic classification, the encryption hides the application signature needed for deep packet inspection-based classification methods. Additionally, some encryption algorithms require fixed plaintext length, which might affect the packet size distribution of the considered application/protocol. This might impact the accuracy of the ML classifiers that rely on packet size-related features. However, encryption does not hide the information carried in other features such as the interarrival time. Numerous ML methods have shown their effectiveness to classify encrypted Internet traffic based on the packet size and interarrival time distributions [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33].

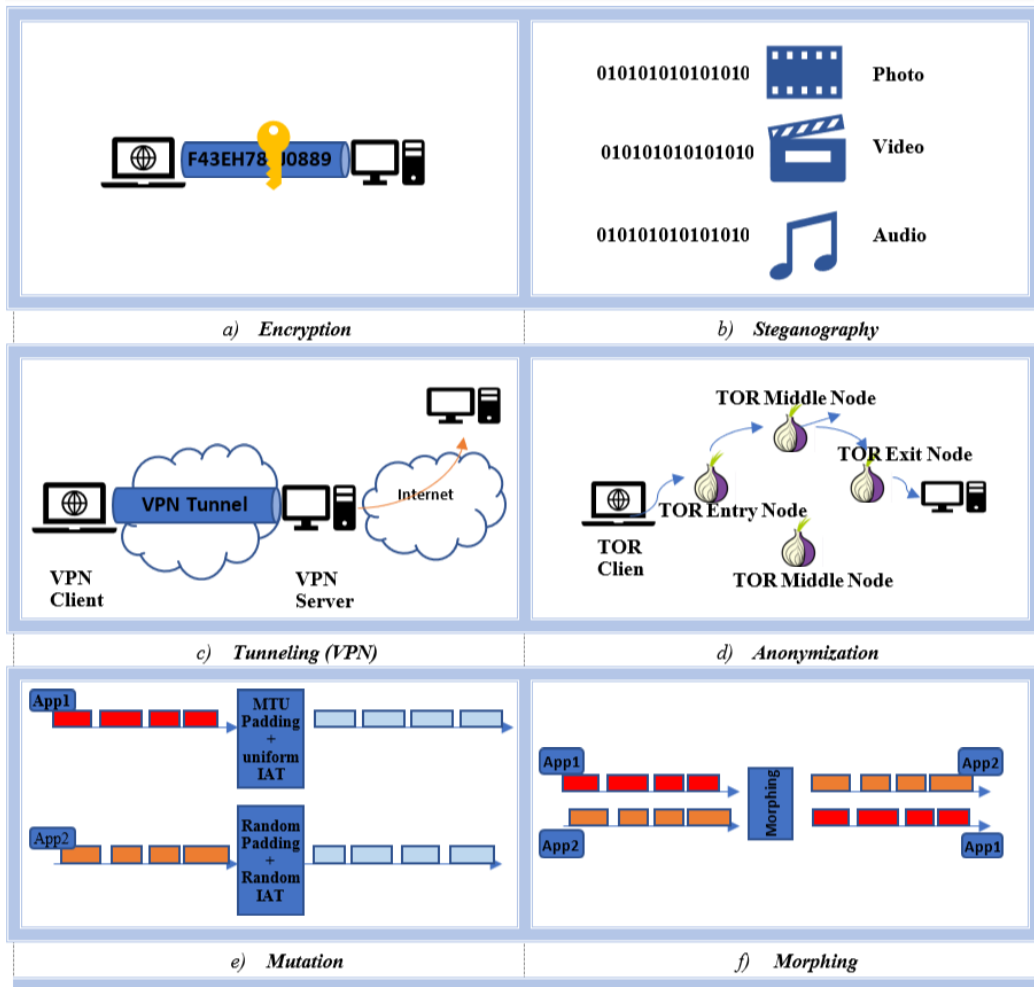


Figure 2.8: Internet traffic obfuscation techniques

2.5.2 Steganography

Steganography is a method consisting of hiding secret data in multimedia documents, as illustrated in Figure 2.8-b. In the network domain, steganography aims at hiding application data into other application packets. Unlike encryption, the goals of steganography are to hide the secret data as well as the fact that these data were sent over the network (i.e., unobservability).

To ensure undetectability of certain protocols, steganography-based methods have been proposed in the literature [296, 297, 298, 299, 300]. One of these methods is *Blindspot* proposed in [279], which is designed to use the current social network to provide the following properties: indistinguishability between anonymous and non-anonymous traffic, unlinkability between sender and receiver, and low delay.

Similarly, *Deepflow* [280] aims at hiding TOR traffic in P2P traffic. Given

the fact that even anonymous communication traffic (i.e., TOR traffic) presents specific patterns and can be detected by censors, Deepflow aims at hiding anonymous communication traffic in the P2P streaming networks (e.g., PPStream) using steganography. To do so, the Deepflow nodes connect to the PPStream network and behave as normal PPStream clients watching video, but they include their data into the communicated video packets by steganography. These packets are transmitted through the normal PPStream nodes to reach the destination.

Facet, another steganography-based obfuscation method [281], aims at hiding the video traffic in Skype video call traffic. To this end, the Facet client sends a message to the Facet server containing the URL of the video it wants to watch. Then, the Facet server downloads the requested video. The video content is passed to a microphone and camera emulators to be transmitted over a Skype video call to the Facet client. The audio sampling frequency and frame density are adjusted to mimic a normal Skype video call and avoid abnormality detection.

CovertCast, proposed in [288], hides the content of web pages into colored matrix images transmitted in a live streaming session (e.g., YouTube). Thus, the transmitted data can be received by multiple clients simultaneously. The clients demodulate the colored matrix images to extract the desired web content. In [289], **DeltaShaper** is proposed to hide the application data into images transmitted through a bi-directional Skype video call, similar to CovertCast. In addition, similar to Facet, a colored matrix carrying the wanted data, is inserted in the video call running in the background. In [301], an experimental study was conducted to evaluate the above mentioned systems: Facet [281], CovertCast [288], and DeltaShaper [289]. The results show that existing ML techniques, specifically decision trees, can detect these methods.

2.5.3 Tunneling

While traffic encryption aims at securing the communicated data during the application sessions, it does not ensure complete privacy protection. In fact, the session metadata (i.e., IP addresses, port numbers etc.) can be disclosed by the attackers, compromising both user privacy and anonymity. Tunneling protocols intend to hide the connection metadata and ensure user privacy.

One famous tunneling service is the **Virtual Private Network (VPN)**. VPN relies on a set of security protocols (e.g., IPSec, IKE, SSL, etc.) to extend the local private network to the public one. As shown in Figure 2.8-c, VPN consists of establishing a tunnel between the VPN client and server, and then the server forwards the client packets to the corresponding destination. The source IP address that appears at the destination is the VPN server IP address. Additionally, the VPN client encrypts data before sending it. However, recent research work in the traffic classification domain has shown the capability of classifying VPN

traffic based on the application name and traffic type [31].

2.5.4 Anonymization

The TCP/IP networking scheme requires well-defined mechanisms providing useful information for traffic classification. One of these mechanisms is the routing process that relies on the IP addresses, port numbers, MAC addresses; hiding this information is key for anonymizing the communicated traffic. In this context, multi-path routing [279, 302, 303] and NATing [304] are among the network functions that affect the accuracy of traffic classification methods, especially those ones relying on the IP and port information.

In this context, The Onion Router (TOR) was proposed to provide traffic anonymity by means of a list of onion routers [305]. The name onion refers to the fact that the TOR packet (called cell) consists of several security layers, and at each node (relay), the cell is decapsulated by the corresponding key. The decapsulated packet is sent to the next TOR. Consequently, the full path is anonymous to the TOR nodes. The TOR network consists of a list of TOR nodes, as shown in Figure 2.8-d. To communicate over TOR, the client has to use a TOR browser. The TOR client will have a predefined list of TOR nodes. To start a session, the TOR client browser contacts a random set of TOR routers and establishes a secret key with each one. These keys are used by the TOR client to encapsulate the TOR packet with different security layers and by the TOR nodes to decapsulate the received packets.

In the VPN case, client anonymity is lost if the VPN node is compromised. However, in the TOR case, client identity is anonymous even to the TOR nodes, given that the full routing path is anonymous. However, many attacks have succeeded in identifying TOR traffic in the aim to block it [306]. These attacks mainly rely on tracking the list of known TOR nodes and trying to detect and compromise TOR node entry and exit. Several traffic classification methods have demonstrated abilities to classify TOR traffic [143, 307, 308, 309]. Recently, many efforts have been made to enhance TOR security by means of TOR bridges and Pluggable Transport Protocols [310, 311].

2.5.5 Mutation

Traffic encryption/tunneling can hide information carried in the packet payload and connection identifiers. However, classification is still possible using statistical flow features, mainly the interarrival time and packet size. To thwart traffic classification, traffic mutation aims at modifying these statistical features, as shown in Figure 2.8-e.

In this context, ***Padding*** is a technique used to hide packet size information. On the other hand, traffic ***shaping*** aims to hide the interarrival time information. A classification of these techniques is included in [312, 313]. For the packet padding

techniques, five methods are described below, where l is the packet original size and $m(l)$ is the size after mutation.

- ***Padding to the Maximum Transmission Unit (MTU)***: this consists of padding the packets to the same size, which is the MTU. In this case, the transformation function can be written as: $m(l) = \text{MTU}$. This method completely hides packet size information but with a very high bandwidth overhead.
- ***Linear padding***: this technique consists of padding the packets based on the following equation: $m(l) = \lceil l/c \rceil * c$, where c is a system parameter and $\lceil l/c \rceil$ is the ceiling of l/c . In this case, the obtained packet lengths are multiples of c .
- ***Exponential padding***: this technique consists of padding the packets exponentially following the equation: $m(l) = \min(2 \log_2 l, \text{MTU})$. In this case, the transformation graph will have a plateau after a certain value of l .
- ***Elephants and mice padding***: this technique consists of padding the packets to a certain value c , if their size is less than c , if not, they are padded to the MTU.
- ***Random padding***: this method consists of randomly padding the current packet size to a size randomly chosen to be between l and MTU, such that: $m(l) = \text{RAND}([l, \text{MTU}])$.

The interarrival time-based mutation, listed in [314], can be summarized as follows:

- ***Constant Interarrival Time (CIT)***: this technique consists of sending the packets at a fixed interarrival time. Like the padding to MTU, this method hides the information carried in the packet interarrival time. However, it presents a very high overhead in terms of latency.
- ***Variable Interarrival Time (VIT)***: this method consists of sending the packets at random time intervals. These intervals are chosen randomly from a uniform distribution between two values I_1 and I_2 .

The trade-offs between overhead, performance, and privacy/information-leakage were considered in [315, 316, 317, 318, 319]. In [315, 316, 317, 318, 319], Lacovazzi et al. formulated traffic masking as an optimization problem to ensure minimal overhead in terms of packet length and interarrival time.

2.5.6 Morphing

Mutation techniques aim at thwarting classification and confusing the classifier. However, morphing techniques aim at confusing the classifier into classifying the target protocol/application traffic as another type, as illustrated in Figure 2.8-f. This is used mostly to avoid censorship of certain protocols/applications and make them look like legal protocols/applications.

Wright et al. proposed a morphing technique to thwart statistical traffic analysis [295]. The proposed solution aims at morphing one class of traffic to look like another class by applying convex optimization techniques. Compared to the padding technique, their proposed method reduces the accuracy of the traffic classifiers with much less overhead. They evaluate their method against two traffic classifiers for VoIP [29] and web traffic [14].

In [320], Wang et al. show that TOR traffic can be detected even after obfuscation with two variants of *obfsproxy*, Format-Transforming Encryption (*FTE*), and two variants of meek. Different proposals for securing TOR traffic have been suggested to counter this [269, 270, 274]. *StegoTorus* [269] is a TOR plugin that aims at providing: *undetectability* and *unblockability* to the TOR traffic. StegoTorus uses steganography to make TOR traffic look like traffic produced by another software. In addition, the fixed-size cells are transformed into variable-length packets, and a novel cryptosystem is applied to make the cyphertext look like random data. The obtained traffic looks like encrypted P2P or HTTP traffic. *SkypeMorph* [270] is another TOR plugin to hide TOR traffic inside Skype traffic. First, a Skype texting session is initiated with a selected out-of-band bridge to exchange public key material. To do so, the client needs to have the bridges Skype ID. After sharing the secret, a video call is initiated between the client and TOR bridge. The data are shaped to look like the audio and video content of a normal Skype call and sent over the instantiated Skype video call session.

ScrambleSuit [274] aims at avoiding TOR blocking by means of two added features: authentication by a secret key shared out-of-band and avoiding traffic analysis by morphing the traffic to resemble whitelisted protocols. To this end, ScrambleSuit determines the shapes of whitelisted protocols by analyzing their packet lengths and interarrival time distributions.

However, the authors in [321] demonstrated that SkypeMorph, StegoTorus, and ScrambleSuit are vulnerable to detection. In fact, the morphed protocols have specific behaviors when certain network actions or conditions occur. Mimicking the reaction to network errors and to specific network conditions is key to avoid detection. In addition, the sessions of mimicked protocols are usually initiated with other protocol sessions, known as Side Protocols (e.g., VoIP session involves three protocols: SIP for signaling the session, RTP for streaming the media, and RTCP). This includes the intra- and interdependence between these protocols. The de-anonymization attacks check the initiated sessions and reactions against network errors to detect the anonymized/morphed protocols [322].

In [323], using Generative Adversarial Network (GAN), Rigaki et al. propose a morphing method for generating network traffic that mimics legal types of traffic. The aim was to avoid malware detection by shaping the malware traffic to look like Facebook chat traffic. In the experimentation phase, a Stratosphere behavioral IPS was installed in a router, while the malware and the GAN were deployed in the local network. The results show that the malware traffic modified by GAN is successfully undetectable. However, the network obfuscation techniques are detectable by statistical characteristics [324]. For example, traffic morphing is only effective when the classifier considers the same feature(s) targeted by the morphing routine [271]. Consequently, traffic classification is still possible considering the existence of unmorphed features.

In this context, in [271], a comparison is conducted between three classifiers on nine obfuscation techniques including tunneling, padding, and mutation. The results show that all considered methods are prone to detection by statistical analysis. Consequently, the authors propose Buffered Fixed-Length Obfuscator (**BuFLO**), which consists of sending fixed size packets at fixed time intervals, to hide all the statistical information. However, BuFLO provides traffic anonymity at the cost of latency and bandwidth overheads.

Mimic Hunter is another proposal to detect protocol mimicry. This is performed by studying the protocol structure, syntax inspection, protocol state, transmit verification, protocol behavior, anomaly detection, network traffic decision making, and protocol profile library [325]. By analyzing the IP traffic statistical features (i.e., packet sizes, interarrival time, and packet order), **tunnel hunter** can successfully identify protocols tunneled inside other protocols such as HTTP, DNS, and SSH [326].

2.5.7 Physical Layer Obfuscation

Traffic classification in the wireless network domain can be performed using side-channel information leak and signal-related patterns. Therefore, thwarting traffic classification employs the same techniques used for wired networks: padding [327, 328, 329, 330], morphing [295], and other approaches specified to the wireless domain: pseudonym [331, 332], identifier-free [333, 334, 335], frequency hopping [336], and jamming [337, 338, 339]. For example, in [340], a traffic-shaping method is proposed that creates multiple virtual Media Access Control (MAC) interfaces over a single wireless card, dynamically scheduling the packets over these interfaces, and reshaping the packet features over each virtual interface.

To summarize, the security goals and detection methods for the different traffic obfuscation techniques are highlighted in Table 2.6.

Table 2.6: Obfuscation techniques

Obfuscation	Goal	Detection Method
Encryption	Confidentiality	Statistical based ML
Steganography	Unobservability	Behavioral based ML
Tunneling	Unlinkability	Statistical based ML
Anonymization	Unlinkability & Undetectability	Statistical based ML
Mutation	Undetectability	Behavioral based ML
Morphing	Unobservability & Undetectability	Behavioral based ML
Physical Layer Obfuscation	Unobservability & Undetectability	Behavioral and statisti- cal based ML

2.6 Key Findings, Limitations, and Recommendations

Internet traffic classification is a well-established research domain with several remaining limitations and issues [34, 341]. In the following, we present our key findings after reviewing the most relevant work related to applying machine learning for traffic classification.

2.6.1 Key findings

- **Data collection:** the data used for traffic classification in many of the reviewed works is outdated. Internet traffic is continually evolving with the emergence of new kinds of traffic, devices, and applications. Therefore, collecting labeled Internet traffic considering new trends is required. In addition, most of the available public datasets are unlabeled or were labeled using unreliable methods such as using deep packet inspection that fails with encrypted traffic or port-based labeling that fails with dynamic port allocation.
- **Data representation:** different methods have been used for representing traffic. In [117], Moore et al. provide a comprehensive set of features based on the packet size and interarrival time, TCP flags, port numbers, and IP addresses. Other methods consider the domain names and protocol request contents. Time series-based features are used for fine granular classification. Another proposed approach is to represent communication patterns between the invoked entities using non-traditional formats. Traffic flows have recently been represented as images. In fact, data representation is chosen according to the classification method. For example, the time

series presentation is used for Markov Model-based classification and the image-based representation is used for DL-based classification, etc.

- **Method selection:** different ML methods exist, and each has advantages and disadvantages. However, not all methods are suitable for traffic classification. DT-based methods have shown their effectiveness in traffic classification [342]. Traffic classification objectives: Internet traffic classification is key to managing both QoS and security in the network. In the literature, different views of the traffic have been considered including protocols, applications, categories, actions, and devices classification.
- **Obfuscation methods:** different obfuscation techniques have been proposed to preserve user privacy by thwarting traffic classification. Some obfuscation methods like the fixed size and fixed interarrival time mutation technique and steganography can avoid any statistical classification. However, the detection of the mutated traffic in these cases can be done using behavioral classification.

2.6.2 Limitations

- **Networking functions:** asymmetric routing, NATing, and tunneling are network functions that influence traffic classifier performance [343]. Therefore, these network processes must be taken into consideration when designing a traffic classifier. The features must be chosen to not be affected by any of these network functions.
- : the classification process needs to be lightweight, especially when applied in real-time processing mode [344]. Data preprocessing, data representation or features computation, and the classification computation overhead must be carefully analyzed from time and processing complexity perspectives. In this context, three main measures are essential: memory space, computational complexity, and processing time. Designing a robust yet accurate classifier is key, but overhead is vital in several cases, especially for real-time network services (e.g., intrusion detection).
- **Real implementation:** Although ML traffic classification has been extensively considered in the literature, few classification frameworks/tools are available [246, 345]. Practically, DPI is still being extensively deployed although it presents privacy concerns. Relying on ML-based classification is still in its infancy; but big names like Cisco [346], Huawei [347], Paloalto [348], and IBM [349], are actively working on combining ML and network functions.
- **Obfuscation:** the legitimacy of classification and obfuscation is controversial. From a privacy perspective, classification is considered as an attack

that compromises user privacy. However, from a network management perspective, obfuscation can be used by the attackers to avoid detection of their attacks. In this context, security and QoS applications favor classification while privacy favors obfuscation.

- **Traffic Sampling:** One of the challenges that hinders traffic classification application is the high speed required at the core network. Extracting features from the packets at the very high speed is not feasible, and therefore traffic sampling becomes an alternative. This can modify traffic characteristics and statistical features and might have a negative impact on the classification accuracy.
- **Data Collection:** data are very important for any ML classification framework. Training the model with representative data is key to ensure the extraction of meaningful patterns that help classify unseen data with high accuracy. Data needs to be collected from different network environments and points (edge, core). Moreover, data labeling should be performed accurately. Data variety is also important; having data from different applications and devices would enable innovative classification applications. Finally, public data availability is key to empower research in this domain.
- **Model Generalization:** if an ML model is generalizable, it means that its application to unseen data presents low bias and variance. To ensure generalization, the model should be tested on data collected from different network environments, which requires operator involvement.
- **Unknown traffic detection:** traffic classification aims at knowing the specific traffic type, application name, or even device type. However, these classes are not static. New applications, devices, and traffic types emerge constantly in the network domain. Consequently, detecting new traffic or attack traffic is key to avoid misclassification. In this case, classification model uncertainty needs to be evaluated relative to future traffic types, and techniques must be further developed for anomaly detection (malicious versus non-malicious anomalies).
- **Robustness:** making the classifier robust to obfuscation or to detect obfuscated traffic is key to avoid misclassification. In this case, unsupervised learning is needed to detect unknown traffic classes. Classifiers should be tested against the different obfuscation techniques, and the features should be chosen to be minimally affected by obfuscation.
- **Model update:** as mentioned before, new traffic types are emerging dynamically in the network domain, especially with the connection of new kinds of devices in the IoT era. This requires re-training the model with

emerging traffic types. Developing ML methods for online training is key to keeping the traffic classifiers updated.

- **Hierarchical classification:** as the classification objectives might differ based on the targeted network function (e.g., QoS management, firewall, intrusion detection, application banning, etc.), providing a hierarchical classification with dynamic granularity level is important to avoid developing several classification modules and thus training several classifiers [350].

2.7 Conclusion

Applying ML in the network domain is one aspect of the future networks, where intelligence will enable innovative network functions including intelligent routing, traffic prediction, and intelligent traffic management. Within this context, traffic classification through ML is essential to enable the intelligent network functions of the future. In this chapter, we reviewed traffic classification through its workflow, including data collection, data representation, ML method selection, and class definitions. In addition, we reviewed obfuscation methods designed to evade traffic classification. After reviewing the literature, the question becomes how to design a robust classification framework that is able to give real-time classification, while accounting for adversarial attacks.

Chapter 3

Data Representation

Convolutional Neural Network (CNN) is a DL architecture that is initially used for image recognition. Being inspired by the visual cortex, CNN has presented very high accuracy when applied on raw images. Inspired by this, the representation of network traffic as images is considered as input to CNN used for traffic classification. A recent work has applied CNN for traffic classification, by transforming the first packet of a network flow to an $n \times n$ gray image [351, 132, 21, 42, 133, 352, 134]. In our work, we 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 [353]. In this chapter, we analyze and compare these 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.

3.1 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.

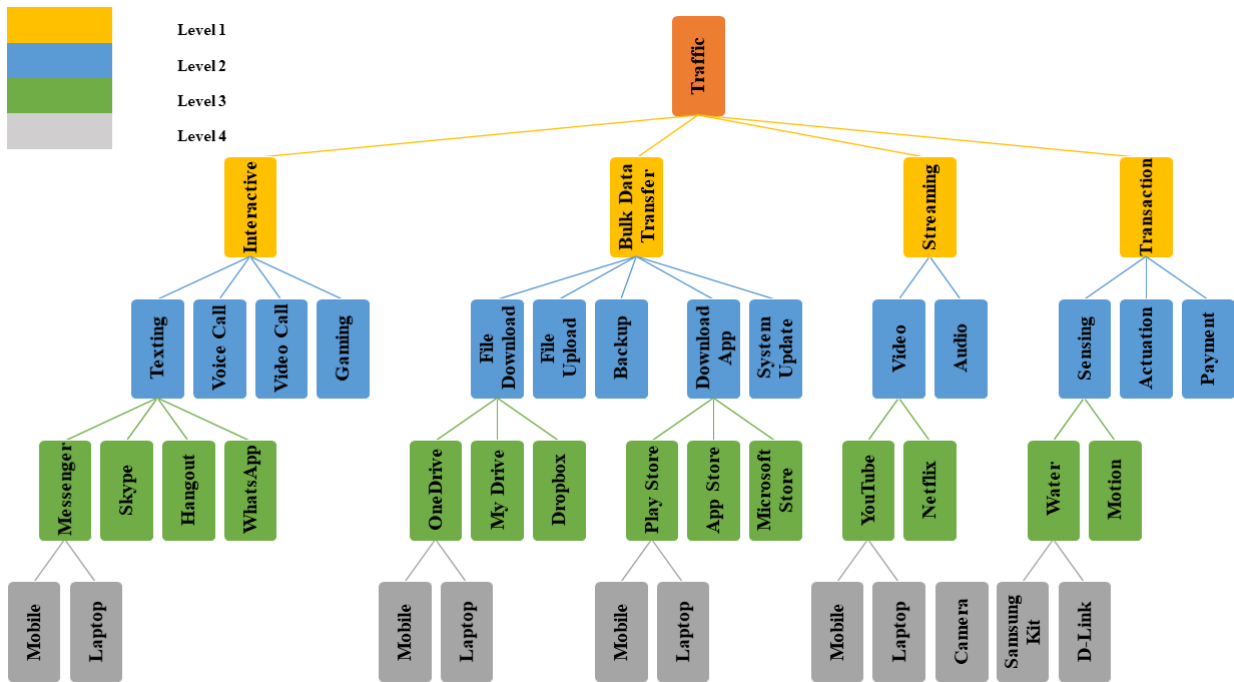


Figure 3.1: Hierarchical Internet Traffic Classification

3.1.1 Hierarchical Classification

As shown in (see Figure 3.1), our classification framework consists of four levels. At level 1, the traffic is classified in one of the four classes: interactive, bulk data transfer, streaming, and transaction. Below is the definition of these classes:

- *Interactive*: this class of traffic 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, having a user 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.
- *Transaction*: 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:



Figure 3.2: Data Collection Setup

file download, file upload, backup, download app, and system update. For the *streaming* traffic, two classes are considered: *video*, and *audio*. Finally, for the *transaction* 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 installed to collect IoT traffic.

3.1.2 Data Collection, Preprocessing and Representation

To train the classification model, data was collected from different types of devices and applications. In order to collect data from the Wi-Fi enabled devices (D-Link Water Sensor, D-Link Camera, D-Link Siren, D-Link Plug, Laptop, and Mobile Phone), a Linux machine was configured as an access point (as shown in Figure 3.2a)). A hotspot network was added, and forwarding was enabled on this device. An entry to the iptables was added that allowed the forwarding of traffic arriving from the WLAN interface (Wlan0) to the Ethernet interface (eth0) connected to the home router. To collect data from the Samsung home kit devices

connected to a hub, the hub was connected to the Ethernet interface(eth0) of the Linux machine (as shown in Figure 3.2b)). Similarly, forwarding was enabled and an entry to the iptables was added to forward all incoming traffic from the eth0 interface to the wlan0 interface connected wirelessly to the home network. One hour of traffic for each type of traffic per device using Wireshark launched on the Linux machine was collected. Table 3.1 summarizes the collected traffic per device and per application. The collection was performed for one hour for each device and application.

In addition, we consider two online data-sets to analyze the effect of tunneling and anonymization. The first data-set is the VPN and non-VPN data-set [354], and the other one is a TOR and non-TOR data-set [355]. These data-sets consist of six classes each, including: chat, file transfer, mail, streaming, torrent, and Voice over IP (VoIP).

Before preprocessing the obtained PCAP files, we filtered them to have only the TCP and UDP sessions. Using the dpkt python library, the flows are extracted per network flow. For each 16 packets of each flow, the features (packet size, the inter-arrival time, the direction, and the transport protocol) are extracted to form a $4 \times 4 \times 4$ vector. To do so, all the flow packets features are recorded in a flow list. The extracted features are normalized within the range [0, 1]. The inter-arrival time is computed by subtracting the packet timestamp from the previous packet timestamp. If the computed time delta is greater than 1 second, the inter-arrival time is set to 1, otherwise, the time delta is divided by 1,000 milliseconds. Similarly, the packet size is checked and if it is greater than 1,500 bytes, the packet size is set to 1. Otherwise, the packet size is normalized by dividing it by 1,500. The packet direction is set to 0 or 1 based on the first flow packet direction determined by the source IP and destination IP addresses. The protocol is set to 0 if it is a UDP packet, and to 1 if it is a TCP packet. The flows extraction algorithm is sketched in Figure 4. After the normalization, the flows are subdivided into M sub-flows of $n \times n$ packets each, as shown in Figure 5. The initial flow is composed of n packets [p1, p2, .., pn]. Each packet Pi is represented by 4 features [Si, Ti, Di, Pi], where Si stands for the packet size, Ti stands for inter-arrival time, Di stands for direction, and Pi stands for transport protocol. This resulted in a $n \times n \times 4$ vector for each sub-flow. At the same time, we saved each feature values independently (i.e. packet size for each 16 packets of the flow, inter-arrival time for each $n \times n$ packets of the flow, the protocol for each 16 packets of the flow and the direction for each $n \times n$ packets of the flow). In this case, the obtained vectors are $n \times n$ for each sub-flow.

Two data representation methods are considered for comparison. The first method is a previous work method considering the first packet of each flow as gray image, as shown in (see Figure 3.3a). Thus, we refer to this method as the gray method in the rest of this chapter. The gray method presents each byte

Level 1		Level 2		Level 3				Level 4		
				Messenger	Whats App	Hangout	Skype	Mobile	Laptop	
Inter-active	1061	Voice Call	338	130	128	80	108	54	76	
		Video Call	330					28	52	
				147	24	82	77	59	49	
		Texting	379	60	151	168	145	63	84	
		Gaming	14					37	45	
								33	34	
								57	3	
								116	52	
								17	128	
								10	4	
				Dropbox	MyDrive	OneDrive		Mobile	Laptop	
Bulk Data Transfer	2135	File upload	619	123	227	269		20	103	
		File Down-load	332				120	77	135	34
		System Update	184					186	83	
		Apps Down-load	1018					40	80	
							14	63		
							46	89		
							93	91		
								706	312	
Streaming	134	Video	69					Mobile	Laptop	Camera
		Audio	65					56	17	4
								55	10	
								D-Link Plug	D-Link Siren	Samsung Smart Plug
Transaction	282	Actuation	200					86	180	12
								D-Link Water Sensor	Samsung Motion Sensor	Samsung Multi Purpose Sensor
		Sensing	82					60	12	13

Table 3.1: Collected Data (Number of Flows)

of the first 784 bytes of each flow. The second method is our proposed method that extracts four features of the first $n \times n$ packets of each flow, as shown in (see Figure 3.3b). 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 chapter. The obtained images are shown in (see Figure 3.6). 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.

3.1.3 Classifier Model

The aim of this chapter is to compare our data representation method to a previous work method with 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 obtained 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

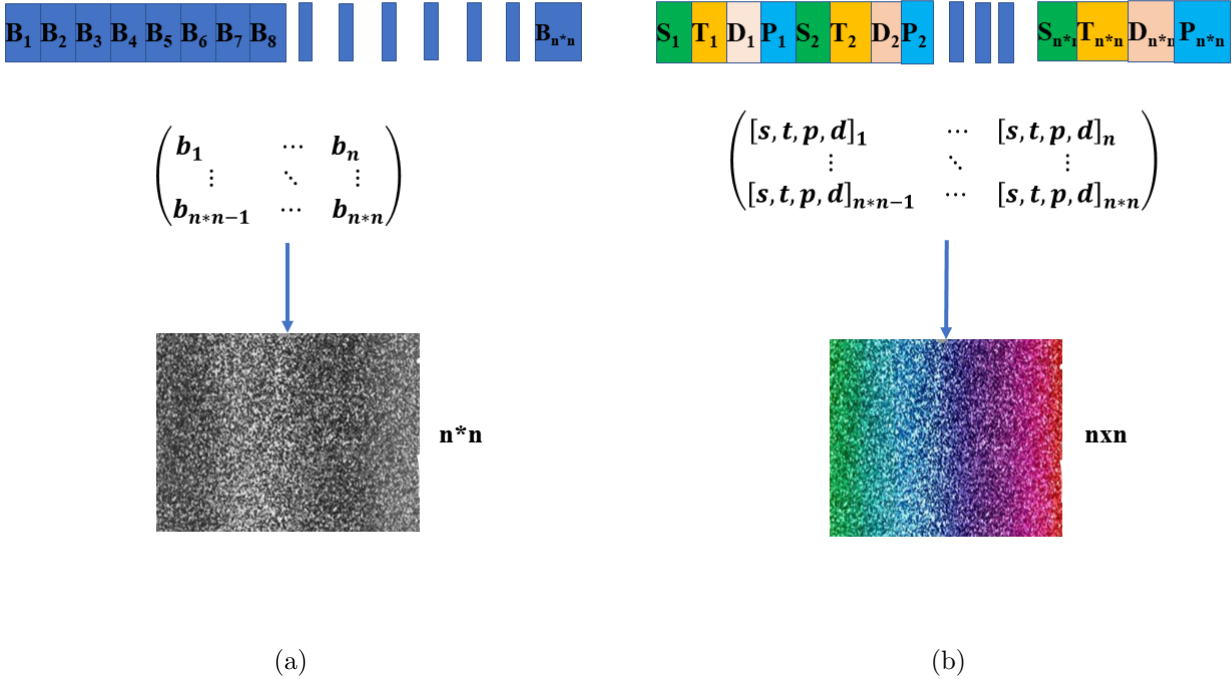


Figure 3.3: Data Representation: (a) Gray images and (b) RGBA images

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 3x3 convolutional filters applied at stride 1. We used 2x2 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 Figure 3.5.

3.2 Comparison Criteria and Methodology

To compare two data representation methods for ML based classification, different aspects have to be considered:

- First, the **classification performance** of the considered representation methods is evaluated at the different levels.
- Second, **features importance** should be analyzed. For traditional ML, 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

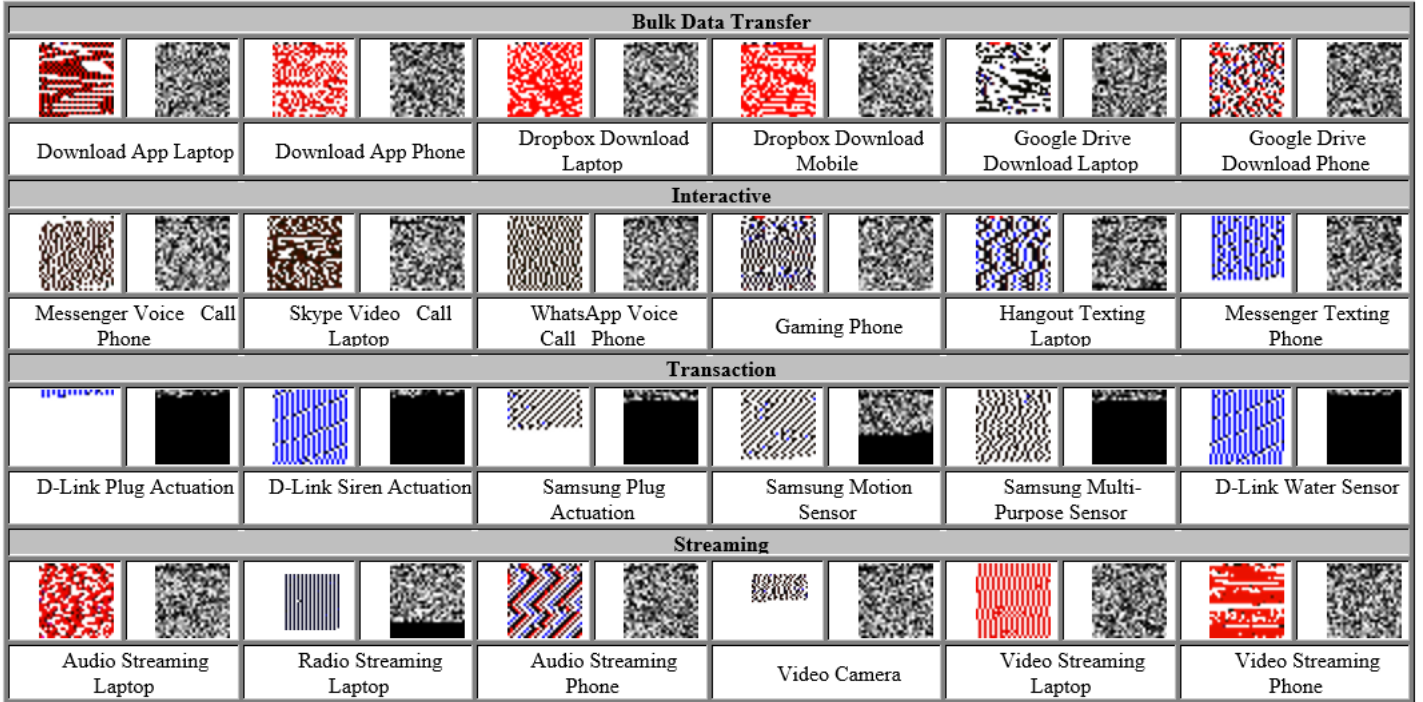


Figure 3.4: Data visualization (at left: RGBA method, and at right: Gray method)

for CNN, the features importance analysis is not tightly related to these measures.

- Third, **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 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 3.2, are considered with an RF classifier for comparison with a state-of-the-art method.

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>	The min packet size in the forward direction
<i>mean_pkt_size_fwd</i>	The mean packet size in the forward direction
<i>max_pkt_size_fwd</i>	The max packet size in the forward direction
<i>std_pkt_size_fwd</i>	The standard deviation packet size in the forward direction
<i>min_pkt_size_bck</i>	The min packet size in the backward direction
<i>mean_pkt_size_bck</i>	The mean size of packets in the backward direction
<i>max_pkt_size_bck</i>	The max packet size in the backward direction
<i>std_pkt_size_bck</i>	The standard deviation packet size in the backward direction
<i>total_size</i>	The total flow size
<i>min_pkt_size</i>	The min packet size in either direction
<i>mean_pkt_size</i>	The mean size of packets in either direction
<i>max_pkt_size</i>	The max packet size in either direction
<i>std_pkt_size</i>	The standard deviation packet size in either direction
<i>min_iat_fwd</i>	The minimum interarrival time in the forward direction
<i>mean_iat_fwd</i>	The mean interarrival time in the forward direction
<i>max_iat_fwd</i>	The maximum interarrival time in the forward direction
<i>std_iat_fwd</i>	The standard deviation interarrival time in the forward direction
<i>min_iat_bck</i>	The minimum interarrival time in the backward direction
<i>mean_iat_bck</i>	The mean interarrival time in the backward direction
<i>max_iat_bck</i>	The maximum interarrival time in the backward direction
<i>std_iat_bck</i>	The standard deviation interarrival time in the backward direction
<i>total_time</i>	The duration of the flow
<i>min_iat</i>	The minimum interarrival time in either direction
<i>mean_iat</i>	The mean interarrival time in either direction
<i>max_iat</i>	The maximum interarrival time in either direction
<i>std_iat</i>	The standard deviation interarrival time in either direction
<i>avg_pkt_fwd</i>	The average number of packets in the forward direction
<i>avg_bytes_fwd</i>	The average number of bytes in the forward direction
<i>avg_pkt_bck</i>	The average number of packets in the backward direction
<i>avg_bytes_bck</i>	The average number of bytes in the backward direction
<i>avg_iat_fwd</i>	The proportion of flow time in the forward direction to the total flow time
<i>avg_iat_bck</i>	The proportion of flow time in the backward direction to the total flow time

Table 3.2: Feature Set for statistical classification

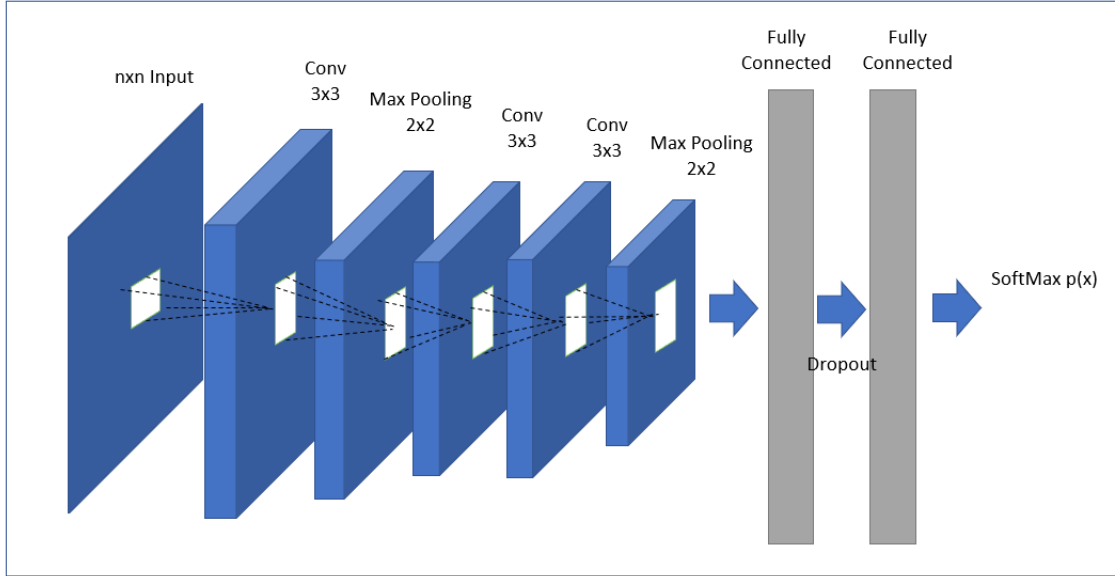


Figure 3.5: The used CNN architecture.

Algorithm 1 Performance test

- 1: **procedure** PERFORMANCE TEST
 - 2: **for** $node_i \in parent_nodes$ **do**
 - 3: train *model* on $train_data_i$
 - 4: test *model* on $test_data_i$
 - 5: **end for**
 - 6: **end procedure**
-

3.2.1 Performance Test

As shown in Algorithm 1, this test is performed at each parent node ($node_i$) for 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 *transaction*. 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*, *Hangout*. At level 4, a classifier is trained for each class 3 traffic 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.

3.2.2 Features Importance Test

In this section, we detailed 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.

Packet Level Representation:

Algorithm 2 Packet level features importance training

```

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

```

The Transmission Control Protocol/Internet Protocol (TCP/IP) packet consists of 25 fields as shown in Table 3.3. For the packet level representation, to analyze the importance of the packet fields, we run an experiment that anonymizes the fields sequentially. At each round, the field presenting the minimum accuracy when anonymized is chosen to be anonymized at the next rounds. This is done until anonymizing all the packet fields. The aim is to know what are the most important fields influencing 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.

Table 3.3: 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 3 Packet level features importance testing

```
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
```

Flow Level Representation:

Algorithm 4 Flow level features importance test

```
1: procedure TRAINING
2:   for feature ∈ flow_features do
3:     train model on feature_vector
4:     test model on feature_vector
5:   end for
6: end procedure
```

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 a $n \times n$ gray image applied to two images sizes 28x28 and 4x4.

3.2.3 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:

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
```

interactive, *bulk data transfer*, *streaming*, and *transaction*. Consequently, the robustness test aims at removing at a time one level 2 class and see 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*.

3.2.4 Features Robustness Test

In this section, we investigate the robustness of the features 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 the effect of traffic encapsulation by VPN which present traffic encryption and meta-data modification such as the port numbers, IP addresses, etc. On the other hand, we aim to investigate the anonymization effect on both methods, given that TOR affects also the statistical features; for example, the packet length. In addition, we consider mutation techniques that aim at anonymizing the main traffic characteristics like 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 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. We mean by packet size anonymization to pad all the packet sizes to the Maximum Transmission Unit (MTU) and to set the packets inter-arrival time to a fixed time interval (i.e. one second).

3.3 Experimentation Results

In this section, we present the comparison tests results. Note that these tests were performed on a Linux machine (Ubuntu 14.04 LTS) with 16 GB RAM and Intel core i7 processor. TFLearn [356] was used as a high-level API for the tensorflow [357] Python library for DL implementation. It should be noted that in all tests cross validation is applied with 4 folds and data is divided into 60% for training, 20% for validation, and 20% for testing. For the CNN classifier, the convolutional layers were optimized using the Adam optimizer, and the Rectifier Linear Unit (ReLU) function is used for activation. Dropout probability is chosen to be 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 RF classifier was implemented using the scikit-learn python library [358], with grid search to optimize the number of trees with [1,100] as the range for search.

3.3.1 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 Figure 3.6, 3.7, and 3.8. It is clear from the results that the RGBA28x28 representation method achieves overall better results at the different classification levels. At level 1, as shown in Figure 3.6, RGBA28x28 and RGBA4x4 present better results than the Gray method. RGBA8x8 and RGBA4x4 present better performance in terms of accuracy, precision, recall. In terms of accuracy, RGBA28x28 presents the highest value with 95.84%, followed by RGBA4x4 with 93.49%, and then Gray with 92.18%. It can be noticed also that RGBA4x4 presents better results than RGBA28x28 in some cases. For example, at level 3, RGBA4x4 gives better accuracy (82.72%) than both the RGBA28x28 and Gray methods (79.59% and 70.4%). 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 presenting different flows from different devices. Furthermore, the results show that the RF method presents better accuracy at the different levels when considering the first 28x28 packets (RF28x28) or the first 4x4 packets (RF4x4).



Figure 3.6: Performance test results (Level 1)

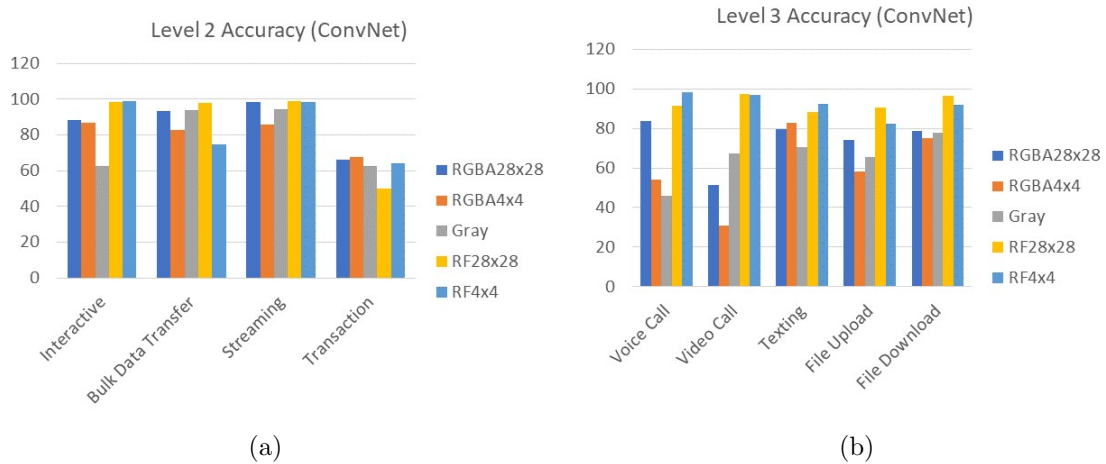


Figure 3.7: Performance test results in terms of accuracy: (a) for Level 2 and (b) for Level 3

3.3.2 Features Importance Test Results

Packet Level Representation:

For the Gray method, the features importance test results are illustrated in Figure 3.9. Figure 3.9a presents 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 makes the accuracy drop by more than 10%. In the testing case, the drop of accuracy is more pronounced, and thus, as shown in (see Figure 3.9b), anonymizing only the data makes the accuracy drop by 10%. Moreover, anonymizing data, source MAC address, and ACK number makes the accuracy drop 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

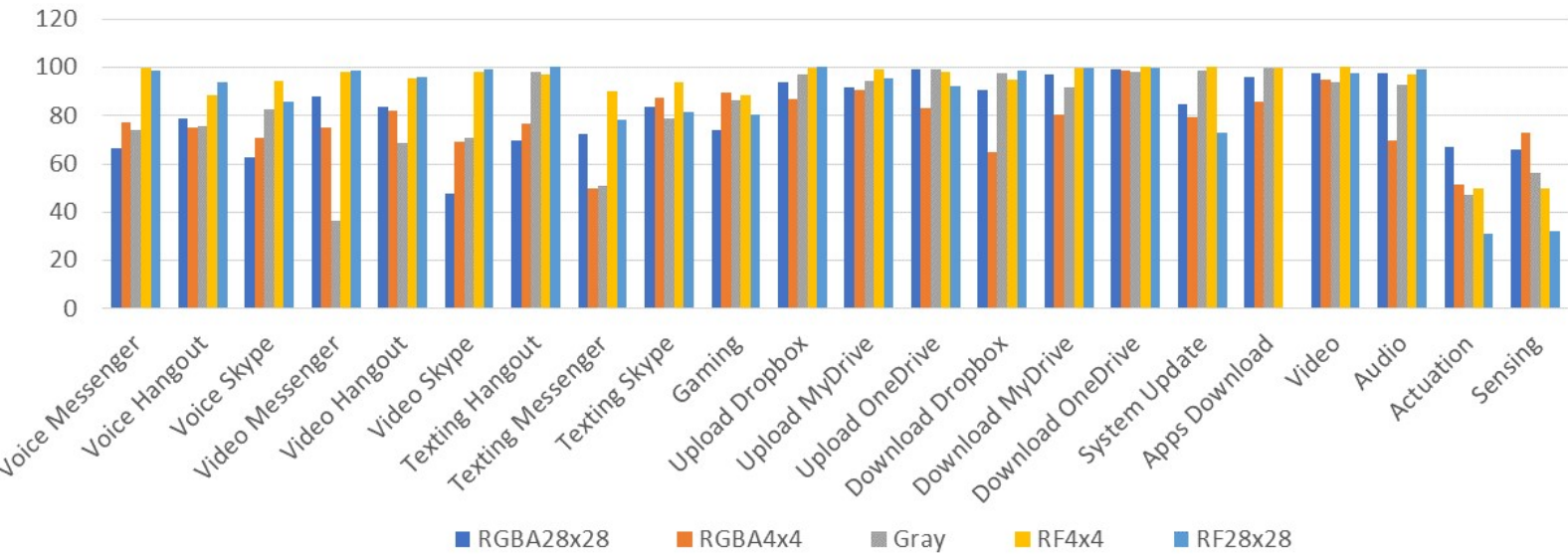


Figure 3.8: Performance test results in terms of accuracy (Level 4)

important feature, as shown in (see Figure 3.9c). However, in the second case, the anonymization of the source MAC and destination MAC addresses makes the accuracy drop noticeably to 15%, as shown in (see Figure 3.9d). Finally, for the VPN data, anonymizing the IP addresses makes the accuracy drop to 60% in the first case. However, in the testing case, the accuracy drop was more pronounced for the same features.

Flow Level Representation:

For the RGBA method, The features importance test results are included in Figure 3.10a and 3.10b. For the 28x28 images, it can be noticed that packet size is the most important feature. This can be observed given that the accuracy, when using the packet sizes of the first 28x28 packets of a flow, is the highest relative to the other features. Similarly, for the RGBA4x4 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.

It can be noticed that the RGBA method, including statistical features, makes the trained model generalizable when the data is collected from different network setups. Moreover, the accuracy results obtained for VPN and TOR data show the immunity toward traffic anonymization. Moreover, the direction feature reflecting a behavioral pattern of the traffic can serve for classification if some mutation techniques, modifying the packet size or inter-arrival time, are applied. However, for the gray method, anonymizing some connection/session variables make the accuracy drop noticeably. This indicates that the learned model is data and

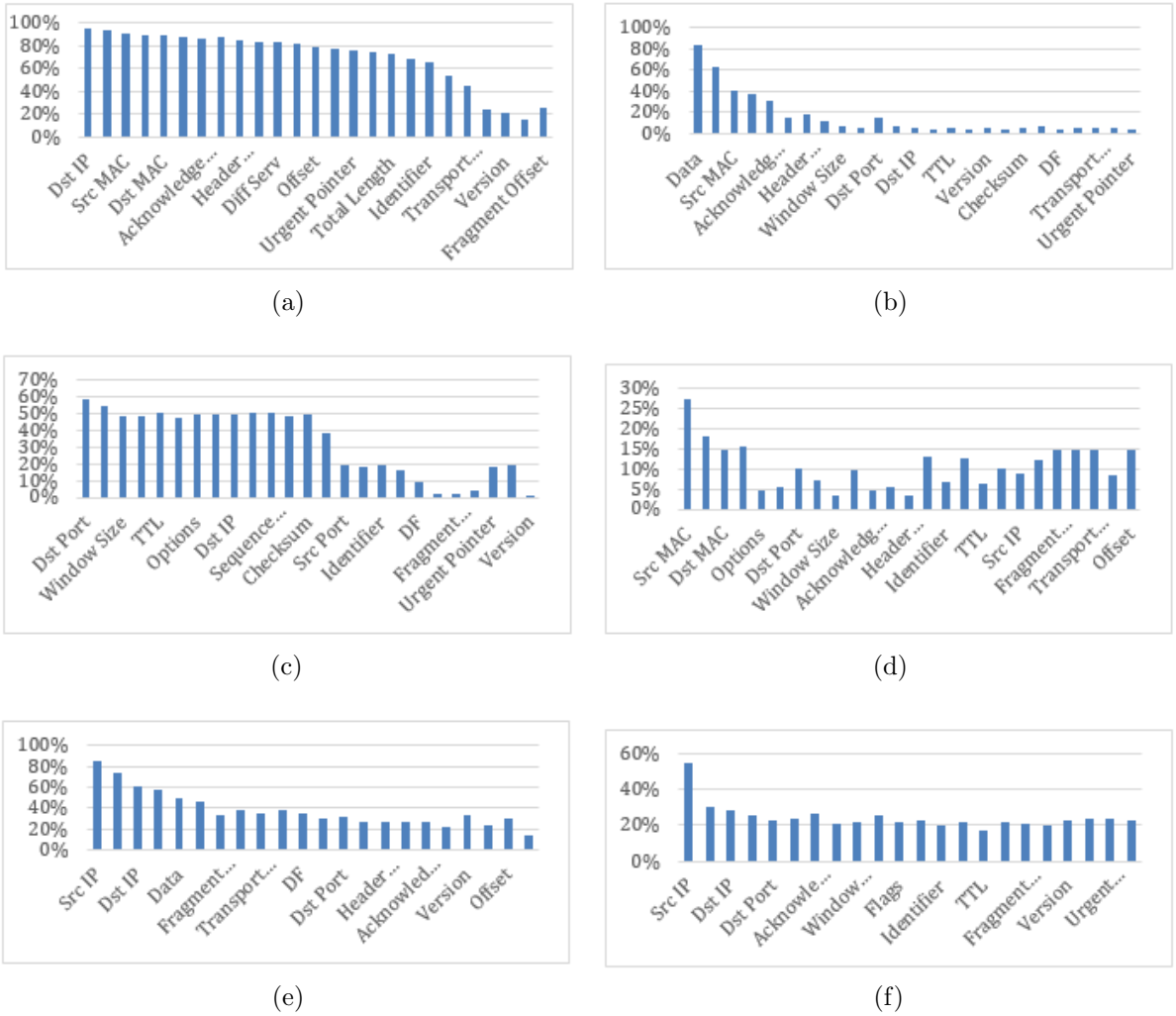


Figure 3.9: Gray method features importance results in terms of accuracy: (a), (c), and (e) for anonymization at training and testing phases using our data, TOR , and VPN data-sets respectively; (b), (d), and (f) for anonymization at the testing phase using our data, Tor, and VPN data-sets respectively.

network dependent and cannot be generalized.

3.3.3 Model Robustness Test Results

The features robustness test results, shown in Figure 3.11, present the accuracy results of the robustness test for the level 1 classes. It is clear that RGBA28x28 and sometimes RGBA4x4 present better results than the Gray method. Similarly, in Figure 3.12, the results of level 2 robustness tests are included. In these tests, for each level 2 class, we remove one of its sub-classes to see if the level 2 classifier

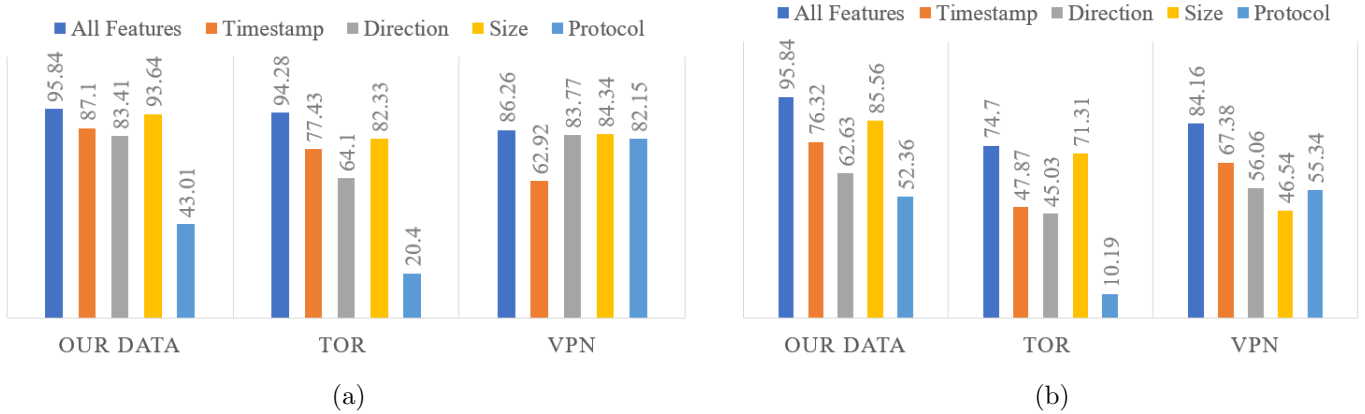


Figure 3.10: RGB method features importance results in terms of accuracy

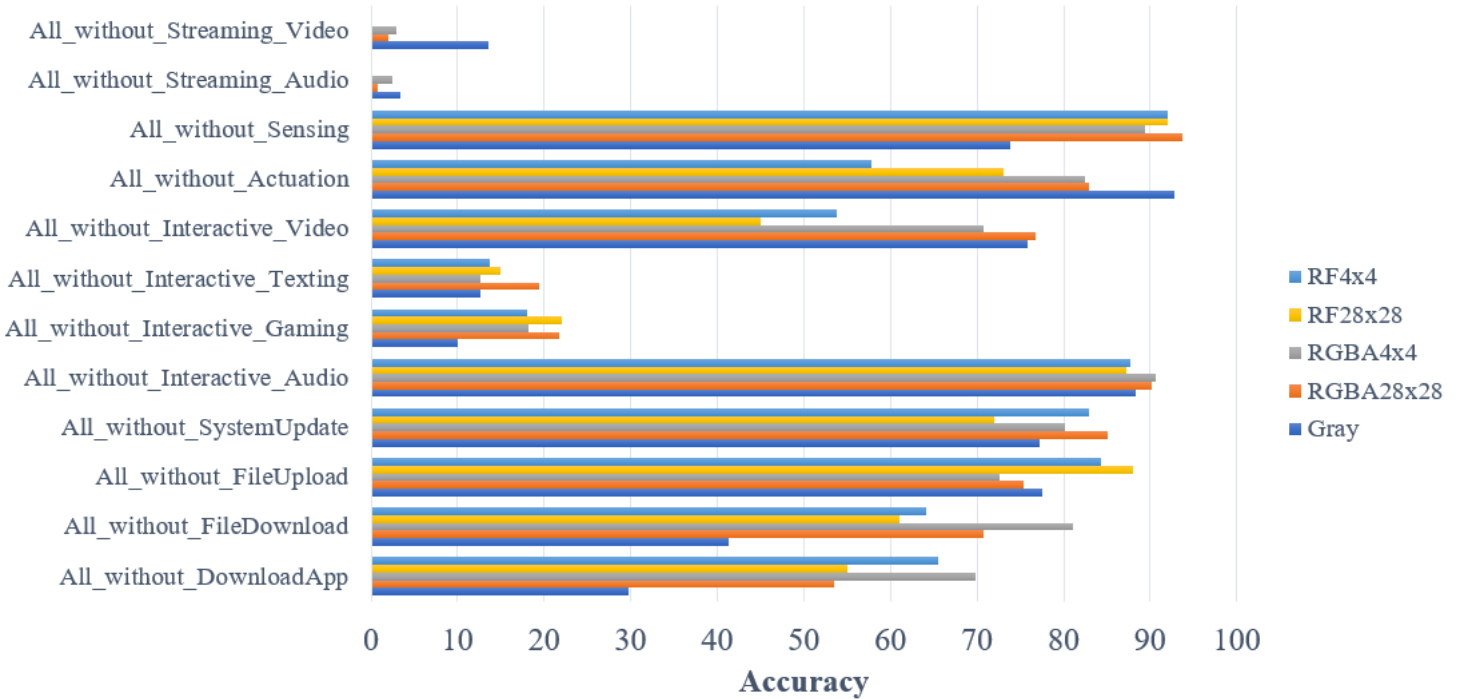


Figure 3.11: Robustness test results in terms of accuracy (Level 1)

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 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

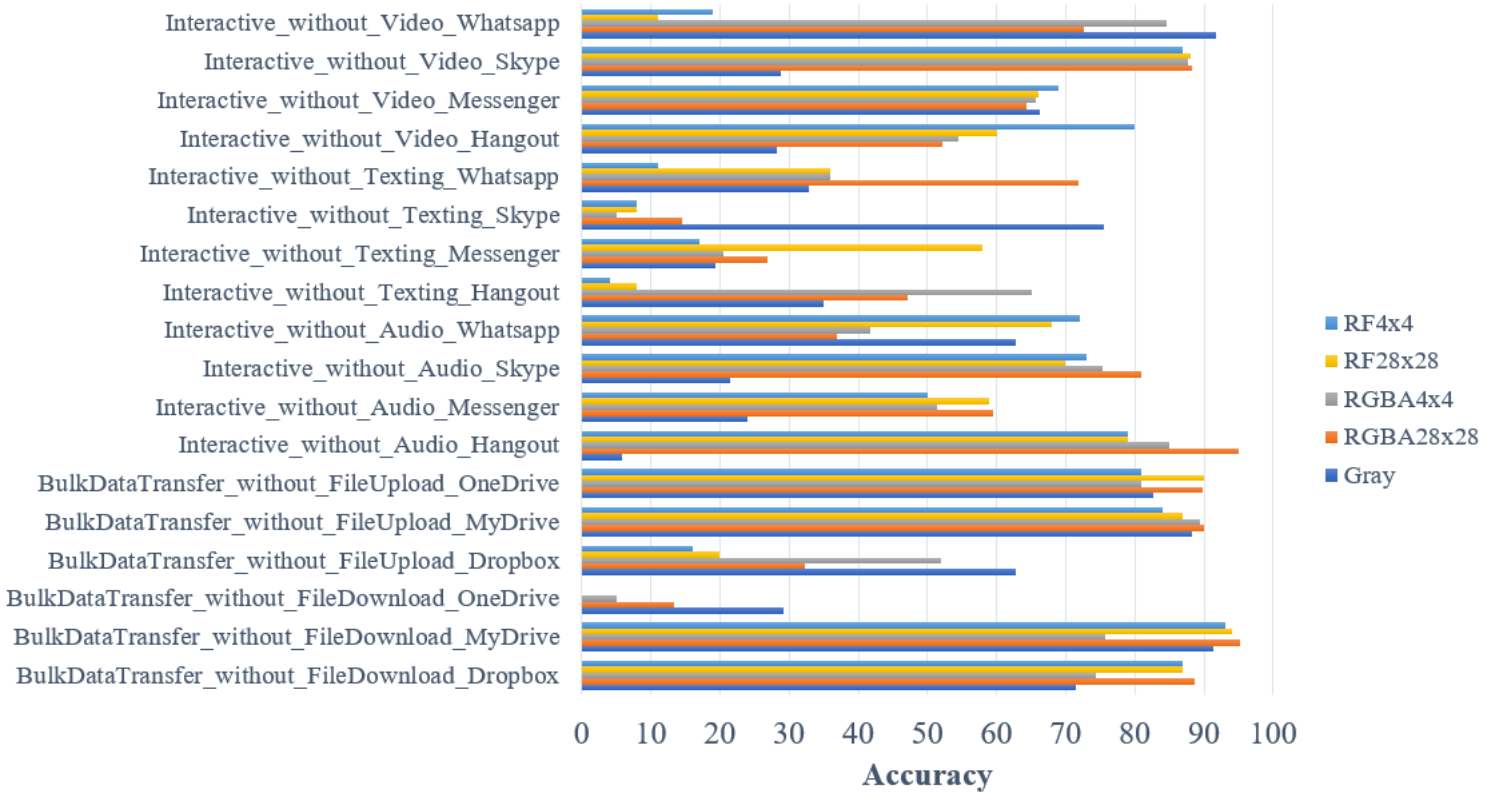


Figure 3.12: Robustness test results in terms of accuracy (Level 2)

the RF method with statistical features in most cases.

3.3.4 Features Robustness Test Results

For TOR and VPN traffic, the results presented in Table3.4 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% as accuracy on VPN data-set). Moreover, the RGBA method gives promising results on TOR traffic (94% as accuracy). Moreover, it can be noticed that 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 Figure3.13, it is clear that the RF based method performance degrades noticeably when the sizes and/or inter-arrival times of the packets are mutated. This indicates that the statistical features are prone the 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,

	RGBA28x28	RGBA4x4	Gray	RF28x28	RF4x4
VPN	86.26%	84.16%	78.68%	93.96%	88.34%
TOR	94.28%	74.7%	64.78%	97.92%	88.13%

Table 3.4: Accuracy Results (in%) with the TOR and VPN data-sets

the anonymization of the packets sizes and/or inter-arrival times test is performed to our data-set at all levels. The results for the different levels are included in Figure 3.14. It can be noticed that the drop in accuracy for all levels is more pronounced for the RF based method than 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.

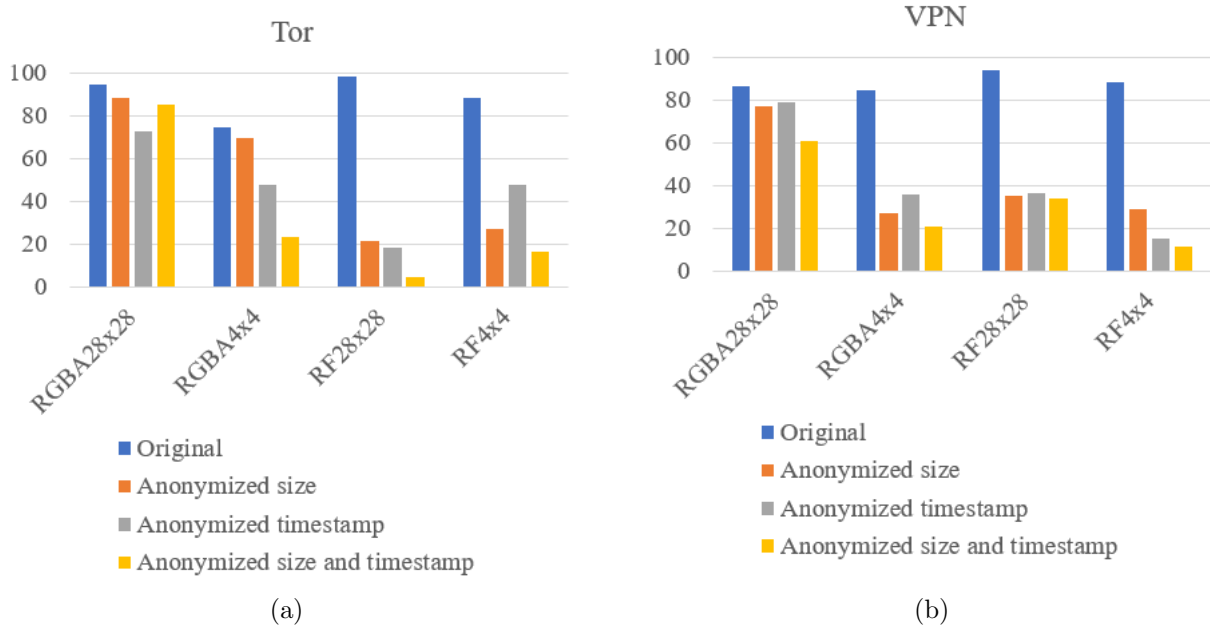
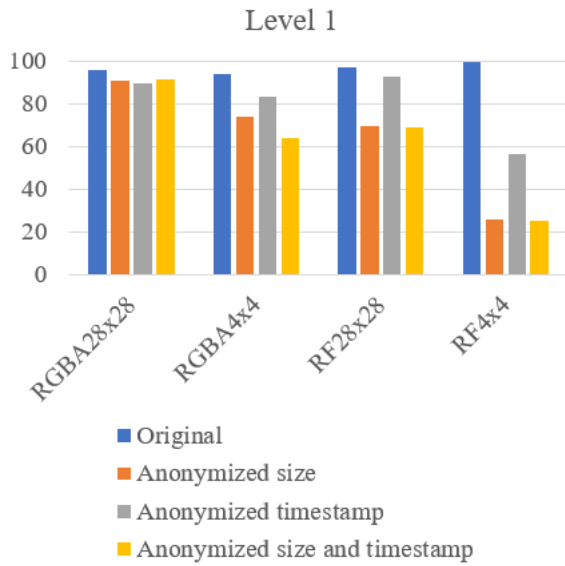


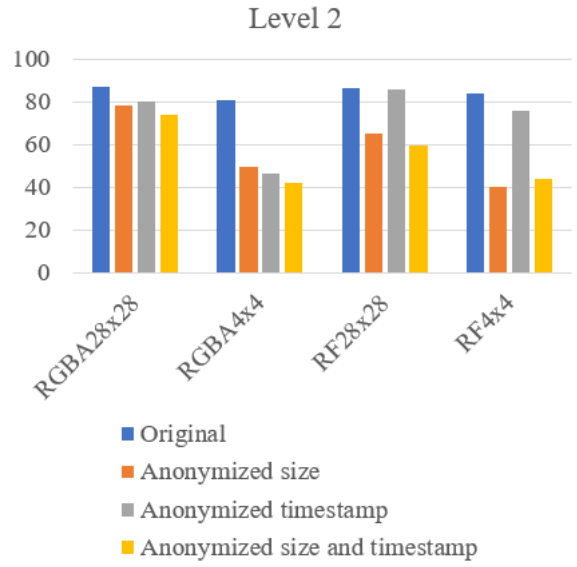
Figure 3.13: Anonymization results in terms of accuracy considering the TOR and VPN data-sets

3.4 Discussions

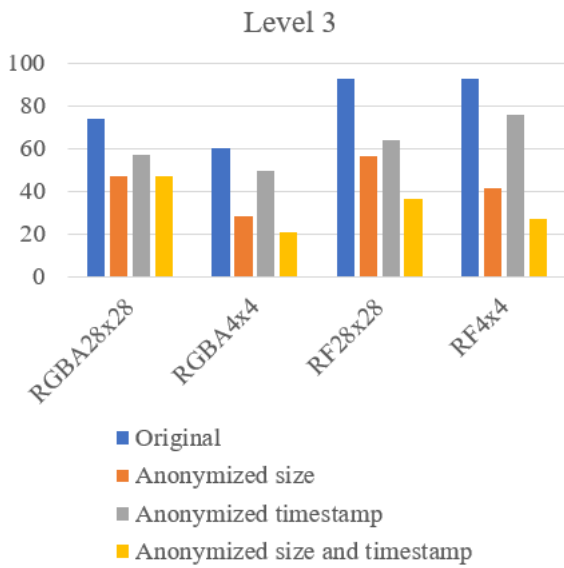
Image recognition is a well-established CNN application. Thus, applying CNN for traffic classification calls for new representation methods of the traffic as images.



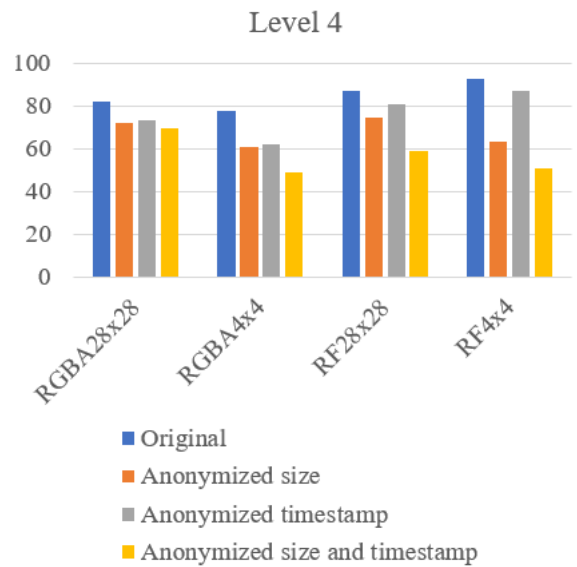
(a)



(b)



(c)



(d)

Figure 3.14: Features robustness test results in terms of accuracy for all levels considering the RGBA28x28, RGBA4x4, RF28x28, and RF4x4: a) level 1, b) level 2, c) level 3, and d) level 4.

In this chapter, 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 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 permits us to design new tests that aim for evaluating 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 (level 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 present visually discernable patterns, but the achieved accuracy at different levels are comparable to those obtained with the RGBA method. However, the features importance test show that the anonymization of certain packet fields, that are network connection dependent, result in significantly altering the classification accuracy.

Finally, the anonymization test results show that the RGBA method presents better accuracy than the Gray one, when trained on tunneled (VPN) or anonymized traffic (TOR). This gives a clear indication that statistical features can present 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.

3.5 Conclusion

An essential benefit of DL over the traditional Machine Learning methods is its representation learning capability. More specifically, CNN were shown to be very effective for images classification. Consequently, representing traffic as images have been considered in the last two years. One direction was to represent the first packet as gray image, and thus traffic raw data is considered for classification. Another method consists of extracting four features from the first $n \times n$ packets per flow. In this chapter, we compared these two representation methods. Three types of tests were performed: features importance test, model robustness, and anonymization test. The results show that features should be carefully selected to not be altered by changing the network environment or by possible obfuscation

techniques. In this 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 avoiding misclassification.

Chapter 4

Detecting Mutation

Mutation techniques alter the traffic statistical features, making it very challenging to know the original traffic type. Moreover, the mutation techniques might change the packets characteristics while maintaining the flow statistical features unchanged. In this case, the detection of abnormal traffic can be evaded. Recently, Deep Learning (DL) has acquired a lot of attention due to its representation learning capabilities. Using our proposed data representation, we propose an unsupervised DL model to detect abnormal traffic and de-anonymize the mutated one. Generative DL architectures, namely Autoencoders (AE) and Generative Adversarial Networks (GAN), have been applied mainly in the computer vision domain to detect abnormality and to denoise images. AE is a DL architecture for extracting data representation, and GAN has the capability to generate fake data samples to enhance the discrimination between real and fake data. In this chapter, we combine AE and GAN to detect abnormal traffic and de-anonymize the mutated one. The proposed architecture consists of an encoder, a decoder, and a discriminator. The encoder-decoder pair form a denoising AE responsible for learning the original data representation and to denoise the mutated one. In parallel, the discriminator is trained to differentiate between the mutated traffic (abnormal) and the denoised traffic (normal). The training of the proposed model relies on data collected from real IoT devices and IoT attacks. The testing results show the robustness of the proposed method to detect mutated traffic and to recover the original one. Note that the proposed model is not limited to IoT traffic and can be applied to any type of network traffic.

4.1 Proposed Abnormal Traffic Detection

In this section, we detail our proposed DL model, the attack model, and the traffic representation method.

4.1.1 Attack Model

Our attack model consists of an attacker trying to modify the packet size (padding) and IAT (shaping), in such a way to hide any information that serves for attack detection or traffic classification. The mutations are therefore of two types, padding and shaping [28, 29], as shown in Figure 4.1. In the following, we summarized the packet padding techniques listed in [359, 360, 361], with s being the packet original size and $m(s)$ being the packet size after mutation.

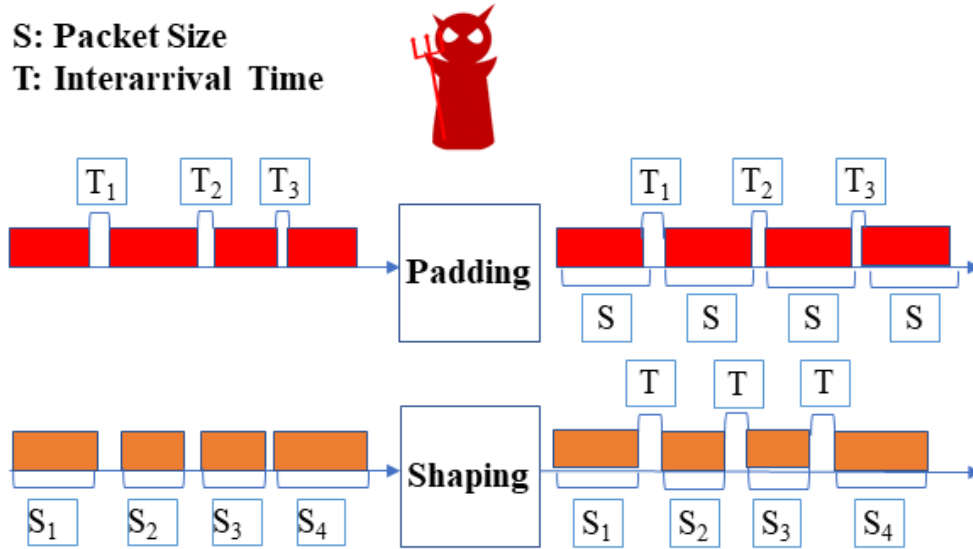


Figure 4.1: Attack model

1. **Padding to the Maximum Transmission Unit (MTU):** this technique consists of padding all the flow packets to the same size, which is the MTU. In this case, the mutation can be expressed by $m(s) = MTU$.
2. **Linear padding:** this technique consists of linearly padding the flow packets sizes. In this case, the mutation equation can be expressed as follows: $m(s) = \lceil s/c \rceil * c$, where c is a parameter to choose, and $\lceil s/c \rceil$ is the ceiling of s/c .
3. **Exponential padding:** this technique consists of padding the packet size in an exponential manner. The mutation can be expressed by the following equation: $m(s) = \min(2^{\log_2(s)}, MTU)$.
4. **Elephants and mice padding:** this technique consists of padding the packet to a certain size c (mice), if the original packet size is less than c . If not, the packet size is padded to the MTU (elephant).

5. **Random padding:** this technique consists of padding the packet randomly to a size chosen randomly from the interval $([s, MTU])$. In this case, the mutation function can be expressed as following: $m(s) = RAND([s, MTU])$.
6. **Probabilistic padding:** this technique assumes that the packet sizes follow a normal distribution. In this case, the mutated size is chosen randomly based on a normal distribution, where the mean (μ) and standard deviation (σ) are computed considering the original traffic packet sizes. In this case, $m(s) = GAUSS(\mu, \sigma)$ [361].

For the IAT shaping techniques, we list in the following the ones included in [362] in addition to a new technique proposed in [361]:

7. **Constant IAT:** this technique consists of sending the packets at a fixed IAT.
8. **Variable IAT:** this technique consists of sending the packets at a variable interval of time randomly chosen from the interval $[I_1, I_2]$.
9. **Probabilistic IAT shaping:** this technique assumes that the IAT follows a normal distribution. Thus, the packets are transmitted at an interval chosen randomly based on a normal distribution, where the mean and standard deviation are computed based on the original traffic packets IAT.

The last method, described in [363], combines shaping and padding.

10. **Fixed size and fixed IAT:** this method consists of padding the size of all the flow packets to the MTU and sending all the packets at a fixed time interval.

This model considers two types of adversaries, including:

- **Malicious adversary:** this type of attackers aims at mutating the attack traffic to evade intrusion detection.
- **Benign adversary:** this type of attackers aims at hiding the traffic characteristics to protect the user privacy.

4.1.2 Deep Learning Model

Our proposed model, illustrated in Figure 4.2, consists of two parts: a denoising AE and a discriminator. The denoising AE consists of an encoder and a decoder.

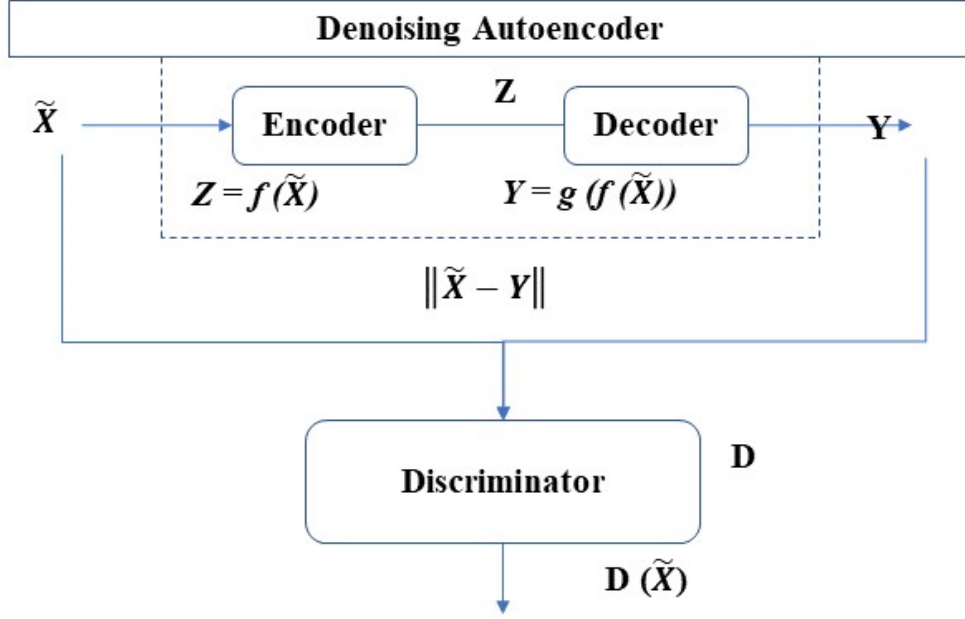


Figure 4.2: Proposed model

The encoder aims at estimating the function f , that maps the input space of X to the latent space of the latent variable Z , where Z is a compressed representation of X . The denoising AE is fed with a dataset containing the mutated version \tilde{x} of the initial data X . The AE aims at minimizing the reconstruction error $L(X, \tilde{X})$. In our case, the loss function is the MSE function, $L(X, \tilde{X}) = (X - g(f(\tilde{X})))^2$, where X is the original data and $g(f(\tilde{X}))$ is the reconstructed data by considering the mutated data as input. This MSE represents the difference between the original and reconstructed flow vectors. The discriminator aims at differentiating between normal and abnormal traffic. In fact, the discriminator will be trained on classifying the mutated data (i.e. the mutated input data) as abnormal and the reconstructed data (i.e. the decoder output) as normal. The output function of the discriminator is a sigmoid function $p(y = y^{(j)}/x) = 1/(1 + e^{-y^{(j)}})$, where $y^{(j)}$ is the output class that can take the two values: 0 for $j = 0$ and 1 for $j = 1$, and the loss function is a cross entropy function $L_D = 1/m \sum_{j=0}^1 y^{(j)} \log(y^{(j)}) + (1 - y^{(j)}) \log(1 - y^{(j)})$.

4.1.3 Proposed Model Workflow

After training the model, the proposed scheme workflow, illustrated in Figure 4.3, consists of: 1) passing the traffic to the discriminator; 2) if the traffic is normal, it is passed to the classifier of normal traffic; 3) If not, it is passed to the denoiser. 4) After denoising, it is passed again to the discriminator. 5) If a normal traffic is detected, it is passed to the classifier; 6) if not, the traffic is detected as abnormal,

so it might be obfuscated or attack traffic. The same workflow is repeated for the abnormal traffic to know the attack type or detect an unknown attack traffic. In fact, two cases are considered. First, when training the model on normal traffic, the aim is to detect attack traffic and the mutated one as abnormal. In this case, the denoising aims at recovering the normal traffic for classification. However, when training the model with attack traffic, at the testing phase the aim is to detect unknown attacks and to denoise the mutated attack traffic to know the exact attack type. Note that the classifier will be trained in a supervised mode to classify the traffic based on the IoT device type in the normal traffic case and based on the attack type in the attack traffic case.

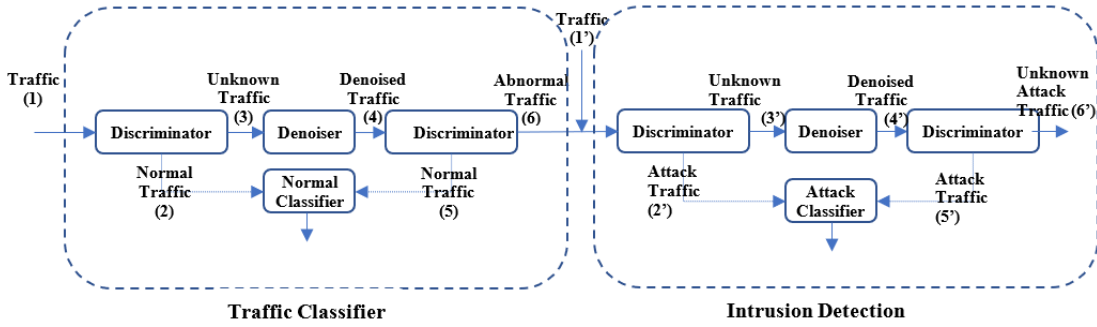


Figure 4.3: Proposed scheme workflow

4.1.4 Data Representation

In our case, the data consists of collected network traffic. This traffic is filtered by flows, where a flow is defined as being the set of packets having the same: source IP address, source port number, destination IP address, destination port number, and protocol (TCP or UDP). For each packet, we extract three features: size (s), IAT (t), and direction (d). For each flow, we consider the first 4×4 packets in either direction. In fact, as shown in Figure 4.3, the extracted features can be visualized in 4×4 RGB images, where the i^{th} pixel RGB coordinates are represented by the i^{th} packet features $[s, t, d]$. Thus, every feature represents a color channel. In our case, R is given for the size, G is given for the inter-arrival time, and B is given for the direction. This data representation is explained in detail in chapter 3 [353].

4.2 Implementation

In this section, we detail the data collection and preprocessing, the model implementation, and the evaluation results.

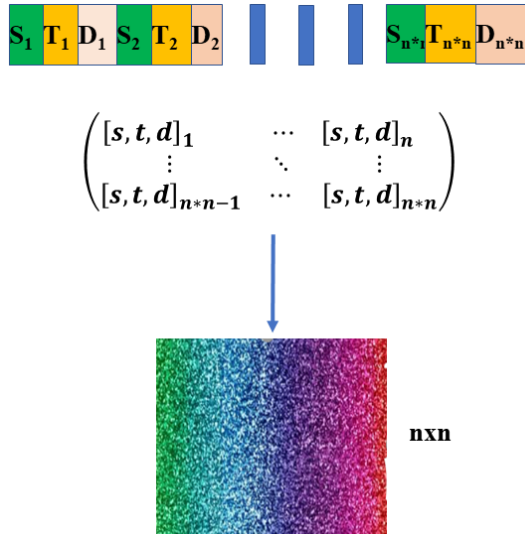


Figure 4.4: Data Representation

4.2.1 Data Collection and Preprocessing

For collecting normal IoT traffic, we used a set of IoT devices, including Wi-Fi-enabled devices and hub-connected devices. The Wi-Fi devices are: D-Link HD 180-Degree Wi-Fi Camera DCS-8200LH, D-Link Wi-Fi Smart Plug DSP-W215, D-Link Wi-Fi Siren DCH-S220, and D-Link Wi-Fi Water Sensor DCH-S160, while the hub-connected devices are components of the Samsung SmartThings Home Monitoring Kit, including a motion sensor, a multi-purpose sensor, and a smart plug. These devices were installed in a home environment and were left to function normally. The Wi-Fi enabled devices are routed through a laptop to the Internet. A bridge is created at this laptop to forward the incoming traffic to the Ethernet interface connected to the home gateway. For the hub-connected devices, the hub is connected to the laptop by Ethernet and this laptop is configured to forward the incoming traffic to the wireless interface connected to the home gateway. In both cases, the traffic is collected at the laptop. One day of traffic for each device was considered for training and one day of traffic was used for testing.

For IoT attack traffic training data, we consider the dataset collected in [364], this dataset consists of multiple PCAP files for each type of attack. We choose one file for each type of attack for training and one file for testing.

To preprocess the data and extract the flows, the dpkt Python library was used [365]. In total, for training, we have 10320 flows of normal traffic and 3308 flows of attack traffic. For testing, we have 258 flows of normal traffic, 350 flows of attack traffic, and 2000 flows of (unknown) attack traffic. The normal traffic is categorized into five classes based on the device model while the attack traffic is categorized into three classes: data theft, denial of service and scanning.

4.2.2 Model Implementation and Training

The proposed model was implemented in Python using the tensorflow library [366]. The encoder, decoder and the discriminator consist of a 2-layer fully connected neural network with 1000 neurons each. The output layer of the decoder and the discriminator is a sigmoid layer, with the difference that the decoder output is of the same dimension of the input; however, the discriminator output is of dimension one. The model is trained with 100 epochs and a batch size of 1, the learning rate is 10^{-3} , and the momentum decay is 0.9 (beta1). The Rectified Linear Unit (ReLU) was used as the activation function for all hidden layers, and the weights are optimized using the Adam optimizer. For generating mutated data, we implement the mutation techniques listed in section III-B ((1) (10)). The training data is randomly mutated, where each data sample is uniformly mutated to one of the 10 mutation techniques.

For testing the effectiveness of the denoising process on the classification accuracy, we implement a CNN classifier with three convolutional layers, two max pooling layers, and two fully connected layers with one dropout layer with 50% dropout probability. Similarly, ReLU is applied for activation in the hidden layers, and the Adam algorithm is used for optimization with a learning rate of 10^{-3} . In addition, we apply cross validation for the classifier training with 4 folds. The performance metric used to evaluate the classifier is the accuracy, which is the ratio of the correctly classified samples to the total number of samples.

4.2.3 Evaluation and Results

For each of the experiments (1) (10), the corresponding mutation technique is applied to the testing data. First, the traffic is passed to the discriminator. Then, it is passed to the denoising AE. Furthermore, the mutated and denoised traffic are passed to a classifier, that classifies the flow based on the device model for normal traffic and based on the attack type for attack traffic.

Samples of the mutated traffic and resulted images after denoising are included in Table 4.5. It is visually clear from the included images that the most the denoiser succeeds in learning the original data representation disregarding the mutation level. The difference between the denoised images and the original ones shows that the model does not overfit the data, however it learns the representation and the noise pattern.

For the normal traffic, the denoiser succeeds in recovering flows very similar to the original ones for most of the mutation techniques, except for the mutations (8) and (9). Similarly, for the attack traffic, the denoiser recovers the main flow representation, except for the mutation (2). In Table 4.1, the MSE between the original data and the mutated one (mutation degradation), the reconstruction loss, and the abnormality detection rate are reported for the normal traffic and the attack traffic.

Mutation method	Normal			Attack		
	Original	Mutated	After denoising	Original	Mutated	After denoising
(1)						
(2)						
(3)						
(4)						
(5)						
(6)						
(7)						
(8)						
(9)						
(10)						

Figure 4.5: Visualized images representing network traffic

The results present a high detection rate for the different mutation techniques, except for the mutation techniques (4), (6), and (9). This can be explained by the fact that (6) and (9) are normal distribution based mutations. In these cases, the main traffic characteristics will remain unchanged and this will harden the differentiation between the original and mutated traffic. (4) is a mice-elephant mutation of the packet sizes. However, in our case (i.e. IoT traffic), most of the packets are of small size and therefore the mutation (4) will have limited affect on the traffic characteristics. Moreover, it can be noticed that the MSE between the recovered data and the original one (reconstruction loss) is lower than the MSE between the mutated data and the original one (mutation degradation) in

Mutation Tech-	Autoencoder Loss		Mutation Loss		Discrimination Rate	
	Normal	Attack	Normal	Attack	Normal	Attack
(1)	0.042	0.0357	0.2713	0.1698	100%	100%
(2)	0.03218	0.1569	0.0171	0.082	100%	100%
(3)	0.0149	0.1569	0.0009	0.0065	88.99%	83.13%
(4)	0.0543	0.0463	0.2704	0.1696	50%	49.94%
(5)	0.067	0.0519	0.0171	0.082	100%	100%
(6)	0.0621	0.0703	0.0129	0.0411	49%	49.88%
(7)	0.0378	0.1569	0.2974	0.3245	100%	100%
(8)	0.0595	0.0257	0.09878	0.1016	99.02%	100%
(9)	0.1728	0.0303	0.2825	0.0012	50%	41.59%
(10)	0.0369	0.0519	0.5687	0.4943	100%	100%

Table 4.1: Testing evaluation results

most of the cases. This means that the denoising process decreases the degradation effect by reconstructing a version of the data that is closer to the original one.

Table 4.2 presents the accuracy of the classification based on the traffic label: device model for normal traffic and attack type for attack traffic. The mutation techniques (1), (4), (7), and (10) affect the accuracy and misleads the classification noticeably in the normal traffic case; however, after denoising, the accuracy increases. However, for the techniques (5), (6), (8), and (9), the denoiser fails to reconstruct the original traffic. This is due to the fact that the randomness will create a denoised traffic of random type. Moreover, for the mutation technique (2), the mutation is linear and this does not affect the CNN classifier accuracy, being immune to the linear degradation of the image. However, for statistical based machine learning methods, the mutation techniques (2), (5), and (6) affect noticeably the classification accuracy. To see the effect of the mutation on the statistical based classification, we include the results of the mutated and denoised traffic statistical based classification in Table 4.2. The statistical classification uses a subset of the Moore features (see Table 3.2) and Random Forest (RF) as classifier. It can be noticed that overall our representation method with CNN classifier outperforms the RF method before and after denoising in the normal traffic case. However, in the attack traffic case, our method gives better results after denoising.

4.3 Discussions

The results show that unsupervised DL architectures are very powerful in learning the data representation. AE is a well-known DL generative model, that is highly effective in extracting compressed representation from image-type data.

Mutation Tech- nique	Before Denoising				After Denoising			
	Normal		Attack		Normal		Attack	
	CNN	RF	CNN	RF	CNN	RF	CNN	RF
(1)	30.14%	54.47%	51.55%	80.84%	65.19%	38.78%	53.1%	37.05%
(2)	93.13%	25%	36.27%	36.24%	79.16%	68.68%	33.41%	33.41%
(3)	99.01%	87.19%	76.25%	90.33%	87.99%	78.79%	33.41%	33.41%
(4)	29.9%	55.2%	51.55%	80.1%	67.4%	40.68%	43.79%	42.57%
(5)	93.13%	25.06%	36.99%	36.21%	20.09%	20.09%	63.12%	38.21%
(6)	95.09%	25.06%	56.55%	62.94%	56.12%	33.88%	36.99%	42.69%
(7)	75.49%	85.6%	33.17%	19.63%	77.69%	55.39%	33.41%	33.41%
(8)	78.18%	81.92%	33.15%	15.96%	62.99%	41.85%	62.05%	38.48%
(9)	80.14%	82.35%	34.24%	39.64%	20.09%	20.09%	62.76%	30.31%
(10)	19.6%	28.18%	33.17%	25.65%	78.43%	49.14%	60.6%	43.22%

Table 4.2: Classification Results

Consequently, after representing network traffic as images, we applied AE to extract the representation patterns from IoT traffic. Moreover, the capability of AE to denoise images is used to overcome the mutation technique challenges. The results show the effectiveness of the proposed method to recover the original traffic representation for different levels of mutation, some of which are rather severe (e.g. fixed packet size and IAT). In fact, the considered mutation techniques cause any statistical classifier to fail. However, one limitation of this work is that we assume that we know ahead of time the mutation technique used by the attacker, so we can decide to choose to apply the classifier before or after denoising. To address this limitation, we can train a classifier on classifying the different mutation techniques. The implementation results show a high classification accuracy (>80%).

4.4 Conclusion

New traffic obfuscation techniques have been developed to thwart classification and avoid detection in the sake of user privacy. These techniques have been employed by attackers to mount their attacks without being detected. While the obfuscation techniques might be detected by behavioral or statistical ML techniques, the recovery of the initial traffic is unfeasible. In this chapter, inspired from a promising DL application, which is image denoising, we transform the traffic to images then combine two well-established DL architectures (AE and GAN) to reconstruct the original traffic and detect abnormal one. The test results show the effectiveness and robustness of the proposed model to detect abnormal traffic in all its variants. After detecting mutated traffic, the challenge becomes harder when morphing is employed to mislead the classifier. Even though the

recovery of morphed traffic is impossible, can generative adversarial DL be used for morphing detection?

Chapter 5

Detecting Morphing

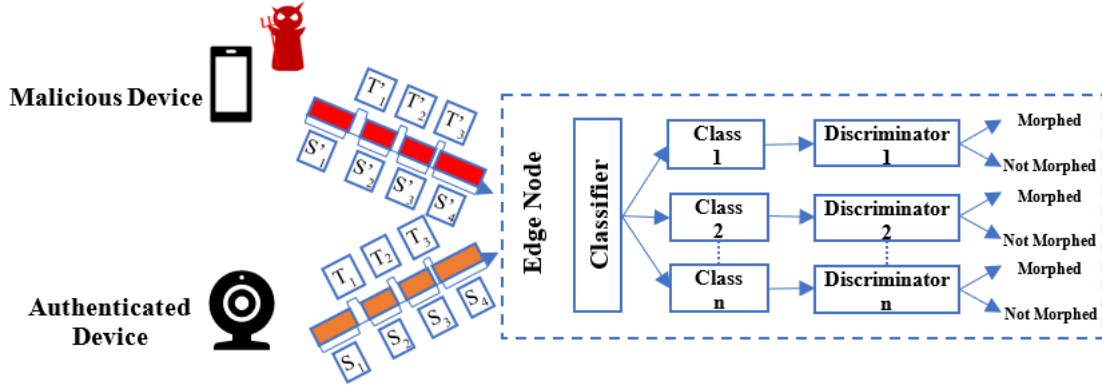
While mutation tries to manipulate the traffic characteristics to confuse the classifier, morphing aims at making one traffic to look like another one. This technique can be used by the attackers to avoid detection. Moreover, in case of ML based device identification, this method can be used for spoofing. In other cases, morphing can be used to guarantee better QoS, by copying time-critical applications traffic. Deep Learning (DL)-based methods have been recently considered for solving complex security tasks. More specifically, the detection of fake media files (images, videos, news, etc.,) have attracted the attention of both academia and industry [367]. In this context, generative adversarial DL architectures have been developed to serve this aim. Inspired by these applications, we consider in this chapter an image-based representation of the network traffic to detect morphed traffic. The proposed solution consists of a generative adversarial DL architecture, including two main parts: a *generator* and a *discriminator*. The *generator* is a Variational Autoencoder (VAE), that has the role of generating morphed network traffic. Contrarily, the *discriminator* aims at detecting morphed traffic by differentiating between the real traffic and the one generated by the VAE. The experiments we conducted considered traffic generated by a set of five different IoT devices. The results show the effectiveness of the proposed solution in generating as well as in detecting morphed traffic.

5.1 Proposed Morphing Detection Solution

In this section, we present our attacker-defender model, the generative DL-based model used for morphing and morphed traffic detection, and our data representation method. For consistency, we list the used notations along with their definitions in Table 5.1.

Table 5.1: Summary of notations

Notation	Definition
Dev_i	i^{th} IoT device
N	Number of IoT devices
$n \times n$	Number of packets considered per flow
F_i	Flow characteristics of device Dev_i
F'_i	Morphed flow characteristics of device Dev_i
FS_i	Flow packet sizes vector
FT_i	Flow packet inter-arrival times vector
FD_i	Flow packet directions vector
Att_i	Attacker imitating device Dev_i
Def	Defender
Enc_i	Encoder training to morph Dev_i traffic
Gen_i	Generator (decoder) training to morph Dev_i traffic
Dis_i	Discriminator used to detect Dev_i morphed traffic



S: Packet Size
T: Interarrival Time

Figure 5.1: Attacker-Defender Model

5.1.1 Attacker-Defender Model

We consider the case where the identification of IoT devices is performed by inspecting the generated traffic. Our network consists of N devices, where each device Dev_i has its identity defined by its flow of packets F_i . At the network edge, statistical features for each $n \times n$ packets of a flow F_i generated by Dev_i are extracted. The features vector F_i consists of three sub-vectors FT_i , FS_i and FD_i , where FT_i is sub-vector consisting of the flow packet inter-arrival times, FS_i is a

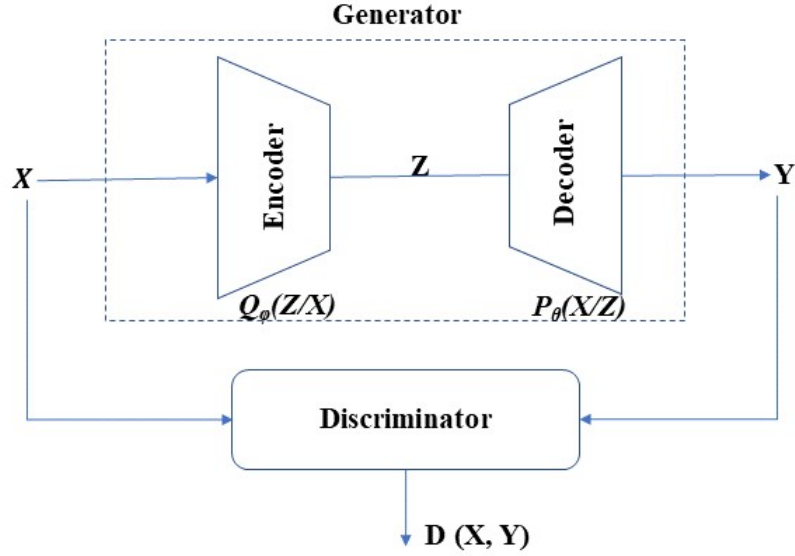


Figure 5.2: Generative Adversarial Deep Learning Model

sub-vector consisting of the flow packet sizes, and FD_i is a sub-vector of the flow packet directions. Our attacker-defender model consists of two players, as shown in Figure 5.1: an adversary Att_i who tries to impersonate a legitimate IoT device by imitating its traffic and generating a similar one F'_i ; and a defender Def at the network edge that tries to detect F'_i as fake (morphed) traffic. Given that the identification of the devices is performed using their generated traffic characteristics, the impersonation attacks try to mimic the real traffic characteristics by means of the following techniques:

1. ***Morphing of the packet sizes***: this technique consists of studying the distribution of the flow packet sizes. The adversary can use this distribution to mimic a legitimate device traffic.
2. ***Morphing of the packets inter-arrival times***: this technique is more sophisticated than the packet size one, given that the rate of sending packets can depend on the device network interface characteristics. This technique consists of studying the distribution of the inter-arrival times and trying to mimic it.
3. ***Morphing of the packet directions***: this technique is the most complicated one, given that the attacker has to control the sender and the receiver. In this case, the attacker tries to inject dummy packets to imitate the direction patterns of the target traffic.
4. ***Morphing of the packet sizes, inter-arrival times, and directions***: in this case, the morphing considers all the traffic characteristics to

mimic the target traffic.

In our case, each attacker Att_i wants to mimic Dev_i traffic F_i . We assume that Att_i is able to sniff this traffic. Then, it trains the VAE to generate similar traffic F'_i . The defender having traffic from the different devices aims at training discriminators to detect morphed traffic for each type. In fact, morphing cannot be easily performed by copying one of the device flow patterns, because network traffic exhibits variability. Alternatively, one should imitate the legitimate device behavior by extracting a general representative flow pattern.

5.1.2 Generative Deep Learning Model

DL generative models can be used for inferring the distribution of the device traffic, permitting attackers to mimic legitimate devices. On the other hand, the discrimination power of the Generative Adversarial Network (GAN) can be used to detect fake traffic. This scenario can be formulated as an adversarial learning problem, where the attacker aims at mimicking victim devices traffic, and the defender aims at detecting any deviations from the norm. In this context, the employed DL generative model consists of three parts as shown in Figure 5.2: the encoder that aims at inferring the distribution $P(F_i)$ for each Dev_i by means of the latent variable z . Applying the VAE concept, the encoder learns the probability distribution $q_\phi(z/x)$, x being the input features vector F_i . The decoder tries to generate a traffic feature vector similar to F_i . Finally, the discriminator aims at differentiating between F_i and F'_i . In the following, we present the loss functions that we define to train our model, Where D_{loss} is the discriminator loss, G_{loss} is the generator loss and AE_{loss} is the autoencoder loss.

The discriminator loss, represented in the following equation, is the cross entropy of the discriminator output probability, consisting of two parts: the first part D_{real} presents the real data detected as real and the second part D_{fake} presents the fake (generated) data detected as fake.

$$\begin{aligned} D_{real} &= (1 - D(x))\log(1 - D(x)); \\ D_{fake} &= D(G(x))\log(D(G(x))); \\ D_{loss} &= D_{fake} + D_{real} \end{aligned}$$

The generator loss, presented in the below equation, is the log loss of the probability of detecting fake (generated) data as real.

$$G_{loss} = (1 - D(G(x)))\log(1 - D(G(x)))$$

The autoencoder loss AE_{loss} , presented below, is expressed in terms of the KL loss and the reconstruction error expressed in terms of the Mean Squared Error (MSE) between original and generated traffic.

$$KL_{loss} = KL(\mathcal{N}(\mu, \sigma^2) || \mathcal{N}(0, 1)) = -\frac{1}{2}(1 + \log(\sigma^2) - \mu^2 - \sigma^2)$$

$$AE_{loss} = ||x - G(x)||^2 + KL_{loss}$$

Training Procedure

For training our model, we consider the traffic of five devices. For each Dev_i , the attacker trains Gen_i to morph the real traffic flows F_i into F'_i . In this case, the discriminator Dis_i aims at enhancing the fake traffic generation. However, at the defender side, the training aims at enabling the discriminator to differentiate between fake (generated) traffic and the real one. As shown in Algorithm 6, training data is passed to the encoder, which extracts the data representation by means of the latent variable z . The decoder generates a flow imitating the input one. The input of the encoder and the output of the decoder are passed to the discriminator as real and fake, respectively. At each epoch, the weights of the model are updated. At the end, one model is saved for the attacker and another one is saved for the defender. The attacker will use the generator part to generate morphed traffic while the defender uses the discriminator part to detect morphed traffic.

Algorithm 6 Training

```

1: procedure TRAINING(train_data)
2:   for  $i \in nb\_of\_devices$  do
3:     for  $j \in nb\_batches$  do
4:       pass  $Batch(train\_data)_j$  to  $enc_i$ 
5:       pass  $Batch(z)_j$  to  $gen_i$ 
6:       pass  $Batch(train\_data)_j$  as real to  $Dis_i$ 
7:       pass  $gen(Batch(z)_j)$  as fake to  $Dis_i$ 
8:       minimize  $loss\_Enc_i$ 
9:       minimize  $loss\_Gen_i$ 
10:      minimize  $loss\_Dis_i$ 
11:      update  $W_i$ 
12:     end for
13:     save  $Enc_i, Gen_i, Dis_i$ 
14:   end for
15: end procedure

```

Testing Procedure

During the testing phase, we run tests between the attacker and defender trained models, as shown in Algorithm 7. The generated data by the attacker, for mimicking Dev_i traffic, is tested against the defender discriminator. This is performed

for each type of morphing including: packet sizes, packet inter-arrival times, direction, and all the above combined. In each case, the trained generator of the attacker is tested against the defender’s discriminator. Although, to test the efficacy of the morphing process, we add a classification phase where we assess the similarity between morphed and original traffic. In each case, a classifier is trained on original data, and then tested on the attacker generated traffic. For example, for the morphing technique where all packet characteristics are morphed, a classifier is trained to classify the morphed traffic flows F'_i based on the device type (five classes). The generated traffic for each device is passed to the classifier to test if it will confuse the classifier. For the other cases, the morphed features of one device Dev_i (sizes FS'_i , inter-arrival times FT'_i , and directions FD'_i) are merged with original flows of other devices and then F'_i is passed to the classifier.

Algorithm 7 Testing

```

1: procedure TESTING(test_data)
2:   for  $i \in nb\_of\_devices$  do
3:     load attacker_trained_modeli
4:     load defender_trained_modeli
5:     pass  $Gen(test\_data)_{att_i}$  to  $Dis_{def_i}$ 
6:     record loss_Enci
7:     record loss_Geni
8:     record Disi_out
9:   end for
10: end procedure

```

5.1.3 Data Representation

Using our proposed data representation described in chapter 2, we extract features from the first $n \times n$ packets, omitting the transport protocol feature. In addition, in this chapter, we consider a different scaling given that we are dealing with IoT traffic having small-sized packets. Consequently, we choose to scale packets sizes between 0 and 255, and inter-arrival times are scaled between 0 and 10 ms. For each packet of the $n \times n$ packets per flow, if its size is greater than 255, the extracted size is set to 255. Similarly, for inter-arrival time, if its is greater than 10 ms, it is set to 255, if not, it is divided by 10 and multiplied by 255. The direction is set to 0, if the packet is outgoing, and to 255, if it is an incoming packet. In the visualized images, R is given to the size, G is given to the inter-arrival time, and B is given to the direction. Note that before passing the features vectors to the DL network, they are all normalized between 0 and 1 (i.e. divided by 255).

5.2 Experimental Analysis

In this section, we present the implementation of the proposed model along with the testing results.

5.2.1 Data Collection

To collect traffic, we consider the set of IoT devices: D-Link HD 180-Degree Wi-Fi Camera DCS-8200LH, D-Link Wi-Fi Smart Plug DSP-W215, D-Link Wi-Fi Siren DCH-S220, D-Link Wi-Fi Water Sensor DCH-S160, and Samsung SmartThings Home Monitoring Kit. Using the same setup described in chapter 3, one day of traffic was collected for the attacker model training and one day was collected for the defender model training. These devices were deployed in a home environment and left to function normally.

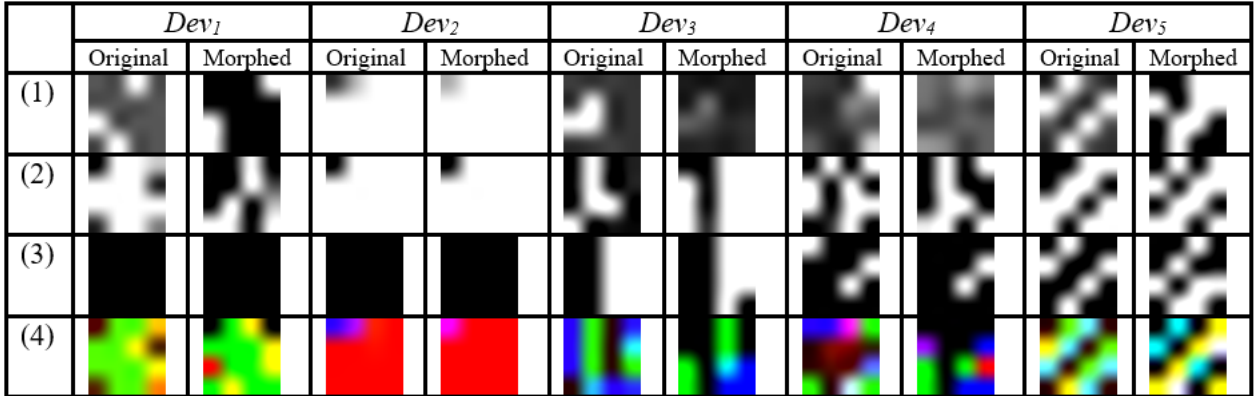


Figure 5.3: 4x4 obtained images

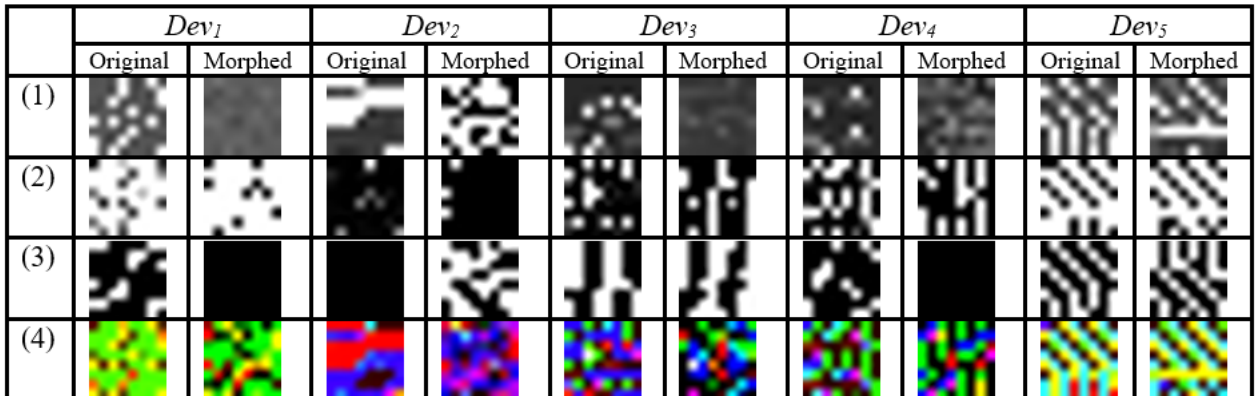


Figure 5.4: 8x8 obtained images

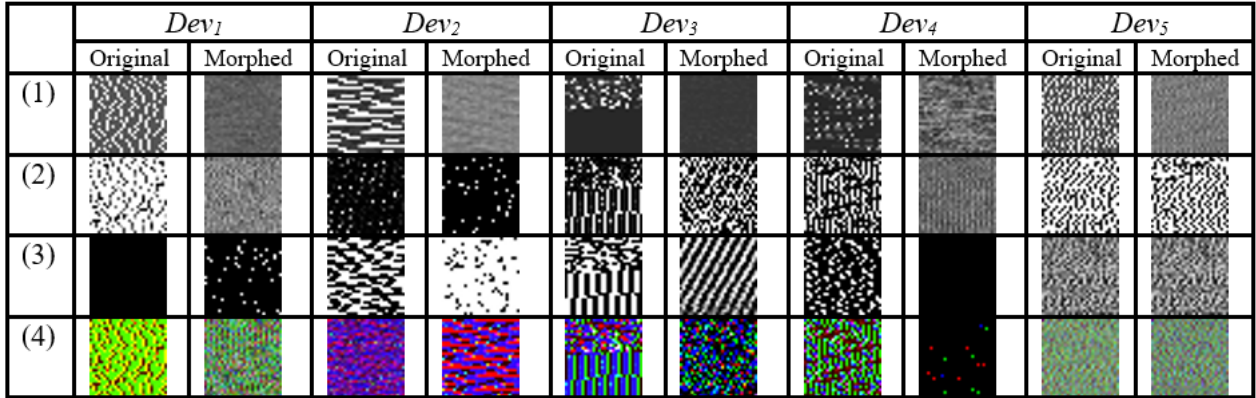


Figure 5.5: 28x28 obtained images

Morphing Technique	Accuracy					
	4x4		8x8		28x28	
	CNN	RF	CNN	RF	CNN	RF
(1)	68.04%	48.25%	46.67%	43.91%	31.56%	22.59%
(2)	52.42%	48.01%	73.27%	48.99%	38.92%	23.17%
(3)	37.63%	33.72%	36.34%	26.14%	25.04%	21.56%
(4)	87.68%	73.43%	89.97%	80.28%	51.56%	52.77%

Table 5.2: Classification Results

Device	4x4				8x8				28x28			
	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
<i>Dev₁</i>	0.30	0.38	0.21	0.10	0.36	8.53E-14	1.22E-11	0.22	0.09	0.21	0.04	0.15
<i>Dev₂</i>	0.35	0.27	0.39	0.33	0.37	0.26	0.49	0.30	0.12	0.26	0.58	0.23
<i>Dev₃</i>	0.03	0.41	0.49	0.32	0.03	0.38	0.46	0.33	0.03	0.40	0.41	0.29
<i>Dev₄</i>	0.10	0.43	0.26	0.33	0.18	0.42	0.19	0.33	0.11	0.23	0.19	0.28
<i>Dev₅</i>	0.33	0.41	0.43	0.39	0.33	0.41	0.46	0.38	0.09	0.22	0.30	0.20
Total	0.22	0.38	0.31	0.32	0.20	0.36	0.32	0.31	0.09	0.26	0.30	0.23

Table 5.3: Morphing Loss

Device	4x4				8x8				28x28			
	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)	(1)	(2)	(3)	(4)
<i>Dev₁</i>	100%	100%	50%	65.97%	50%	60.82%	50%	50%	54.12%	50%	100%	50%
<i>Dev₂</i>	100%	88.65%	89.17%	96.90%	100%	97.93%	93.81%	69.07%	51.54%	100%	100%	80.92%
<i>Dev₃</i>	97.42%	97.93%	81.44%	100%	50%	94.84%	86.08%	90.20%	48.45%	90.72%	60.30%	83.50%
<i>Dev₄</i>	100%	68.55%	94.84%	97.42%	50.51%	94.84%	91.75%	75.25%	46.90%	61.85%	93.29%	80.92%
<i>Dev₅</i>	93.81%	98.45%	88.14%	94.32%	100%	90.2%	92.78%	66.49%	50.51%	51.03%	51.03%	65.56%
Total	95.97%	92.98%	80.72%	90.92%	70.10%	87.73%	82.88%	70.2%	50.30%	70.72%	80.92%	66.18%

Table 5.4: Detection Rate

5.2.2 Model Implementation

We used the Tensorflow library to implement the proposed model in Python [368]. Each component of the model consists of a 2-layer fully connected neural network with 1,000 neurons each. The encoder outputs two values z_mean and z_sigma . The decoder output is a Sigmoid layer with same size as the input, the discriminator output is of one dimension. Each model was trained with 10 epochs and a batch size of 1, the learning rate is 10^{-3} , and the momentum decay is 0.9 (beta1). We used the Rectified Linear Unit (ReLU) as the activation function for all hidden layers, and the weights were optimized using the Adam optimizer. To test the morphing correctness, we refer to classification to see if the morphed traffic is classified correctly. To this aim, we use two classifiers, a CNN classifier applied to our data representation and a Random Forest classifier applied to statistical features extracted per flow (see Table 3.2). The CNN classifier consists of three convolutional layers, two max pooling layers, and two fully connected layers with one dropout layer with 50% dropout probability. ReLU is used as an activation function in the hidden layers, and the Adam algorithm is used for optimization with a learning rate of 10^{-3} . The RF classifier is implemented using scikit-learn [358], search grid is used for parameters tuning with range of 1,000 trees. Cross validation with four folds is applied for training the classifiers. To evaluate the considered classifiers, we refer to the accuracy metric, which consists of the ratio of the correctly classified samples to the total number of samples.

5.2.3 Testing Results

In the following, we detailed the testing results of the morphing process performed by the attacker, and the detection done at the defender side.

Morphing Results

The results are reported for each morphing technique and each device. In addition, different values of n are considered (4, 8, and 28). To visually assess the morphing process, we visualize the original and morphed traffic, as shown in Figure 5.3, Figure 5.4, and Figure 5.5. These images show that the morphing process succeeded in generating similar traffic characteristics considering different types of morphing. Also we included the reconstruction loss, consisting of the MSE between original and generated traffic in Table 5.3. The classification results of both classifiers, CNN and RF, are included in Table 5.2. The results show the effectiveness of morphing the traffic, especially when considering all features (packet sizes, inter-arrival times, and directions). It is clear that the classification results depend on the number of packets considered per flow n (see Table 5.2). The best morphed traffic classification result is obtained with 8x8 packets, with the morphing technique (4) and the CNN classifier, which is 87.68%. Considering the

morphing of a single feature (packet sizes, inter-arrival times, or directions), it can be noticed also that morphing one feature is not sufficient to confuse noticeably the classifier, and that morphing the packet sizes with 4x4 packets gives better results than the other features (68.04%). For 8x8 and 28x28 packets, morphing the inter-arrival times gives better results (73.27%, and 38.92% respectively).

Detection Results

Concerning the detection rate, the results shown in Table 5.4 reveal the defender ability in detecting any kind of morphing with a detection rate up to 95%. It can be noticed that overall single features morphing tests ((1), (2), and (3)) present better detection rates than the all features morphing (4). Looking at the influence of n on the detection rate of the single features morphing, we can remark that: for $n = 4$, using packet sizes information, morphing can be detected with high accuracy; for $n = 8$, inter-arrival times information give better insight to detect morphing; finally, for $n = 28$, packet directions can be better used for morphing detection.

Concerning the relation between morphing loss and morphed traffic detection rate, they decrease when n increases (see Table 5.3 and Table 5.4). At small scale (4x4), the variety of the patterns lead to high morphing loss (0.32 with morphing (4)) and thus, a high discrimination rate (90.92% with morphing (4)). However, with high scale (28x28), the morphing decreases (to 0.3 with morphing (4)), and the discrimination rate decreases also (to 66.18% with morphing (4)).

5.3 Discussions

The results show that detection of morphed traffic is possible using adversarial learning. In fact, the network traffic presents diversity even for the same device. Morphing techniques try to extract the common patterns between the different flows pertaining to a specific traffic type (per app or per device). In this work, we show that the differentiation between fake and original traffic is possible by means of a quasi-raw data representation, showing the packet characteristics. In contrast, the detection was not possible when using pure statistical features.

The visualization of the original and morphed traffic, in Figure 5.3, Figure 5.4, and Figure 5.5, show that the generative DL model is able to extract the original traffic characteristics. This was also reported in the reconstruction loss between the original traffic and the generated one. On the other hand, the adversarial learning succeeded in detecting morphed traffic as shown in the recorded discrimination rates (see Table 5.4).

This work paves the way towards a new direction in the intrusion detection domain, where new techniques can be applied for detecting abnormal traffic. In fact, morphing can be used by attackers to hide attacks other than the ones con-

sidered in this work. Attackers can rely on real apps traffic (Facebook, Skype, etc.) to shape their attacks flows and avoid detection by IDS. Applying the new data representation method with adversarial learning are key for obfuscated traffic detection. Actually, morphing is one of many other obfuscation techniques that could be applied to thwart classification and intrusion detection. The presented solution could be applied for any type of obfuscation, including mutation. In this chapter, different values of n were considered, and the results show that for large n the morphing fails and that the characteristics of the traffic of the different devices become similar, as shown in the classification results of the morphed traffic ($\sim 50\%$). The question then becomes how to optimize the choice of n , which will be considered in the next chapter.

5.4 Conclusion

ML-based techniques have emerged in the network domain for traffic classification, intrusion detection, and more recently for IoT device identification. However, these methods are prone to failure when traffic obfuscation techniques are used. More specifically, morphing, which consists of imitating other traffic characteristics, is a critical challenge for statistical based IDSes. In this chapter, we present a generative adversarial DL-based method to detect morphed traffic, where different morphing techniques were applied to IoT traffic. Even though this was presented in the context of device identification/authentication, it can be applied for other types of traffic and for other classification goals (e.g. QoS management). The results show the effectiveness of the adversarial learning in accounting for adversarial attacks and detecting intrusion.

Chapter 6

Real-time Traffic Classifier

In this chapter, we aim at finding the sufficient number of packets that guarantees good classification accuracy while optimizing the response time. Before optimizing n , we first visualize data for different values of n . To do so, we consider features reduction techniques. The traditional techniques like principal component analysis (PCA), locality preserving projections (LPP), discriminative locality alignment (DLA), linear discriminative analysis (LDA) present limitations, when applied in the image domain. In addition, these techniques are limited to linear models and do not preserve non-linearity. Thus, t-SNE (Stochastic Neighbor Embedding) is proposed to reduce high-dimensional data (like images) while reserving the data non-linearity. While traditional features selection and reduction techniques rely on the correlation or mutual information between the data features and the label vector to select the best features, the problem to define the sufficient number of instances for time series data calls for different approaches. In this context, our aim is to have a measure that reflects the model performance as a function of the number of considered packets. To do so, we propose a confidence measure based on the variations in the model training accuracy and the average mutual information between the packets features and the label vector. This measure can help in finding a value of n that reduces the time overhead while guaranteeing good classification accuracy. In addition, we propose a new classifier model to enhance the classification accuracy by training successive models on the streaming traffic data. The proposed ensemble method is based on the average of the individual classifiers predictions.

6.1 Data Visualization

To visualize high dimensional data, a method, called t-SNE, was proposed by Laurens van der Maaten in [369]. t-SNE is a modified version of the Stochastic Neighbor Embedding (SNE) algorithm. It consists of transforming high dimensional data into low dimensional components by minimizing the Kullback-

Leibler divergence between the probability distributions in the initial space of picking neighbor points and the probability distribution in the transformed space to pick these points as neighbors. The first step is to convert the euclidean distances between data points in the high dimensional space into conditional probabilities that reflect the similarity between these points. The conditional probability $p_{j|i}$ expresses the probability of picking x_j from the neighbors of x_i in proportion to the probability density of x_i neighbors as a Gaussian centered distribution at x_i . For nearby points, this probability is high; however, for faraway points it is small. In this case, $p_{i|i}$ and $p_{j|j}$ are set to zero. $p_{j|i}$ is expressed as following: $p_{j|i} = \frac{\exp(-\|x_i-x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i-x_k\|^2/2\sigma_i^2)}$; where σ_i is the variance of the Gaussian distribution centered at x_i . Furthermore, $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$ The same is represented in the low dimensional space with $q_{j|i} = \frac{\exp(-\|y_i-y_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|y_i-y_k\|^2/2\sigma_i^2)}$ and $q_{ij} = \frac{(1+\|y_i-y_j\|^2)^{-1}}{\sum_{k \neq i} (1+\|y_i-y_k\|^2)^{-1}}$. Thus, the method relies on minimizing the KL between $p_{j|i}$ and $q_{j|i}$, $KL(P||Q) = \sum_{k \neq i} p_{ij} \log(\frac{p_{ij}}{q_{ij}})$. This minimization is solved by using gradient descent. A very important attribute of t-SNE is the perplexity $perp(P) = 2^H(P_i)$, where $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$ this value reflects the number of neighbors as density based clustering.

In our work, we have images of dimensions $n \times n$, to visualize the effect of n on the data distribution, we applied t-SNE to visualize the data for different values of n . In Figure 6.1, we include some of the obtained figures.

It is clear from the images, included in Figure 6.1, that the number of packets affects the distributions and the separation between the different classes. This indicates that the choice of n affects the classification accuracy and the differentiation between the instances of the different classes.

6.2 Confidence Measure

Based on the results in previous chapters, the choice of n affects the classification accuracy. Increasing n does not always guarantee better accuracy while increasing the training overhead. Moreover, waiting for large number of packets to perform the testing classification is not acceptable in real time applications like intrusion detection. In this case, finding the optimal number of packets or the optimal value of n that guarantees good testing accuracy while minimizing the required number of packets is key. This applies to other cases of time series data where for example being able to recognize the voice from the first samples is required for real-time applications. While the problem can be approached as a traditional problem of features selection or reduction techniques, these techniques are not effective in the case of streaming data being unable to have the full vector of features ahead of time. Thus, the features importance should be analysed as the data comes in to reach a level where the considered data features are enough to

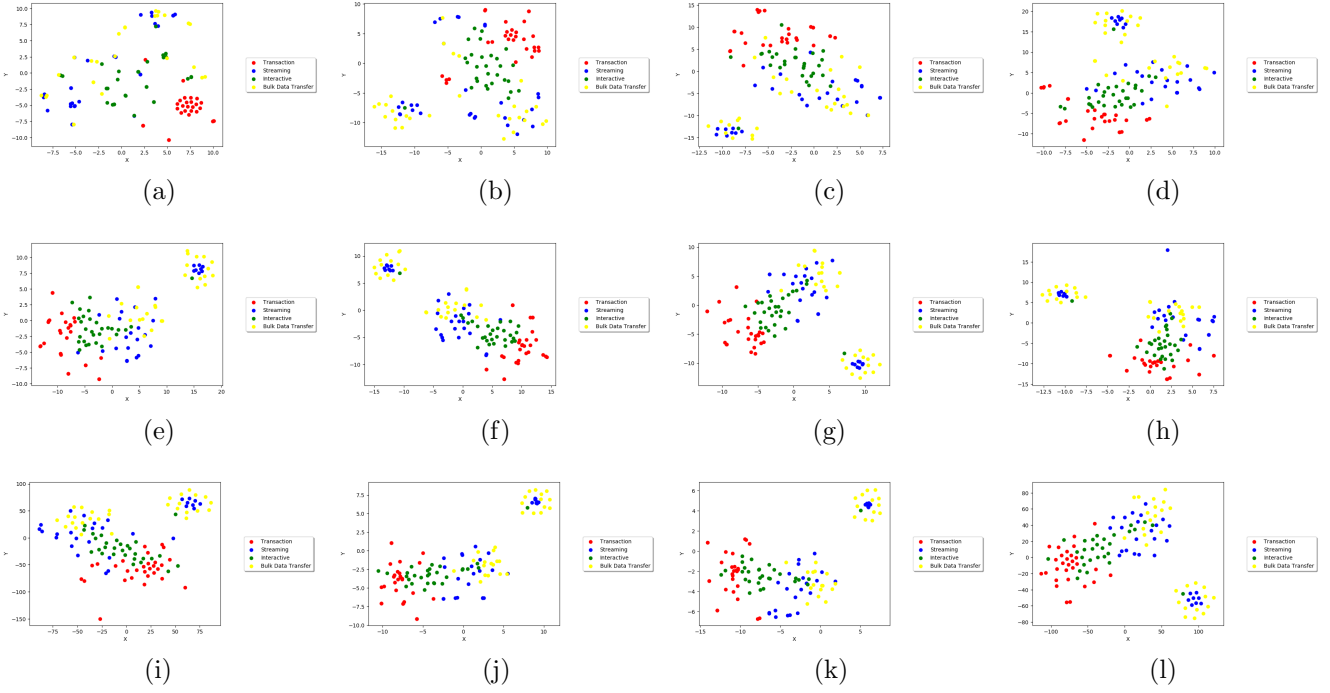


Figure 6.1: t-SNE visualization of Level 1 data for different values of n : (a) $n = 2$, (b) $n = 4$, (c) $n = 6$, (d) $n = 8$, (e) $n = 10$, (f) $n = 12$, (g) $n = 14$, (h) $n = 16$, (i) $n = 18$, (j) $n = 20$, (k) $n = 22$, (l) $n = 24$.

guarantee good classification accuracy. In this section, we present our proposed confidence measure based on the mutual information between the features and the class labels vector and the training accuracy. Then, this measure is used to choose an optimal n , meeting the compromise between the response time and the test accuracy.

In chapter 2, we presented our data representation method that consists of extracting basic packet characteristics: size, inter-arrival time, and direction from the first $n \times n$ packets and transforming them into RGB images. The flows data can be represented in three channels: R, G, B representing the size, timestamp, and direction. As shown in 6.2, a sample flow can be represented by three matrices X_R , X_G , and X_B , where each matrix component, r_i , g_i , and b_i represents a pixel of the RGB image. Thus each component of these matrices is represented by a random variable R_i , G_i , and B_i for R, G, and B channels.

To optimize the choice of n , we rely on computing the mutual information between the image and the label vector, $I(X, Y)$. The choice of the optimal value of n is obtained when the mutual information between X and Y is maximized, where $X = (X_R, X_G, X_B)$. The mutual information is the average sum of the individual mutual information between each color components and the label vector, as shown in the following:

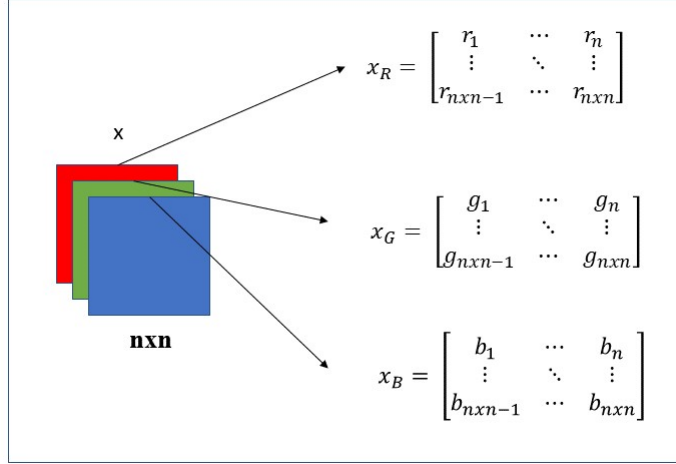


Figure 6.2: Image Representation

$$\begin{aligned}
 I_R &= \frac{1}{n \times n} \times \sum_{i=0}^{i=n \times n} I(R_i, Y); \\
 I_G &= \frac{1}{n \times n} \times \sum_{i=0}^{i=n \times n} I(G_i, Y); \\
 I_B &= \frac{1}{n \times n} \times \sum_{i=0}^{i=n \times n} I(B_i, Y); \\
 I &= \frac{I_R + I_G + I_B}{3}
 \end{aligned}$$

In Algorithm 8, we present our algorithm that aims at finding the best value of n to meet the compromise between the accuracy and the classification performance. To do so, we define a new confidence measure that is based on the training accuracy and the mutual information between the data and the label vector. The rate of change of this confidence measure is expressed in terms of the rate of change in the accuracy and the mutual information, as follows:

$$\begin{aligned}
 \delta_{Acc}(n) &= \frac{Acc(n) - Acc(n-1)}{Acc(n-1)}; \\
 \delta_{MI}(n) &= \frac{MI(n) - MI(n-1)}{MI(n-1)}; \\
 \delta_C(n) &= \frac{\delta_{Acc}(n) + \delta_{MI}(n)}{2};
 \end{aligned} \tag{6.1}$$

where $Acc(n)$ is the training accuracy when $n \times n$ packets are considered and $MI(n)$ is the average mutual information between the features vector of

the $n \times n$ packets features and the class labels vector, $MI(n) = \frac{\sum_0^{n \times n} MI(x_i, Y)}{n \times n}$. Consequently, the confidence $C(n)$ can be expressed in terms of $\delta_C(n)$ as follows:

$$C(n) = C(n-1) \times (1 + \delta_C(n)) \quad (6.2)$$

It should be noticed here that a necessary condition should be met to have $0 < C(n) < 1$, is that $\delta_C(n) < \frac{1-C(n-1)}{C(n-1)}$. In our case, we check if this condition is correct, if not, $\delta_C(n)$ is set to $\frac{1-C(n-1)}{C(n-1)}$. In fact, this bound is found based on the fact that $0 < C(n) < 1$, consequently, $-C(n-1) < C(n) - C(n-1) < 1 - C(n-1)$. Thus, having $0 < C(n-1) < 1$, we can divide all the inequality's sides by it. As a result, we get $-1 < \delta_C(n) < \frac{1-C(n-1)}{C(n-1)}$.

Algorithm 8 Confidence Measure Based Choice of n

```

1: procedure OPTIMIZE  $n(\text{Acc}, \text{MI})$ 
2:    $n = N_{min}$ 
3:    $C(0) \leftarrow (\text{Acc}(0) + \text{MI}(0))/2$ 
4:   while  $n < N_{max}$  do
5:      $\delta_{Acc}(n) \leftarrow \frac{\text{Acc}(n) - \text{Acc}(n-1)}{\text{Acc}(n) + \text{Acc}(n-1)}$ 
6:      $\delta_{MI}(n) \leftarrow \frac{\text{MI}(n) - \text{MI}(n-1)}{\text{MI}(n) + \text{MI}(n-1)}$ 
7:      $C(n) \leftarrow C(n) = C(n-1) \times (1 + \frac{\delta_{Acc}(n) + \delta_{MI}(n)}{2})$ 
8:   end while
9:    $max_C = C(0)$ 
10:   $n_{optimal} = 0$ 
11:  for  $n \in \text{range}(N_{min}, N_{max})$  do
12:    if  $C(n) > max_C$  then
13:       $n_{optimal} = n$ 
14:       $max_C = C(n)$ 
15:    return  $n_{optimal}$ 
16:  end for
17: end procedure

```

In Algorithm 8, we include the algorithm of optimizing the choice of n . At the training phase, first, the features for the first $N_{max} \times N_{max}$ packets are extracted. N_{max} is chosen based on the application and the training data size. In our case, we choose $N_{max} = 28$. For getting the training accuracy, we train $N_{max} - N_{min}$ models by successively increasing n by 1. The training accuracy is computed based on the validation data being 20% of the training one. Then, after computing the training accuracy, the mutual information at each value of n is computed between each packet feature and the label vector. The average sum is computed and the confidence measure is calculated as stated above. Finally, the optimal n is the one giving the maximal confidence measure.

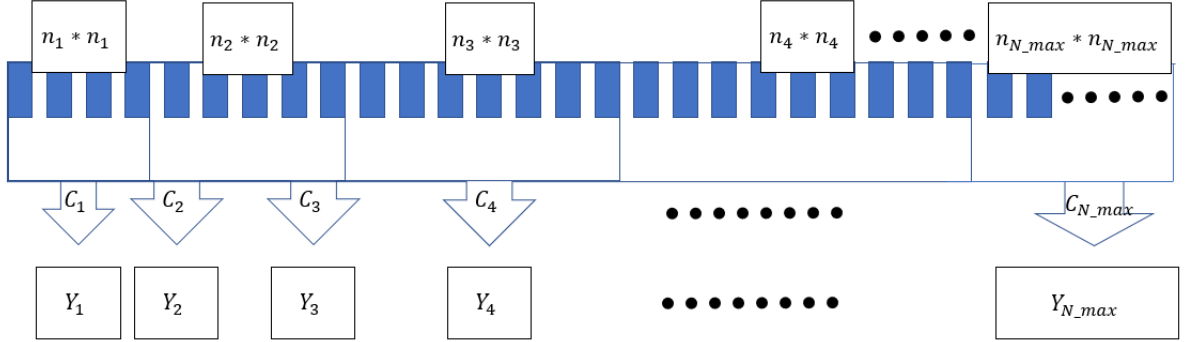


Figure 6.3: Classifier Model

6.3 Ensemble Classifier Model

In this section, we present our classifier model. This model aims at enhancing the classification accuracy at certain value of n by considering the preceding classifiers results. This model considers a streaming data type, in our case the successive packets of a flow. At each value of n , a model is trained with input of dimensions $n \times n$. The classification results of the preceding classifiers are aggregated and the weighted average is used to get the ensemble decision.

As shown in Figure 6.3, our model consists of successive classifiers of different dimensions. For each value of n , a classifier C_i , where $i = 2, \dots, N_{max}$, is trained on data with dimension $n \times n$. At each n , the classifications at previous values of n are considered. The successive classifiers form an ensemble model, where the classifiers predictions are averaged to get the final decision. In the algorithm below, we present how these classifiers predictions are aggregated and how the accuracy of the ensemble classifier is computed. It should be noted, as shown in Algorithm 9, that the probabilistic output of the classifiers is considered and not only the class label. Consequently, the aggregated results include also the classification confidence.

Algorithm 9 Ensemble Classification

```

1: procedure OPTIMIZE ACCURACY( $n$ )
2:   for  $item \in X_{test}$  do
3:     while  $i < n$  do
4:        $loadmodel_i$ 
5:        $P_{item}(i) = model_i.predict(item)$ 
6:     end while
7:      $P_{item} \leftarrow \frac{\sum 0nP_{item}(i)}{n}$ 
8:   end for
9: end procedure

```

6.4 Experimentation Results

In this section, we include the results considering our dataset at different levels in addition to an online voice dataset consisting of spoken digits [370].

6.4.1 Confidence Measure Results

Examples of the results, including the training accuracy, the mutual information, and the computed confidence measure, are shown in Figure 6.4: a) for level 1, b) to e) for level 2, f) to k) for level 3, and l) to z) for level 4. These figures clearly indicate that one can choose the value of n corresponding to the maximum of the confidence value in the aim to achieve good classification performance at an early stage.

6.4.2 Ensemble Classifier Results

For assessing the proposed ensemble classifier, we compare different ensemble techniques to compute the final decision, including: average sum, weighted sum, and majority voting. In addition, we compare the performance of the ensemble method compared to the individual classifiers performance. For the average method, we computed the final decision based on Algorithm 9. For the weighted sum, the probabilistic output of each classifier is multiplied by its confidence measure before computing the average over the sum of weights (confidence measures of the successive classifiers). However, for the majority voting, the final decision is computed based on the maximum number of votes for the output class. In table 6.6, we include the average results for the different levels and the voice data. It is clear from the results, that the average voting gives the best results. Moreover, it can be noticed that the ensemble based accuracy is greater than the accuracy obtained by the individual classifiers.

6.4.3 Optimal n Results

In this section, we present the results obtained by applying the proposed algorithm for the optimal choice of n by considering our data collected with hierarchical classification. Furthermore, to verify the generalization of the proposed model to time series data, the proposed confidence measure and the classification algorithm was applied to a voice dataset of spoken digits [370]. In this case, n represents the number of samples in the time domain. However, the mutual information is computed after transforming n samples from the domain into Mel-Frequency Cepstral Coefficients (MFCCs) features. Similar to the case of network flows, the voice time series are transformed into gray images of their MFCCs and CNN is applied. The results show the efficiency of the proposed confidence measure along with the ensemble classifier in the case of time series data.

Data	Average	Weighted Average	Majority Voting	Individual
Level 1				
All	89.77%	86.81%	89.76%	87.34%
Level 2				
Bulk Data Transfer	94.74%	92.98%	94.52%	92.41%
Interactive	81.08%	77.74%	79.92%	78.52%
Streaming	94.03%	92.32%	94.03%	93.04%
Transaction	66.67%	66.67%	67.32%	63.16%
Average	84.13%	82.43%	83.95%	81.78%
Level 3				
Audio Interactive	74.26%	70.80%	72.87%	65.79%
Video Interactive	37.94%	37.89%	38.01%	42.68%
Texting Interactive	80.63%	80.41%	79.33%	75.78%
File Upload	75.56%	74.58%	73.95%	69.74%
File Download	89.60%	86.94%	89.05%	88.55%
Average	71.60%	70.12%	70.64%	68.51%
Level 4				
Actuation	77.51%	76.06%	72.90%	73.71%
Download App	99.02%	98.90%	98.73%	96.87%
Dropbox Download	87.17%	85.95%	86.07%	85.88%
Dropbox Upload	97.77%	95.31%	97.62%	96.70%
Gaming	87.88%	86.87%	85.52%	79.12%
Hangout Audio	56.81%	58.14%	58.28%	69.65%
Hangout Texting	87.23%	86.68%	87.31%	84.24%
Hangout Video	91.03%	90.79%	91.51%	88.55%
Messenger Audio	97.57%	97.71%	97.64%	97.31%
Messenger Texting	91.67%	91.67%	91.67%	91.36%
Messenger Video	52.23%	53.69%	51.49%	49.24%
MyDrive Download	96.74%	93.41%	96.32%	95.12%
MyDrive Upload	96.69%	96.69%	96.78%	95.51%
OneDrive Download	99.06%	99.01%	99.03%	98.81%
OneDrive Upload	90.11%	89.69%	90.55%	90.54%
Sensing	74.29%	74.51%	77.12%	78.21%
Skype Audio	47.43%	48.52%	47.29%	39.00%
Skype Texting	90.42%	90.55%	89.78%	91.32%
Skype Video	94.68%	97.45%	94.91%	90.51%
System Update	95.09%	95.00%	94.54%	89.72%
Audio Streaming	59.84%	60.10%	60.10%	58.57%
Video Streaming	96.88%	95.97%	97.13%	94.06%
Average	84.87%	84.67%	84.65%	83.36%
Voice	75.18%	74.40%	75.04%	67.19%

Table 6.1: Ensemble Accuracy Results

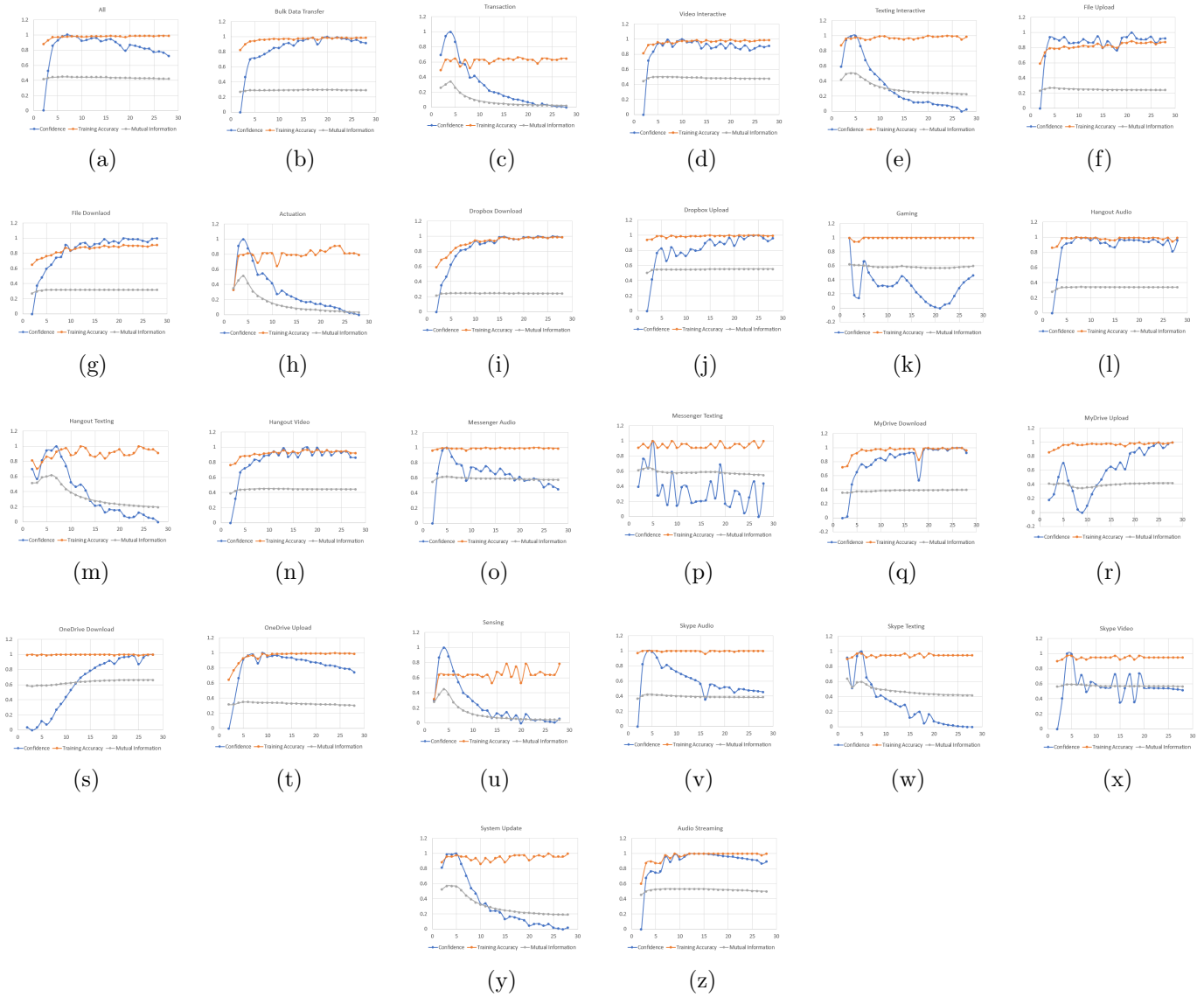


Figure 6.4: Confidence Measure Results

In table 6.4.3, we include the results of n and the corresponding accuracy results when relying on the mutual information, confidence or the training accuracy and the corresponding maximum attained in the testing phase. It is clear from the results that relying on the maximum mutual information to find the optimal n presents low average in number of packets (36 at level 1, 127 at level 2, 35 at level 3, 164 at level 4, and 110 samples for the voice data) and good accuracy (87.94 at level 1, 84.70 at level 2, 66.95 at level 3, 82.34% at level 4, and 50.20% for the voice data). However, relying on the maximum training accuracy to find the optimal n presents high number of packets (729 at level 1, 402 at level 2,

540 at level 3, 229 at level 4, and 2610 samples for the voice data) and better accuracy (91.55% at level 1, 81.59% at level 2, 70.84% at level 3, 84.93% at level 4, and 67.40% for the voice data). Having the confidence as combination of these two measures, relying on the maximum confidence for finding the optimal n the number of packets is higher than of the mutual information (49 at level 1, 149 at level 2, 286 at level 3, 196 at level 4, and 510 samples for the voice data) but it presents higher accuracy (88.11% at level 1, 86.19% at level 2, 71.16% at level 3, 84.43% at level 4, and 70.60% samples for the voice data). It can be noticed also that applying ensemble, the accuracy presented in the column "Ens" presents an enhancement to the accuracy obtained with individual classifiers presented in the column "Ind". It should be noted here that $N = n \times n$ packets for the case of network traffic and the number of samples for the voice data.

To evaluate the enhancement and/or degradation in terms of number of packets and the of accuracy when choosing n relying on the mutual information, training accuracy, or our proposed confidence measure, we computed the difference between the obtained n and accuracy and the ones obtained from the testing phase, as shown in table 6.4.3. It can be noticed that on average, our proposed confidence measure, if used to choose the optimal n , present better enhancement in the number packets than the training accuracy (as shown in table 6.4.3) and lower degradation in the accuracy than the mutual information method (as shown in table 6.4.3). The confidence measure method presents an enhancement of 680 in the number of packets with degradation of 3.44% in the classification accuracy at Level 1 and a degradation of 1.29% in the accuracy with the ensemble method. At Level 2, it presents an enhancement of 202 in the number of packets with degradation of 3.28% in the individual classification accuracy and an enhancement of 0.63% with the ensemble method. At Level 3, it presents an enhancement of 105 in the number of packets with degradation of 8.29% in the classification accuracy which increases to 2.15% with the ensemble method. At Level 4, it presents an enhancement of 54 in the number of packets with accuracy degradation of 6.88%, which decreases to 2.29% with ensemble method. Considering the voice data, the proposed algorithm for finding $n_{optimal}$ based on the confidence measure presents an enhancement of 1300 in the number of samples with degradation of 8.80% in the accuracy and an enhancement of 7.40% in the accuracy with ensemble method (see table 6.7).

To evaluate the enhancement and/or degradation in terms of time when choosing n relying on the mutual information, training accuracy, or our proposed confidence measure, we computed the difference between the flow time for the obtained n and the one obtained for the best n at the testing phase, as shown in table 6.6. It can be noticed that on average, our proposed confidence measure, if used to choose the optimal n , present better enhancement in the required flow time than the training accuracy. The confidence measure method presents an enhancement of 5.94 seconds in the flow time at Level 1, 2.78 seconds at Level 2, 3.09 seconds at Level 3, and 0.96 seconds at Level 4. Considering the voice data, the results

Data	MI			Training			Confidence			Testing		
	N	Ind(%)	Ens(%)	N	Ind(%)	Ens(%)	N	Ind	Ens(%)	N	Ind(%)	Ens(%)
Level 1												
All	36	87.94	90.09	729	91.55	91.55	49	88.11	90.26	729	91.55	91.55
Level 2												
Bulk Data Transfer	324	95.34	96.07	484	95.02	96.48	400	94.01	96.23	324	95.34	96.07
Interactive	121	86.30	86.51	441	92.60	79.27	81	86.19	85.67	441	92.60	79.27
Streaming	49	85.23	89.93	361	96.64	97.32	100	92.62	93.29	625	97.99	97.32
Transaction	16	71.93	66.67	324	42.11	66.67	16	71.93	66.67	16	71.93	66.67
Average	127	84.70	84.80	402	81.59	84.93	149	86.19	85.46	351	89.46	84.83
Level 3												
Audio Interactive	49	63.70	72.88	676	63.27	77.06	81	68.23	69.34	256	77.03	77.00
Video Interactive	36	36.04	35.71	400	41.09	38.82	100	38.78	37.20	289	65.26	39.01
Texting Interactive	16	76.00	80.00	400	79.00	81.00	25	78.00	81.00	441	84.00	83.00
File Upload	25	71.27	72.36	441	80.00	80.91	441	80.00	80.91	441	80.00	80.91
File Download	49	87.75	87.65	784	90.81	92.85	784	90.81	92.85	529	90.99	92.15
Average	35	66.95	69.72	540	70.84	74.13	286	71.16	72.26	391	79.46	74.41
Level 4												
Hangout Texting	36	80.85	72.34	144	100	95.74	49	91.49	74.47	144	100	95.74
OneDrive Download	784	98.92	99.03	9	98.71	98.92	784	98.92	99.03	196	99.25	99.14
Audio Streaming	64	57.56	61.05	81	57.56	61.05	169	57.56	61.05	121	61.05	61.05
Gaming	4	81.82	81.82	4	81.82	81.82	4	81.82	81.82	100	90.91	90.91
MyDrive Download	784	99.44	99.16	529	99.02	98.74	676	99.30	98.88	784	99.44	99.16
Messenger Video	16	53.51	52.63	64	47.37	54.82	25	55.70	54.82	25	55.70	54.82
Video Streaming	49	89.07	89.07	49	89.07	89.07	49	89.07	89.07	64	99.64	100
Download App	16	96.82	98.50	64	95.51	99.25	16	96.82	98.50	361	98.88	99.07
MyDrive Upload	784	96.47	96.47	625	95.29	96.47	784	96.47	96.47	361	100	97.65
Sensing	16	76.47	76.47	289	88.24	76.47	16	76.47	76.47	36	88.24	76.47
Hangout Video	81	89.17	89.17	324	91.25	92.92	324	91.25	92.92	225	94.58	92.92
OneDrive Upload	36	83.81	79.30	576	97.75	98.57	81	85.25	88.52	484	99.39	98.98
Messenger Audio	25	99.12	99.12	169	97.35	97.35	25	99.12	99.12	25	99.12	99.12
Skype Video	16	100	87.50	16	100	87.50	16	100	87.50	16	100	87.50
Skype Texting	4	86.21	86.21	16	89.66	96.55	25	89.66	96.55	625	100	89.66
Dropbox Upload	729	99.32	98.63	225	98.63	99.32	625	99.32	98.63	81	99.32	98.63
Skype Audio	16	48.37	48.37	9	48.37	48.37	16	48.37	48.37	36	48.50	48.37
System Update	9	92.50	97.50	576	87.50	95.00	25	90.00	95.00	121	95.00	95.00
Hangout Audio	64	59.34	52.15	49	64.65	52.27	49	64.65	52.27	625	94.19	64.77
Dropbox Download	49	78.32	84.62	676	95.45	94.06	529	93.71	93.36	676	95.45	94.06
Actuation	16	60.98	85.37	529	53.66	75.61	16	60.98	85.37	361	90.24	75.61
Messenger Texting	16	83.33	91.67	25	91.67	91.67	25	91.67	91.67	64	100	91.67
Average	164	82.34	83.01	229	84.93	85.52	196	84.43	84.54	251	91.31	86.83

Table 6.2: Best Choice of n

Data	MI Enhancement			
	N	Ind(%)	Ens(%)	Ens test(%)
Level 1				
All	-693	-3.61	-1.46	-1.46
Level 2				
Bulk Data Transfer	0	0.00	0.73	0.00
Interactive	-320	-6.30	-6.09	7.24
Streaming	-576	-12.75	-8.05	-7.38
Transaction	0	0.00	-5.26	0.00
Average	-224	-4.76	-4.67	-0.03
Level 3				
Audio Interactive	-207	-13.33	-4.15	-4.12
Video Interactive	-253	-29.22	-29.55	-3.29
Texting Interactive	-425	-8.00	-4.00	-3.00
File Upload	-416	-8.73	-7.64	-8.55
File Download	-480	-3.24	-3.34	-4.51
Average	-356	-12.50	-9.74	-4.69
Level 4				
Hangout Texting	-108	-19.15	-27.66	-23.40
OneDrive Download	588	-0.32	-0.22	-0.11
Audio Streaming	-57	-3.49	0.00	0.00
Gaming	-96	-9.09	-9.09	-9.09
MyDrive Download	0	0.00	-0.28	0.00
Messenger Video	-9	-2.19	-3.07	-2.19
Video Streaming	-15	-10.57	-10.57	-10.93
Download App	-345	-2.06	-0.37	-0.56
hline MyDrive Upload	423	-3.53	-3.53	-1.18
Sensing	-20	-11.76	-11.76	0.00
Hangout Video	-144	-5.42	-5.42	-3.75
OneDrive Upload	-448	-15.57	-20.08	-19.67
Messenger Audio	0	0.00	0.00	0.00
Skype Video	0	0.00	-12.50	0.00
Skype Texting	-621	-13.79	-13.79	-3.45
Dropbox Upload	648	0.00	-0.68	0.00
Skype Audio	-20	-0.13	-0.13	0.00
System Update	-112	-2.50	2.50	2.50
Hangout Audio	-561	-34.85	-42.05	-12.63
Dropbox Download	-627	-17.13	-10.84	-9.44
Actuation	-345	-29.27	-4.88	9.76
Messenger Texting	-48	-16.67	-8.33	0.00
Average	-8	-8.98	-8.31	-3.82

Table 6.3: Effect of choosing optimal n based on the MI measure

Data	Training Acc Enhancement			
	N	Ind(%)	Ens(%)	Ens test(%)
Level 1				
All	0	0.00	0.00	0.00
Level 2				
Bulk Data Transfer	160	-0.32	1.13	0.41
Interactive	0	0.00	-13.33	0.00
Streaming	-264	-1.34	-0.67	0.00
Transaction	308	-29.82	-5.26	0.00
Average	51	-7.87	-4.53	0.10
Level 3				
Audio Interactive	420	-13.76	0.03	0.06
Video Interactive	111	-24.17	-26.44	-0.19
Texting Interactive	-41	-5.00	-3.00	-2.00
File Upload	0	0.00	0.91	0.00
File Download	255	-0.18	1.87	0.70
Average	149	-8.62	-5.33	-0.28
Level 4				
Hangout Texting	0	0.00	-4.26	0.00
OneDrive Download	-187	-0.54	-0.32	-0.22
Audio Streaming	-40	-3.49	0.00	0.00
Gaming	-96	-9.09	-9.09	-9.09
MyDrive Download	-255	-0.42	-0.70	-0.42
Messenger Video	39	-8.33	-0.88	0.00
Video Streaming	-15	-10.57	-10.57	-10.93
Download App	-297	-3.36	0.37	0.19
MyDrive Upload	264	-4.71	-3.53	-1.18
Sensing	253	0.00	-11.76	0.00
Hangout Video	99	-3.33	-1.67	0.00
OneDrive Upload	92	-1.64	-0.82	-0.41
Messenger Audio	144	-1.77	-1.77	-1.77
Skype Video	0	0.00	-12.50	0.00
Skype Texting	-609	-10.34	-3.45	6.90
Dropbox Upload	144	-0.68	0.00	0.68
Skype Audio	-27	-0.13	-0.13	0.00
System Update	455	-7.50	0.00	0.00
Hangout Audio	-576	-29.55	-41.92	-12.50
Dropbox Download	0	0.00	-1.40	0.00
Actuation	168	-36.59	-14.63	0.00
Messenger Texting	-39	-8.33	-8.33	0.00
Average	-21	-6.38	-5.79	-1.31

Table 6.4: Effect of choosing optimal n based on the training accuracy

Data	Confidence Enhancement			
	N	Ind(%)	Ens(%)	Ens test(%)
Level 1				
All	-680	-3.44	-1.29	-1.29
Level 2				
Bulk Data Transfer	76	-1.34	0.89	0.16
Interactive	-360	-6.40	-6.93	6.40
Streaming	-525	-5.37	-4.70	-4.03
Transaction	0	0.00	-5.26	0.00
Average	-202	-3.28	-4.00	0.63
Level 3				
Audio Interactive	-175	-8.80	-7.69	-7.67
Video Interactive	-189	-26.48	-28.06	-1.81
Texting Interactive	-416	-6.00	-3.00	-2.00
File Upload	0	0.00	0.91	0.00
File Download	255	-0.18	1.87	0.70
Average	-105	-8.29	-7.20	-2.15
Level 4				
Hangout Texting	-95	-8.51	-25.53	-21.28
OneDrive Download	588	-0.32	-0.22	-0.11
Audio Streaming	48	-3.49	0.00	0.00
Gaming	-96	-9.09	-9.09	-9.09
MyDrive Download	-108	-0.14	-0.56	-0.28
Messenger Video	0	0.00	-0.88	0.00
Video Streaming	-15	-10.57	-10.57	-10.93
Download App	-345	-2.06	-0.37	-0.56
MyDrive Upload	423	-3.53	-3.53	-1.18
Sensing	-20	-11.76	-11.76	0.00
Hangout Video	99	-3.33	-1.67	0.00
OneDrive Upload	-403	-14.14	-10.86	-10.45
Messenger Audio	0	0.00	0.00	0.00
Skype Video	0	0.00	-12.50	0.00
Skype Texting	-600	-10.34	-3.45	6.90
Dropbox Upload	544	0.00	-0.68	0.00
Skype Audio	-20	-0.13	-0.13	0.00
System Update	-96	-5.00	0.00	0.00
Hangout Audio	-576	-29.55	-41.92	-12.50
Dropbox Download	-147	-1.75	-2.10	-0.70
Actuation	-345	-29.27	-4.88	9.76
Messenger Texting	-39	-8.33	-8.33	0.00
Average	-54	-6.88	-6.77	-2.29

Table 6.5: Effect of choosing optimal n based on the proposed confidence measure

	MI	Acc train	C	Acc test	MI diff	Acc train diff	C diff
Level 1							
All	0.64	6.76	0.82	6.76	-6.12	0.00	-5.94
Level 2							
Interactive	2.60	6.56	2.00	6.56	-3.95	0.00	-4.56
Streaming	0.78	5.05	1.51	8.57	-7.79	-3.52	-7.06
Bulk Data Transfer	2.58	3.63	3.09	2.58	0.00	1.05	0.51
Transaction	0.92	6.53	0.92	0.92	0.00	5.62	0.00
Average	1.72	5.44	1.88	4.66	-2.94	0.79	-2.78
Level 3							
Audio Interactive	1.20	8.90	1.80	4.20	-3.00	4.70	-2.39
Video Interactive	0.90	4.78	2.01	3.81	-2.91	0.97	-1.80
Texting Interactive	0.86	13.48	1.67	14.59	-13.73	-1.12	-12.93
File Upload	0.43	4.81	4.81	4.81	-4.38	0.00	0.00
File Download	0.41	5.35	5.35	3.69	-3.28	1.66	1.66
Average	0.76	7.46	3.13	6.22	-5.46	1.24	-3.09
Level 4							
Hangout Texting	1.77	5.17	2.29	5.17	-3.41	0.00	-2.88
OneDrive File Download	4.72	0.06	4.72	1.29	3.43	-1.23	3.43
Audio Streaming	2.32	2.86	5.59	4.10	-1.78	-1.24	1.49
Gaming Interactive	0.22	0.22	0.22	4.80	-4.58	-4.58	-4.58
MyDrive File Download	5.71	3.94	4.95	5.71	0.00	-1.77	-0.77
Messenger Video	0.44	1.36	0.74	0.74	-0.29	0.62	0.00
Video Streaming	0.37	0.37	0.37	0.47	-0.11	-0.11	-0.11
Download App	0.33	0.75	0.33	1.95	-1.61	-1.20	-1.61
MyDrive File Upload	2.97	2.45	2.97	1.58	1.39	0.87	1.39
Sensing	0.93	6.54	0.93	1.40	-0.47	5.14	-0.47
Hangout Video	1.95	3.92	3.92	3.16	-1.21	0.76	0.76
OneDrive File Upload	0.67	5.83	1.25	5.06	-4.39	0.76	-3.81
Messenger Audio	0.33	1.47	0.33	0.33	0.00	1.15	0.00
Skype Video	0.34	0.34	0.34	0.34	0.00	0.00	0.00
Skype Texting	0.25	1.00	1.82	8.70	-8.45	-7.70	-6.88
Dropbox File Upload	17.95	6.38	15.64	2.53	15.42	3.86	13.11
Skype Audio	0.35	0.17	0.35	0.89	-0.54	-0.72	-0.54
System Update	0.51	3.62	1.46	2.88	-2.37	0.74	-1.42
Hangout Audio	1.86	1.46	1.46	9.60	-7.74	-8.14	-8.14
Dropbox File Download	0.47	5.41	4.29	5.41	-4.95	0.00	-1.12
Actuation	0.91	7.42	0.91	6.49	-5.58	0.93	-5.58
Messenger Texting	1.04	2.10	2.10	5.44	-4.40	-3.33	-3.33
Average	2.11	2.86	2.59	3.55	-1.44	-0.69	-0.96

Table 6.6: Enhancement in terms of time

MI			Training			Confidence			Testing		
N	Ind(%)	Ens(%)	N	Ind(%)	Ens(%)	N	Ind(%)	Ens(%)	N	Ind(%)	Ens(%)
110	50.20	51.40	2610	67.40	80.40	510	70.60	82.00	1810	79.40	74.60
Enhancement											
MI				Training				Confidence			
N	Ind(%)	Ens (%)	Ens test(%)	N	Ind(%)	Ens (%)	Ens test(%)	N	Ind(%)	Ens (%)	Ens test(%)
2100	-12.00	1.00	5.80	2100	-12.00	1.00	5.80	-1300	-8.80	2.60	7.40

Table 6.7: Voice Results

in terms of time are proportional to the number of samples with a sampling rate of 8000HZ.

6.5 Formal Description and Discussions

6.5.1 Problem Description

Suppose we have a time series data that we want to classify. The problem is to define the optimal number of samples that should be considered to guarantee good classification accuracy while minimizing the time to wait for classification of the stream of data.

6.5.2 Proposed Solution

In the literature, many techniques have been proposed for features selection or reduction. However, these techniques if applied in the time series case could result in choosing features from the high dimension data (vector) without accounting for the cost of waiting to receive this data and thus the time to make the classification. Moreover, the importance of these features might be slightly marginally better than the low dimensional ones. In our case, we aim at searching in a sequential manner to have enough important features that guarantee good classification accuracy. In other terms, we search for the subset of the time-series data or subset dimension that guarantee optimal quality of information and classification accuracy. The empirical solution consists of training several models and choosing the model with highest validation accuracy. However, the validation accuracy does not always imply best testing accuracy. Therefore, there is a need to combining e the empirical results with a measure that reflects the quality of the data. To do so, we define a new measure, called model confidence that reflects the variation in the training accuracy and the average mutual information, as presented in (6.1), and (6.2). Thus, there are several question to answer:

- What not relying only on the mutual information?

- Why the average mutual information presents a maximum?
- Does the accuracy have a well-defined shape and in what conditions?
- What is the possibility of having low accuracy degradation than the optimal validation accuracy?

In the following, we present main properties of the proposed confidence measure:

Property 1: if the mutual information between the input vector $[x_1, \dots, x_N]$ and the label vector presents a global maximum at the point α , and the accuracy is increasing, implies $\exists \beta/\beta > \alpha$, for which $C(\beta) = \max_n C(n)$.

Analysis:

$$\delta_C(n) = \frac{\delta_{Acc}(n) + \delta_{MI}(n)}{2}$$

Thus, $C(n)$ can be:

- presenting a global maximum $\Rightarrow \delta_{C(\beta)} = 0 \Rightarrow \delta_{Acc}(\beta) + \delta_{MI}(\beta) = 0 \Rightarrow \delta_{Acc}(\beta) = -\delta_{MI}(\beta)$. Given that $Acc(n)$ is increasing $\Rightarrow \delta_{Acc}(\beta) > 0 \Rightarrow \delta_{MI}(\beta) < 0$. Consequently, the accuracy achieved relying on the confidence measure is greater than the one achieved by relying on the mutual information $Acc(\beta) > Acc(\alpha)$.
- monotonically increasing $\Rightarrow \delta_{C(\beta)} > 0 \Rightarrow \delta_{Acc}(\beta) + \delta_{MI}(\beta) > 0 \Rightarrow \delta_{Acc}(\beta) > -\delta_{MI}(\beta)$.
- monotonically increasing before α and fluctuating after it. In this case, the maximum is reached at α or later.

Property 2: Given a time series $[x(1, \dots), x_N]$, the average mutual information of the subset vectors increased until a value of n, β such as $MI(\beta)$ is maximum.

Analysis:

This can be explained by the fact that for small values of n increasing n results in having a larger vector with more information, until a certain n , where repeated patterns appear.

$$\begin{aligned}
MI(n) &= \frac{\sum_0^{n \times n} MI(x_i, Y)}{n \times n} \\
MI(n+1) &= \frac{\sum_0^{(n+1) \times (n+1)} MI(x_i, Y)}{(n+1) \times (n+1)} \\
MI(n+1) - MI(n) &= \frac{(n \times n) \times \sum_0^{(n+1) \times (n+1)} MI(x_i, Y) - (n+1) \times (n+1) \times \sum_0^{n \times n} MI(x_i, Y)}{(n \times n) \times (n+1) \times (n+1)} \\
&= \frac{(n \times n) \times \sum_0^{(n+1) \times (n+1)} MI(x_i, Y) - (2n+2) \times \sum_0^{n \times n} MI(x_i, Y)}{(n \times n) \times (n+1) \times (n+1)}
\end{aligned} \tag{6.3}$$

The first term in the numerator consists of the sum of $(2n+2) \times MI(x_i, Y)$, with $(n \times n) < i < ((n+1) \times (n+1))$, and thus, in total, we have $(n \times n) \times (2n+2) \times MI(x_i, Y)$, and the second term consists of $(n \times n) \times (2n+2) \times MI(x_i, Y)$, with $0 < i < (n \times n)$. Consequently, for the samples or packets after the patterns are repeated presents the same quantity of information or less and thus the first term is equal or less than the second term $\Rightarrow MI(n+1) - MI(n) \leq 0$.

Property 3: Given an ensemble of classifiers C_1, C_2, \dots, C_n with accuracy of classification $Acc_1, Acc_2, \dots, Acc_n$, the ensemble classification with average sum of the probabilistic output is most probably increasing.

Analysis: let us assume that the classifier output is a random variable that can take one of two values True (T), and False (F). In the worst case, the probability to have a true classification is equal to the probability of having false classification $P_{C_i}(F) = P_{C_i}(T)$. The ensemble accuracy is the number of correctly classified instances over the total number of instances and it can be written as follows: $Acc_{ensemble}(n) = \frac{Total_{True}(n)}{Total_{True}(n) + Total_{False}(n)}$. At $n+1$, the accuracy increase if the $Total_{True}(n+1) > Total_{True}(n)$ or $Total_{False}(n+1) < Total_{False}(n)$, given that $Total_{True}(n) + Total_{False}(n) = cst$. Considering a test instance, we have four possibilities when a new classifier C_n is added:

- a correctly classified instance remains correctly classified with probability $P_{C \Rightarrow C}$;
- a correctly classified instance becomes falsely classified with probability $P_{C \Rightarrow F}$;
- a falsely classified instance remains falsely classified with probability $P_{F \Rightarrow C}$;
- and a falsely classified instance becomes correctly classified with probability $P_{F \Rightarrow F}$.

The probability that $Total_{True}(n+1) \geq Total_{True}(n)$ is equal to:

$$\begin{aligned}
P(Total_{True}(n+1) \geq Total_{True}(n)) &= P_{C \Rightarrow C} + P_{C \Rightarrow F} \times P\left(\sum_0^n True_i > \sum_0^n False_i\right) \\
&\quad + P_{F \Rightarrow C} + P_{F \Rightarrow F} \times P\left(\sum_0^n True_i > \sum_0^n False_i\right)
\end{aligned} \tag{6.4}$$

While the probability that $Total_{True}(n+1) < Total_{True}(n)$ is equal to:

$$\begin{aligned}
P(Total_{True}(n+1) < Total_{True}(n)) &= P_{C \Rightarrow C} \times (1 - P\left(\sum_0^n True_i > \sum_0^n False_i\right)) + P_{C \Rightarrow F} \\
&\quad + P_{F \Rightarrow C} \times (1 - P\left(\sum_0^n True_i > \sum_0^n False_i\right)) + P_{F \Rightarrow F}
\end{aligned} \tag{6.5}$$

Thus, the difference between $P(Total_{True}(n+1) \geq Total_{True}(n))$ and $P(Total_{True}(n+1) < Total_{True}(n))$ can be written as follows:

$$\begin{aligned}
Diff &= P(Total_{True}(n+1) \geq Total_{True}(n)) - P(Total_{True}(n+1) < Total_{True}(n)) = \\
&= P_{C \Rightarrow C} \times P\left(\sum_0^n True_i > \sum_0^n False_i\right) + P_{C \Rightarrow F} \times (P\left(\sum_0^n True_i > \sum_0^n False_i\right) - 1) + \\
&= P_{F \Rightarrow C} \times P\left(\sum_0^n True_i > \sum_0^n False_i\right) + P_{F \Rightarrow F} \times (P\left(\sum_0^n True_i > \sum_0^n False_i\right) - 1) \\
&= (P_{C \Rightarrow C} + P_{F \Rightarrow C}) \times P\left(\sum_0^n True_i > \sum_0^n False_i\right) - (P_{C \Rightarrow F} \\
&\quad + P_{F \Rightarrow F}) \times P\left(\sum_0^n True_i > \sum_0^n False_i\right)
\end{aligned} \tag{6.6}$$

In the worst case, the probability of the four possibilities are equally probable: $P_{C \Rightarrow C} = P_{C \Rightarrow F} = P_{F \Rightarrow C} = P_{F \Rightarrow F}$ and in this case:

- if $P(\sum_0^n True_i > \sum_0^n False_i) = P(\sum_0^n True_i < \sum_0^n False_i)$, the accuracy does not increase.
- if $P(\sum_0^n True_i > \sum_0^n False_i) > P(\sum_0^n True_i < \sum_0^n False_i)$, the accuracy increases.

- if $P(\sum_0^n True_i > \sum_0^n False_i) < P(\sum_0^n True_i < \sum_0^n False_i)$, the accuracy decreases.

Thus, in the worst case, the accuracy increases or remains the same with a probability of $2/3$.

Property 4: Given an ensemble of classifiers C_1, C_2, \dots, C_n with accuracy of classification $Acc_1, Acc_2, \dots, Acc_n$, $\exists \beta$ after which the ensemble accuracy remains constant or decreases.

Analysis: As stated above the accuracy of the ensemble method is calculated as follows: $Acc_{ensemble}(n) = \frac{Total_{True}(n)}{Total_{True}(n) + Total_{False}(n)}$. After a certain number of n , the number of classifiers correctly classifying an instance is sufficient to guarantee a correct classification if $P_{C \Rightarrow F} = 0.5$.

6.6 Conclusion

In this chapter, we proposed a model confidence measure that helps in choosing a value of n that optimize the testing accuracy while minimizing the training overhead, and the testing response time. To do so, we relied on the mutual information between the packets features and the class vector and the training accuracy to define a model confidence measure. This measure helps in choosing the optimal value of n that guarantees a good accuracy level. However, in some cases, real-time applications impose very low response time and thus the classification at early stages is highly required. In this case, we propose an ensemble classification method that considers the average of the classification results given by the set of successive classifiers of different dimensions (i.e. successive values of n). The classification results show that the proposed methods present enhancement in terms of accuracy compared to the individual classifiers. Combining the confidence measure stopping criteria and the ensemble classification, the accuracy of the model is enhanced at low values of n .

Chapter 7

Conclusion

Internet traffic is in a continuous evolution. With the emergence of the IoT, billions of things will be connected to the Internet. This poses new challenges given the high scale, heterogeneity, big amount of generated data, and security and privacy requirements. In this context, new techniques are needed to manage both QoS and security in the future network. From a QoS perspective, the traditional techniques/protocols for QoS guarantee suffer from their deployment and management complexity. Besides, from a security perspective, the traditional IDSes rely on pre-configured rules, presenting also management complexity. In addition, detecting unknown attacks require new IDS techniques. Machine Learning is proposed as a solution to these limitations. In this thesis, we aimed at applying DL for traffic classification based on different QoS and security requirements. To do so, we started by defining a hierarchical set of classes presenting different levels of granularity. In addition, we proposed a new data representation method to transform the flows into RGB images. Applying different DL based architectures, CNN gives the best results. Relying on statistical features, we show that our representation method is immune towards traffic encryption or anonymization. However, this does prevent adversarial attacks that might manipulate the traffic characteristics like packets sizes (padding) and inter-arrival time (shaping), and thus a traffic classifier should account for such attacks. Even if in some cases the recovery of original traffic is not possible, the detection is essential to avoid mis-classification. In this context, we considered defending two main obfuscation techniques, namely: mutation and morphing. Mutation aims at modifying the traffic characteristics in the aim of confusing the classifier by padding and shaping the traffic. On the other hand, morphing aims at imitating the traffic characteristics to make a flow look like another flow. This presents a critical security issue, when the attackers morph their traffic to look like normal traffic to avoid detection. To defend mutation we proposed a generative adversarial DL framework consisting of a denoising autoencoder and a discriminator. The denoiser is to de-mutate traffic and the discriminator to detect mutation. To defend morphing, we rely on variational autoencoder to generate morphed

traffic and the discriminator part of GAN is used for fake traffic detection. Last but not least, we worked on optimizing the choice of the number of packets to achieve good accuracy results. In this aim, we rely on mutual information and the training accuracy to propose a new confidence measure. Mutual information is computed between the features and the labels, while increasing n . The results show that we can decide to stop at a certain n , where the confidence is maximum to save time and achieve good accuracy. On the other hand, we propose a new ensemble method consisting of successive classifiers to enhance the classification performance. The classification results at precedent values of n are aggregated in proportion to the individual models confidence. The weighted average voting system is shown to achieve better accuracy than the individual classifiers. This model can be applied for any type of time series data to meet the compromise between response time and classification accuracy.

As a future work, we aim to integrate all the proposed modules together and evaluate the error rate, when the same traffic is passed to the proposed classification framework. In this case, a classification confidence measure should be developed based on the decision of each module. In other words, the detection of mutation or morphing in some parts of the flows should affect the classification confidence. Moreover, this framework modules are tested offline, however, we need to work on a real implementation in a real network. Moreover, the issue of online training of the proposed framework should be considered, taking into account the change in the optimal value of n .

Bibliography

- [1] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff, “A brief history of the internet,” *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 5, pp. 22–31, 2009.
- [2] O. Salman, I. Elhajj, A. Chehab, and A. Kayssi, “Iot survey: An sdn and fog computing perspective,” *Computer Networks*, vol. 143, pp. 221–246, 2018.
- [3] B. Park, Y. Won, J. Chung, M.-s. Kim, and J. W.-K. Hong, “Fine-grained traffic classification based on functional separation,” *International Journal of Network Management*, vol. 23, no. 5, pp. 350–381, 2013.
- [4] G. Aceto, A. Dainotti, W. De Donato, and A. Pescapé, “Portload: taking the best of two worlds in traffic classification,” in *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, pp. 1–5, IEEE, 2010.
- [5] A. Tongaonkar, R. Keralapura, and A. Nucci, “Challenges in network application identification.” in *LEET*, 2012.
- [6] D. Qin, J. Yang, J. Wang, and B. Zhang, “Ip traffic classification based on machine learning,” in *2011 IEEE 13th International Conference on Communication Technology*, pp. 882–886, IEEE, 2011.
- [7] J. Dromard, P. Owezarski, V. Mozo, A. Ordozgoiti, and B. Vakaruk, “Deliverable algorithms description: Traffic pattern evolution and unsupervised network anomaly detection ontic d4. 2,” 2016.
- [8] N. Namdev, S. Agrawal, and S. Silkari, “Recent advancement in machine learning based internet traffic classification,” *Procedia Computer Science*, vol. 60, pp. 784–791, 2015.
- [9] T. S. Tabatabaei, M. Adel, F. Karray, and M. Kamel, “Machine learning-based classification of encrypted internet traffic,” in *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pp. 578–592, Springer, 2012.

- [10] G. Cheng and Y. Hu, “Encrypted traffic identification based on n-gram entropy and cumulative sum test,” in *Proceedings of the 13th International Conference on Future Internet Technologies*, p. 9, ACM, 2018.
- [11] B. Niemczyk and P. Rao, “Identification over encrypted channels,” *Black-Hat USA*, 2014.
- [12] R. Alshammari and A. N. Zincir-Heywood, “Machine learning based encrypted traffic classification: Identifying ssh and skype,” in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–8, IEEE, 2009.
- [13] R. Alshammari and A. N. Zincir-Heywood, “How robust can a machine learning approach be for classifying encrypted voip?,” *Journal of Network and Systems Management*, vol. 23, no. 4, pp. 830–869, 2015.
- [14] L. Bernaille and R. Teixeira, “Early recognition of encrypted applications,” in *International Conference on Passive and Active Network Measurement*, pp. 165–175, Springer, 2007.
- [15] Z. Cao, G. Xiong, Y. Zhao, Z. Li, and L. Guo, “A survey on encrypted traffic classification,” in *International Conference on Applications and Techniques in Information Security*, pp. 73–81, Springer, 2014.
- [16] D. J. Arndt and A. N. Zincir-Heywood, “A comparison of three machine learning techniques for encrypted network traffic analysis,” in *2011 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 107–114, IEEE, 2011.
- [17] C. Gu, S. Zhang, and Y. Sun, “Realtime encrypted traffic identification using machine learning,” *JSW*, vol. 6, no. 6, pp. 1009–1016, 2011.
- [18] G. He, B. Xu, L. Zhang, and H. Zhu, “Mobile app identification for encrypted network flows by traffic correlation,” *International Journal of Distributed Sensor Networks*, vol. 14, no. 12, p. 1550147718817292, 2018.
- [19] Z. Hejun and Z. Liehuang, “Encrypted network behaviors identification based on dynamic time warping and k-nearest neighbor,” *Cluster Computing*, pp. 1–10, 2017.
- [20] J. Liu, Y. Fu, J. Ming, Y. Ren, L. Sun, and H. Xiong, “Effective and real-time in-app activity analysis in encrypted internet traffic streams,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 335–344, ACM, 2017.

- [21] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, “Deep packet: A novel approach for encrypted traffic classification using deep learning,” *Soft Computing*, pp. 1–14, 2017.
- [22] E. Mahdavi, A. Fanian, and H. Hassannejad, “Encrypted traffic classification using statistical features,” *ISeCure*, vol. 10, no. 1, 2018.
- [23] Q. Wang, A. Yahyavi, B. Kemme, and W. He, “I know what you did on your smartphone: Inferring app usage over encrypted data traffic,” in *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 433–441, IEEE, 2015.
- [24] Q. Xu, Y. Liao, S. Miskovic, Z. M. Mao, M. Baldi, A. Nucci, and T. Andrews, “Automatic generation of mobile app signatures from traffic observations,” in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 1481–1489, IEEE, 2015.
- [25] S. Rezaei and X. Liu, “Deep learning for encrypted traffic classification: An overview,” *IEEE communications magazine*, vol. 57, no. 5, pp. 76–81, 2019.
- [26] S. Leroux, S. Bohez, P.-J. Maenhaut, N. Meheus, P. Simoens, and B. Dhoedt, “Fingerprinting encrypted network traffic types using machine learning,” in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–5, IEEE, 2018.
- [27] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, “Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic,” in *10th {USENIX} Workshop on Offensive Technologies ({WOOT} 16)*, 2016.
- [28] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, “Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic,” in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 439–454, IEEE, 2016.
- [29] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, “Robust smartphone app identification via encrypted network traffic analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 63–78, 2017.
- [30] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, “A survey of methods for encrypted traffic classification and analysis,” *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.
- [31] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, “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, IEEE, 2017.
- [32] Y. Fu, H. Xiong, X. Lu, J. Yang, and C. Chen, “Service usage classification with encrypted internet traffic in mobile messaging apps,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 11, pp. 2851–2864, 2016.
- [33] Y. Liu, J. Chen, P. Chang, and X. Yun, “A novel algorithm for encrypted traffic classification based on sliding window of flow’s first n packets,” in *2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCI)*, pp. 463–470, IEEE, 2017.
- [34] A. Dainotti, A. Pescape, and K. C. Claffy, “Issues and future directions in traffic classification,” *IEEE network*, vol. 26, no. 1, pp. 35–40, 2012.
- [35] A. C. Callado, C. A. Kamienski, G. Szabó, B. P. Gero, J. Kelner, S. F. Fernandes, and D. F. H. Sadok, “A survey on internet traffic identification.,” *IEEE Communications Surveys and Tutorials*, vol. 11, no. 3, pp. 37–52, 2009.
- [36] T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 56–76, 2008.
- [37] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, “A survey of payload-based traffic classification approaches,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2013.
- [38] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, “Towards the deployment of machine learning solutions in network traffic classification: a systematic survey,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1988–2014, 2018.
- [39] “Vpn 2016 — datasets — research — canadian institute for cybersecurity — unb.” <https://www.unb.ca/cic/datasets/vpn.html>. (Accessed on 10/02/2019).
- [40] “Tor 2017 — datasets — research — canadian institute for cybersecurity — unb.” <https://www.unb.ca/cic/datasets/tor.html>. (Accessed on 10/02/2019).
- [41] B. Yamansavascular, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil, “Application identification via network traffic classification,” in *2017 International Conference on Computing, Networking and Communications (ICNC)*, pp. 843–848, IEEE, 2017.

- [42] H. Huang, H. Deng, J. Chen, L. Han, and W. Wang, “Automatic multi-task learning system for abnormal network traffic detection.,” *International Journal of Emerging Technologies in Learning*, vol. 13, no. 4, 2018.
- [43] “Computer laboratory - data.” <https://www.cl.cam.ac.uk/research/srg/netos/projects/brasil/> (Accessed on 10/02/2019).
- [44] A. W. Moore and D. Zuev, “Internet traffic classification using bayesian analysis techniques,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 50–60, ACM, 2005.
- [45] F. Ertam and E. Avci, “A new approach for internet traffic classification: Ga-wk-elm,” *Measurement*, vol. 95, pp. 135–142, 2017.
- [46] Z. Li, R. Yuan, and X. Guan, “Accurate classification of the internet traffic based on the svm method,” in *2007 IEEE International Conference on Communications*, pp. 1373–1378, IEEE, 2007.
- [47] J. Cao, Z. Fang, G. Qu, H. Sun, and D. Zhang, “An accurate traffic classification model based on support vector machines,” *International Journal of Network Management*, vol. 27, no. 1, p. e1962, 2017.
- [48] T. Auld, A. W. Moore, and S. F. Gull, “Bayesian neural networks for internet traffic classification,” *IEEE Transactions on neural networks*, vol. 18, no. 1, pp. 223–239, 2007.
- [49] J. Ran, X. Kong, G. Lin, D. Yuan, and H. Hu, “A self-adaptive network traffic classification system with unknown flow detection,” in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1215–1220, IEEE, 2017.
- [50] S. Huang, K. Chen, C. Liu, A. Liang, and H. Guan, “A statistical-feature-based approach to internet traffic classification using machine learning,” in *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pp. 1–6, IEEE, 2009.
- [51] H. Shi, H. Li, D. Zhang, C. Cheng, and X. Cao, “An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification,” *Computer Networks*, vol. 132, pp. 81–98, 2018.
- [52] R. Yuan, Z. Li, X. Guan, and L. Xu, “An svm-based machine learning method for accurate internet traffic classification,” *Information Systems Frontiers*, vol. 12, no. 2, pp. 149–156, 2010.
- [53] B. Schmidt, D. Kountanis, and A. Al-Fuqaha, “Artificial immune system inspired algorithm for flow-based internet traffic classification,” in *2014*

- IEEE 6th International Conference on Cloud Computing Technology and Science*, pp. 664–667, IEEE, 2014.
- [54] Y. Ding, “Imbalanced network traffic classification based on ensemble feature selection,” in *2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pp. 1–4, IEEE, 2016.
- [55] G. Sun, T. Chen, Y. Su, and C. Li, “Internet traffic classification based on incremental support vector machines,” *Mobile Networks and Applications*, vol. 23, no. 4, pp. 789–796, 2018.
- [56] Y. Huang, Y. Li, and B. Qiang, “Internet traffic classification based on min-max ensemble feature selection,” in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3485–3492, IEEE, 2016.
- [57] S. Dong, D. Zhou, and W. Ding, “The study of network traffic identification based on machine learning algorithm,” in *2012 Fourth International Conference on Computational Intelligence and Communication Networks*, pp. 205–208, IEEE, 2012.
- [58] M. Dashevskiy and Z. Luo, “Two methods for reliable classification of network traffic,” *Progress in Artificial Intelligence*, vol. 1, no. 3, pp. 223–234, 2012.
- [59] “<https://wand.net.nz/old/wand/publications/barcelona-2001.pdf>.”
<https://wand.net.nz/old/wand/publications/barcelona-2001.pdf>. (Accessed on 10/02/2019).
- [60] P. Perera, Y.-C. Tian, C. Fidge, and W. Kelly, “A comparison of supervised machine learning algorithms for classification of communications network traffic,” in *International Conference on Neural Information Processing*, pp. 445–454, Springer, 2017.
- [61] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, “Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification,” in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pp. 135–148, ACM, 2004.
- [62] J. Erman, M. Arlitt, and A. Mahanti, “Traffic classification using clustering algorithms,” in *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pp. 281–286, ACM, 2006.
- [63] J. Erman, A. Mahanti, and M. Arlitt, “Qrp05-4: Internet traffic identification using machine learning,” in *IEEE Globecom 2006*, pp. 1–6, IEEE, 2006.

- [64] L. Peng, B. Yang, Y. Chen, and Z. Chen, “Effectiveness of statistical features for early stage internet traffic identification,” *International Journal of Parallel Programming*, vol. 44, no. 1, pp. 181–197, 2016.
- [65] L. Peng, H. Zhang, Y. Chen, and B. Yang, “Imbalanced traffic identification using an imbalanced data gravitation-based classification model,” *Computer Communications*, vol. 102, pp. 177–189, 2017.
- [66] F. Hernández-Campos, A. Nobel, F. Smith, and K. Jeffay, “Statistical clustering of internet communication patterns,” *computing science and statistics*, vol. 35, 2003.
- [67] N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, pp. 5–16, 2006.
- [68] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, “Flow clustering using machine learning techniques,” in *International workshop on passive and active network measurement*, pp. 205–214, Springer, 2004.
- [69] S. Zander, T. Nguyen, and G. Armitage, “Self-learning ip traffic classification based on statistical flow characteristics,” in *International Workshop on Passive and Active Network Measurement*, pp. 325–328, Springer, 2005.
- [70] S. Zander, T. Nguyen, and G. Armitage, “Automated traffic classification and application identification using machine learning,” in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN’05) 1*, pp. 250–257, IEEE, 2005.
- [71] A. Este, F. Gringoli, and L. Salgarelli, “Support vector machines for tcp traffic classification,” *Computer Networks*, vol. 53, no. 14, pp. 2476–2490, 2009.
- [72] J. Park, H.-R. Tyan, and C.-C. J. Kuo, “Ga-based internet traffic classification technique for qos provisioning,” in *2006 International Conference on Intelligent Information Hiding and Multimedia*, pp. 251–254, IEEE, 2006.
- [73] M. Dusi, F. Gringoli, and L. Salgarelli, “Ip traffic classification for qos guarantees: The independence of packets,” in *2008 Proceedings of 17th International Conference on Computer Communications and Networks*, pp. 1–8, IEEE, 2008.
- [74] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, “Internet traffic classification demystified: myths, caveats, and the best practices,” in *Proceedings of the 2008 ACM CoNEXT conference*, p. 11, ACM, 2008.

- [75] “mawi.wide.ad.jp.” <http://mawi.wide.ad.jp/mawi/>. (Accessed on 10/02/2019).
- [76] V. Carela-Español, P. Barlet-Ros, A. Bifet, and K. Fukuda, “A streaming flow-based technique for traffic classification applied to 12+ 1 years of internet traffic,” *Telecommunication Systems*, vol. 63, no. 2, pp. 191–204, 2016.
- [77] S. Mongkolluksamee, V. Visoottiviseth, and K. Fukuda, “Combining communication patterns & traffic patterns to enhance mobile traffic identification performance,” *Journal of Information Processing*, vol. 24, no. 2, pp. 247–254, 2016.
- [78] F. Allard, R. Dubois, P. Gompel, and M. Morel, “Tunneling activities detection using machine learning techniques,” *Journal of Telecommunications and Information Technology*, pp. 37–42, 2011.
- [79] H. Dong, G.-L. Sun, and D.-D. Li, “A hybrid method for network traffic classification,” in *Proceedings of 2013 2nd International Conference on Measurement, Information and Control*, vol. 1, pp. 653–656, IEEE, 2013.
- [80] F. Ghofrani, A. Keshavarz-Haddad, and A. Jamshidi, “A new probabilistic classifier based on decomposable models with application to internet traffic,” *Pattern Recognition*, vol. 77, pp. 1–11, 2018.
- [81] R. Raveendran and R. R. Menon, “A novel aggregated statistical feature based accurate classification for internet traffic,” in *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*, pp. 225–232, IEEE, 2016.
- [82] R. Wang, L. Shi, and B. Jennings, “Ensemble classifier for traffic in presence of changing distributions,” in *2013 IEEE Symposium on Computers and Communications (ISCC)*, pp. 000629–000635, IEEE, 2013.
- [83] D. M. Divakaran, L. Su, Y. S. Liau, and V. L. Thing, “Slic: Self-learning intelligent classifier for network traffic,” *Computer Networks*, vol. 91, pp. 283–297, 2015.
- [84] X. Chen, J. Zhang, Y. Xiang, and W. Zhou, “Traffic identification in semi-known network environment,” in *2013 IEEE 16th International Conference on Computational Science and Engineering*, pp. 572–579, IEEE, 2013.
- [85] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho, “Seven years and one day: Sketching the evolution of internet traffic,” in *IEEE INFOCOM 2009*, pp. 711–719, IEEE, 2009.

- [86] H. A. H. Ibrahim, O. R. A. Al Zuobi, M. A. Al-Namari, G. MohamedAli, and A. A. A. Abdalla, "Internet traffic classification using machine learning approach: Datasets validation issues," in *2016 Conference of Basic Sciences and Engineering Studies (SGCAC)*, pp. 158–166, IEEE, 2016.
- [87] R. Hasibi, M. Shokri, and M. Dehghan, "Augmentation scheme for dealing with imbalanced network traffic classification using deep learning," *arXiv preprint arXiv:1901.00204*, 2019.
- [88] "Nbar2 or next generation nbar - cisco." https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/network-based-application-recognition-nbar/qa_c67697963.html. (Accessed on 10/02/2019).
- [89] L. Peng, H. Zhang, B. Yang, M. Su, and Y. Chen, "On the effectiveness of packet sampling for early stage traffic identification," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 468–473, Dec 2016.
- [90] "Dpi engine - r&space 2 — deep packet inspection from ipoque gmbh, a rohde & schwarz company." <https://www.ipoque.com/products/dpi-engine-rspace-2>. (Accessed on 10/02/2019).
- [91] "Github - thomasbhatia/openspi: Openspi v.3.10." <https://github.com/thomasbhatia/OpenDPI>. (Accessed on 10/02/2019).
- [92] "Application layer packet classifier for linux." <http://17-filter.sourceforge.net/>. (Accessed on 10/02/2019).
- [93] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "ndpi: Open-source high-speed deep packet inspection," in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 617–622, IEEE, 2014.
- [94] S. Alcock and R. Nelson, "Libprotoident: traffic classification using lightweight packet inspection," *WAND Network Research Group, Tech. Rep.*, 2012.
- [95] Z. Aouini, A. Kortebi, and Y. Ghamri-Doudane, "Towards understanding residential internet traffic: From packets to services," in *2016 7th International Conference on the Network of the Future (NOF)*, pp. 1–7, IEEE, 2016.

- [96] M. Dusi, F. Gringoli, and L. Salgarelli, “Quantifying the accuracy of the ground truth associated with internet traffic traces,” *Computer Networks*, vol. 55, no. 5, pp. 1158–1167, 2011.
- [97] H. Alizadeh and A. Zúquete, “Traffic classification for managing applications networking profiles,” *Security and Communication Networks*, vol. 9, no. 14, pp. 2557–2575, 2016.
- [98] “Coralreef software suite.” <https://www.caida.org/tools/measurement/coralreef/>. (Accessed on 10/02/2019).
- [99] F. Dehghani, N. Movahhedinia, M. R. Khayyambashi, and S. Kianian, “Real-time traffic classification based on statistical and payload content features,” in *2010 2nd International Workshop on Intelligent Systems and Applications*, pp. 1–4, IEEE, 2010.
- [100] “Cisco visual networking index: Forecast and trends, 20172022 white paper - cisco.” <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html>. (Accessed on 10/02/2019).
- [101] B. M. A. Abdalla, H. A. Jamil, M. Hamdan, J. S. Bassi, I. Ismail, and M. N. Marsono, “Multi-stage feature selection for on-line flow peer-to-peer traffic identification,” in *Asian Simulation Conference*, pp. 509–523, Springer, 2017.
- [102] L. Peng, H. Zhang, B. Yang, and Y. Chen, “Feature evaluation for early stage internet traffic identification,” in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 511–525, Springer, 2014.
- [103] Y. Aun, S. Manickam, and S. Karuppayah, “A review on features’ robustness in high diversity mobile traffic classifications,” *International journal of communication networks and information security*, vol. 9, no. 2, p. 294, 2017.
- [104] M. Shafiq, X. Yu, and D. Wang, “Robust feature selection for im applications at early stage traffic classification using machine learning algorithms,” in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 239–245, IEEE, 2017.
- [105] A. Dainotti, A. Pescapé, and H.-c. Kim, “Traffic classification through joint distributions of packet-level statistics,” in *2011 IEEE Global Telecommunications Conference-GLOBECOM 2011*, pp. 1–6, IEEE, 2011.

- [106] S. Filiposka and I. Mishkovski, “Smartphone users traffic characteristics and modelling,” *Transactions on Networks and Communications*, vol. 1, no. 1, pp. 14–40, 2013.
- [107] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “BlinC: multilevel traffic classification in the dark,” in *ACM SIGCOMM computer communication review*, vol. 35, pp. 229–240, ACM, 2005.
- [108] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos, “Profiling the end host,” in *International Conference on Passive and Active Network Measurement*, pp. 186–196, Springer, 2007.
- [109] J. Cao, A. Chen, I. Widjaja, and N. Zhou, “Online identification of applications using statistical behavior analysis,” in *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pp. 1–6, IEEE, 2008.
- [110] M. Meiss, F. Menczer, and A. Vespignani, “Properties and evolution of internet traffic networks from anonymized flow data,” *ACM Transactions on Internet Technology (TOIT)*, vol. 10, no. 4, p. 15, 2011.
- [111] S.-W. Lee, J.-S. Park, H.-S. Lee, and M.-S. Kim, “A study on smart-phone traffic analysis,” in *2011 13th Asia-Pacific Network Operations and Management Symposium*, pp. 1–7, IEEE, 2011.
- [112] J. Y. Chung, Y. Choi, B. Park, and J. W.-K. Hong, “Measurement analysis of mobile traffic in enterprise networks,” in *2011 13th Asia-Pacific Network Operations and Management Symposium*, pp. 1–4, IEEE, 2011.
- [113] B. Mitevski and S. Filiposka, “Smartphone traffic review,” in *International Conference on ICT Innovations*, pp. 291–301, Springer, 2013.
- [114] T. Okabe, T. Kitamura, and T. Shizuno, “Statistical traffic identification method based on flow-level behavior for fair voip service,” in *1st IEEE Workshop on VoIP Management and Security, 2006.*, pp. 35–40, IEEE, 2006.
- [115] Y. Hu, D.-M. Chiu, and J. C. Lui, “Application identification based on network behavioral profiles,” in *2008 16th International Workshop on Quality of Service*, pp. 219–228, IEEE, 2008.
- [116] K. Yu, Y. Liu, L. Qing, B. Wang, and Y. Cheng, “Positive and unlabeled learning for user behavior analysis based on mobile internet traffic data,” *IEEE Access*, vol. 6, pp. 37568–37580, 2018.
- [117] A. Moore, D. Zuev, and M. Crogan, “Discriminators for use in flow-based classification,” tech. rep., 2013.

- [118] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, “Acas: automated construction of application signatures,” in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pp. 197–202, ACM, 2005.
- [119] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, “Analyzing android encrypted network traffic to identify user actions,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, 2015.
- [120] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and A. S. Uluagac, “Peek-a-boo: I see your smart home activities, even encrypted!,” *arXiv preprint arXiv:1808.02741*, 2018.
- [121] Q. Xu, T. Andrews, Y. Liao, S. Miskovic, Z. M. Mao, M. Baldi, and A. Nucci, “Flowr: a self-learning system for classifying mobile application traffic,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 569–570, 2014.
- [122] M. Hur and M.-S. Kim, “Towards smart phone traffic classification,” in *2012 14th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–4, IEEE, 2012.
- [123] A. Murgia, G. Ghidini, S. P. Emmons, and P. Bellavista, “Lightweight internet traffic classification: A subject-based solution with word embeddings,” in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 1–8, IEEE, 2016.
- [124] R. Gonzalez, F. Manco, A. Garcia-Duran, J. Mendes, F. Huici, S. Niccolini, and M. Niepert, “Net2vec: Deep learning for the network,” *arXiv preprint arXiv:1705.03881*, 2017.
- [125] Y.-H. Goo, K.-S. Shim, S.-K. Lee, and M.-S. Kim, “Payload signature structure for accurate application traffic classification,” in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–4, IEEE, 2016.
- [126] J. Nowak, M. Korytkowski, R. Nowicki, R. Scherer, and A. Siwocha, “Random forests for profiling computer network users,” in *International Conference on Artificial Intelligence and Soft Computing*, pp. 734–739, Springer, 2018.
- [127] Z. Zhang, Z. Zhang, P. P. Lee, Y. Liu, and G. Xie, “Toward unsupervised protocol feature word extraction,” *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 10, pp. 1894–1906, 2014.
- [128] G. Maier, F. Schneider, and A. Feldmann, “A first look at mobile hand-held device traffic,” in *International Conference on Passive and Active Network Measurement*, pp. 161–170, Springer, 2010.

- [129] Z. Chen, B. Yu, Y. Zhang, J. Zhang, and J. Xu, "Automatic mobile application traffic identification by convolutional neural networks," in *2016 IEEE Trustcom/BigDataSE/ISPA*, pp. 301–307, IEEE, 2016.
- [130] Z. Chen, K. He, J. Li, and Y. Geng, "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, IEEE, 2017.
- [131] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2017.
- [132] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *International Conference on Neural Information Processing*, pp. 858–866, Springer, 2017.
- [133] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, vol. 24, 2015.
- [134] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, pp. 712–717, IEEE, 2017.
- [135] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, "A multi-level internet traffic classifier using deep learning," in *2018 9th International Conference on the Network of the Future (NOF)*, pp. 68–75, IEEE, 2018.
- [136] G. Aceto, D. Ciunzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning," in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1–8, IEEE, 2018.
- [137] Z. Liu, R. Wang, N. Japkowicz, Y. Cai, D. Tang, and X. Cai, "Mobile app traffic flow feature extraction and selection for improving classification robustness," *Journal of Network and Computer Applications*, vol. 125, pp. 190–208, 2019.
- [138] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "Networkprofiler: Towards automatic fingerprinting of android apps," in *2013 Proceedings IEEE INFOCOM*, pp. 809–817, IEEE, 2013.
- [139] B. Schmidt, A. Al-Fuqaha, A. Gupta, and D. Kountanis, "Optimizing an artificial immune system algorithm in support of flow-based internet traffic classification," *Applied Soft Computing*, vol. 54, pp. 1–22, 2017.

- [140] S. Mongkolluksamee, V. Visoottiviseth, and K. Fukuda, “Enhancing the performance of mobile traffic identification with communication patterns,” in *2015 IEEE 39th Annual Computer Software and Applications Conference*, vol. 2, pp. 336–345, IEEE, 2015.
- [141] Z. Yuan and C. Wang, “An improved network traffic classification algorithm based on hadoop decision tree,” in *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*, pp. 53–56, IEEE, 2016.
- [142] L. Hu and L. Zhang, “Real-time internet traffic identification based on decision tree,” in *World Automation Congress 2012*, pp. 1–3, IEEE, 2012.
- [143] J. Lingyu, L. Yang, W. Bailing, L. Hongri, and X. Guodong, “A hierarchical classification approach for tor anonymous traffic,” in *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, pp. 239–243, IEEE, 2017.
- [144] N. Jing, M. Yang, S. Cheng, Q. Dong, and H. Xiong, “An efficient svm-based method for multi-class network traffic classification,” in *30th IEEE International Performance Computing and Communications Conference*, pp. 1–8, IEEE, 2011.
- [145] Y. Hong, C. Huang, B. Nandy, and N. Seddigh, “Iterative-tuning support vector machine for network traffic classification,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 458–466, IEEE, 2015.
- [146] M. Sabzekar, M. H. Y. Moghaddam, and M. Naghibzadeh, “Tcp traffic classification using relaxed constraints support vector machines,” in *Integration of Practice-Oriented Knowledge Technology: Trends and Perspectives*, pp. 129–139, Springer, 2013.
- [147] W. Zhou, L. Dong, L. Bic, M. Zhou, and L. Chen, “Internet traffic classification using feed-forward neural network,” in *2011 International Conference on Computational Problem-Solving (ICCP)*, pp. 641–646, IEEE, 2011.
- [148] S. Dong, D. Zhou, W. Zhou, W. Ding, and J. Gong, “Research on network traffic identification based on improved bp neural network,” *Applied Mathematics & Information Sciences*, vol. 7, no. 1, 2013.
- [149] D. Smit, K. Millar, C. Page, A. Cheng, H.-G. Chew, and C.-C. Lim, “Looking deeper: Using deep learning to identify internet communications traffic,” in *2017 Australasian Conference of Undergraduate Research (ACUR)*, 2017.

- [150] Y. Liu, S. Zhang, B. Ding, X. Li, and Y. Wang, “A cascade forest approach to application classification of mobile traces,” in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2018.
- [151] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, “State-of-the-art deep learning: Evolving machine intelligence toward tomorrow’s intelligent network traffic control systems,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.
- [152] D. Hahn, N. Apthorpe, and N. Feamster, “Detecting compressed clear-text traffic from consumer internet of things devices,” *arXiv preprint arXiv:1805.02722*, 2018.
- [153] R. Vinayakumar, K. Soman, and P. Poornachandran, “Applying convolutional neural network for network intrusion detection,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1222–1228, IEEE, 2017.
- [154] C. Zhang, P. Patras, and H. Haddadi, “Deep learning in mobile and wireless networking: A survey,” *IEEE Communications Surveys & Tutorials*, 2019.
- [155] D. Gugelmann, “Deep learning and machine learning for network traffic analysis,” *SIGS Technology Conference 2017 Hacking Day*, 2017.
- [156] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [157] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
- [158] “Cs231n convolutional neural networks for visual recognition.” <http://cs231n.github.io/convolutional-networks/>. (Accessed on 10/02/2019).
- [159] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
- [160] S. Rezaei and X. Liu, “How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets,” *arXiv preprint arXiv:1812.09761*, 2018.

- [161] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, “Network traffic classifier with convolutional and recurrent neural networks for internet of things,” *IEEE Access*, vol. 5, pp. 18042–18050, 2017.
- [162] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [163] “Recurrent neural network - wikipedia.” https://en.wikipedia.org/wiki/Recurrent_neural_network. (Accessed on 10/02/2019).
- [164] A. Munther, R. R. Othman, A. S. Alsaadi, and M. Anbar, “A performance study of hidden markov model and random forest in internet traffic classification,” in *Information Science and Applications (ICISA) 2016*, pp. 319–329, Springer, 2016.
- [165] A. Dainotti, A. Pescape, P. S. S. Rossi, G. Iannello, F. Palmieri, and G. Ventre, “Qrp07-2: An hmm approach to internet traffic modeling,” in *IEEE Globecom 2006*, pp. 1–6, IEEE, 2006.
- [166] J. E. B. Maia and R. Holanda Filho, “Internet traffic classification using a hidden markov model,” in *2010 10th International Conference on Hybrid Intelligent Systems*, pp. 37–42, IEEE, 2010.
- [167] S. E. Gómez, B. C. Martínez, A. J. Sánchez-Esguevillas, and L. H. Callejo, “Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal,” *Computer Networks*, vol. 127, pp. 68–80, 2017.
- [168] C. Wang, T. Xu, and X. Qin, “Network traffic classification with improved random forest,” in *2015 11th International Conference on Computational Intelligence and Security (CIS)*, pp. 78–81, IEEE, 2015.
- [169] G. Sun, L. Liang, T. Chen, F. Xiao, and F. Lang, “Network traffic classification based on transfer learning,” *Computers & electrical engineering*, vol. 69, pp. 920–927, 2018.
- [170] M. Shafiq, X. Yu, and D. Wang, “Network traffic classification using machine learning algorithms,” in *International Conference on Intelligent and Interactive Systems and Applications*, pp. 621–627, Springer, 2017.
- [171] A. K. J. Michael, E. Valla, N. S. Neggatu, and A. W. Moore, “Network traffic classification via neural networks,” tech. rep., University of Cambridge, Computer Laboratory, 2017.
- [172] B. Wang, J. Zhang, Z. Zhang, W. Luo, and D. Xia, “Traffic identification in big internet data,” in *Big Data Concepts, Theories, and Applications*, pp. 129–156, Springer, 2016.

- [173] M. Shafiq, X. Yu, A. K. Bashir, H. N. Chaudhry, and D. Wang, “A machine learning approach for feature selection traffic classification using security analysis,” *The Journal of Supercomputing*, vol. 74, no. 10, pp. 4867–4892, 2018.
- [174] M. Usama, J. Qadir, A. Raza, H. Arif, K.-L. A. Yau, Y. Elkhatib, A. Husain, and A. Al-Fuqaha, “Unsupervised machine learning for networking: Techniques, applications and research challenges,” *IEEE Access*, vol. 7, pp. 65579–65615, 2019.
- [175] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, “Can’t you hear me knocking: Identification of user actions on android apps via traffic analysis,” in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pp. 297–304, ACM, 2015.
- [176] G. Dewaele, Y. Himura, P. Borgnat, K. Fukuda, P. Abry, O. Michel, R. Fontugne, K. Cho, and H. Esaki, “Unsupervised host behavior classification from connection patterns,” *International Journal of Network Management*, vol. 20, no. 5, pp. 317–337, 2010.
- [177] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, “Identifying and discriminating between web and peer-to-peer traffic in the network core,” in *Proceedings of the 16th international conference on World Wide Web*, pp. 883–892, ACM, 2007.
- [178] J. Yuan, Z. Li, and R. Yuan, “Information entropy based clustering method for unsupervised internet traffic classification,” in *2008 IEEE International Conference on Communications*, pp. 1588–1592, IEEE, 2008.
- [179] C. Yang, F. Wang, and B. Huang, “Internet traffic classification using db-scan,” in *2009 WASE International Conference on Information Engineering*, vol. 2, pp. 163–166, IEEE, 2009.
- [180] J. Zhang, Z. Qian, G. Shou, and Y. Hu, “Traffic identification method based on on-line density based spatial clustering algorithm,” in *2010 2nd IEEE International Conference on Network Infrastructure and Digital Content*, pp. 270–274, IEEE, 2010.
- [181] L. Bernaille, R. Teixeira, and K. Salamatian, “Early application identification,” in *Proceedings of the 2006 ACM CoNEXT conference*, p. 6, ACM, 2006.
- [182] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, “Traffic classification on the fly,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, 2006.

- [183] M. Zhang, H. Zhang, B. Zhang, and G. Lu, “Encrypted traffic classification based on an improved clustering algorithm,” in *International Conference on Trustworthy Computing and Services*, pp. 124–131, Springer, 2012.
- [184] J. Höchst, L. Baumgärtner, M. Hollick, and B. Freisleben, “Unsupervised traffic flow classification using a neural autoencoder,” in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, pp. 523–526, IEEE, 2017.
- [185] D. Li, Y. Zhu, and W. Lin, “Traffic identification of mobile apps based on variational autoencoder network,” in *2017 13th International Conference on Computational Intelligence and Security (CIS)*, pp. 287–291, IEEE, 2017.
- [186] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, pp. 2672–2680, 2014.
- [187] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, “Efficient gan-based anomaly detection,” *arXiv preprint arXiv:1802.06222*, 2018.
- [188] D. Li, D. Chen, J. Goh, and S.-k. Ng, “Anomaly detection with generative adversarial networks for multivariate time series,” *arXiv preprint arXiv:1809.04758*, 2018.
- [189] L. Deecke, R. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft, “Image anomaly detection with generative adversarial networks,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 3–17, Springer, 2018.
- [190] S. Tripathi, Z. C. Lipton, and T. Q. Nguyen, “Correction by projection: Denoising images with generative adversarial networks,” *arXiv preprint arXiv:1803.04477*, 2018.
- [191] D. Warde-Farley and Y. Bengio, “Improving generative adversarial networks with denoising feature matching,” 2016.
- [192] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, “Adversarial autoencoders,” *arXiv preprint arXiv:1511.05644*, 2015.
- [193] S. Ger and D. Klabjan, “Autoencoders and generative adversarial networks for anomaly detection for sequences,” *arXiv preprint arXiv:1901.02514*, 2019.
- [194] X. Wang, Y. Du, S. Lin, P. Cui, and Y. Yang, “Self-adversarial variational autoencoder with gaussian anomaly prior distribution for anomaly detection,” *arXiv preprint arXiv:1903.00904*, 2019.

- [195] H. S. Vu, D. Ueta, K. Hashimoto, K. Maeno, S. Pranata, and S. M. Shen, “Anomaly detection with adversarial dual autoencoders,” *arXiv preprint arXiv:1902.06924*, 2019.
- [196] L. Beggel, M. Pfeiffer, and B. Bischl, “Robust anomaly detection in images using adversarial autoencoders,” *arXiv preprint arXiv:1901.06355*, 2019.
- [197] A. Creswell and A. A. Bharath, “Denoising adversarial autoencoders,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 4, pp. 968–984, 2018.
- [198] R. Chalapathy and S. Chawla, “Deep learning for anomaly detection: A survey,” *arXiv preprint arXiv:1901.03407*, 2019.
- [199] S. Liu, J. Hu, S. Hao, and T. Song, “Improved em method for internet traffic classification,” in *2016 8th International Conference on Knowledge and Smart Technology (KST)*, pp. 13–17, IEEE, 2016.
- [200] Y. Zhao, J. Chen, G. You, and J. Teng, “Network traffic classification model based on mdl criterion,” in *Advanced Multimedia and Ubiquitous Engineering*, pp. 1–8, Springer, 2016.
- [201] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. T. Yang, “Internet traffic classification using constrained clustering,” *IEEE transactions on parallel and distributed systems*, vol. 25, no. 11, pp. 2932–2943, 2013.
- [202] M. Laner, P. Svoboda, and M. Rupp, “Detecting m2m traffic in mobile cellular networks,” in *IWSSIP 2014 Proceedings*, pp. 159–162, IEEE, 2014.
- [203] G. Szabó, J. Szüle, Z. Turányi, and G. Pongrácz, “Multi-level machine learning traffic classification system,” in *The Eleventh International Conference on Networks*, pp. 69–77, 2012.
- [204] Z. A. Shaikh and D. G. Harkut, “A novel framework for network traffic classification using unknown flow detection,” in *2015 Fifth International Conference on Communication Systems and Network Technologies*, pp. 116–121, IEEE, 2015.
- [205] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, “Robust network traffic classification,” *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 4, pp. 1257–1270, 2015.
- [206] J. Zhang, C. Chen, Y. Xiang, and W. Zhou, “Robust network traffic identification with unknown applications,” in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pp. 405–414, ACM, 2013.

- [207] J. Zhang, X. Chen, Y. Xiang, and W. Zhou, “Zero-day traffic identification,” in *Cyberspace Safety and Security*, pp. 213–227, Springer, 2013.
- [208] W. Li and A. W. Moore, “A machine learning approach for efficient traffic classification,” in *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 310–317, IEEE, 2007.
- [209] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and A. V. Vasilakos, “An effective network traffic classification method with unknown flow detection,” *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 133–147, 2013.
- [210] J. Zhang, Y. Xiang, W. Zhou, and Y. Wang, “Unsupervised traffic classification using flow statistical properties and ip packet payload,” *Journal of Computer and System Sciences*, vol. 79, no. 5, pp. 573–585, 2013.
- [211] T. Bakhshi and B. Ghita, “On internet traffic classification: A two-phased machine learning approach,” *Journal of Computer Networks and Communications*, vol. 2016, 2016.
- [212] T. Glennan, C. Leckie, and S. M. Erfani, “Improved classification of known and unknown network traffic flows using semi-supervised machine learning,” in *Australasian conference on information security and privacy*, pp. 493–501, Springer, 2016.
- [213] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, “Machine learning in software defined networks: Data collection and traffic classification,” in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pp. 1–5, IEEE, 2016.
- [214] A. Fahad, A. Almalawi, Z. Tari, K. Alharthi, F. S. Al Qahtani, and M. Cheriet, “Semtra: A semi-supervised approach to traffic flow labeling with minimal human effort,” *Pattern Recognition*, vol. 91, pp. 1–12, 2019.
- [215] C. Rotsos, J. Van Gael, A. W. Moore, and Z. Ghahramani, “Probabilistic graphical models for semi-supervised traffic classification,” in *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, pp. 752–757, ACM, 2010.
- [216] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, “Offline/realtime traffic classification using semi-supervised learning,” *Performance Evaluation*, vol. 64, no. 9-12, pp. 1194–1213, 2007.

- [217] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, "A graph-based consensus maximization approach for combining multiple supervised and unsupervised models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 15–28, 2011.
- [218] A. M. Almalawi, A. Fahad, Z. Tari, M. A. Cheema, and I. Khalil, " k nnwvc: An efficient k -nearest neighbors approach based on various-widths clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 68–81, 2015.
- [219] A. Vlăduțu, D. Comănesci, and C. Dobre, "Internet traffic classification based on flows' statistical properties with machine learning," *International Journal of Network Management*, vol. 27, no. 3, p. e1929, 2017.
- [220] X. Li, F. Qi, D. Xu, and X.-s. Qiu, "An internet traffic classification method based on semi-supervised support vector machine," in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–5, IEEE, 2011.
- [221] F. Qian, G.-m. Hu, and X.-m. Yao, "Semi-supervised internet network traffic classification using a gaussian mixture model," *AEU-International Journal of Electronics and Communications*, vol. 62, no. 7, pp. 557–564, 2008.
- [222] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 5–16, 2007.
- [223] G.-L. Sun, Y. Xue, Y. Dong, D. Wang, and C. Li, "An novel hybrid method for effectively classifying encrypted traffic," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pp. 1–5, IEEE, 2010.
- [224] H. Shi, H. Li, D. Zhang, C. Cheng, and W. Wu, "Efficient and robust feature extraction and selection for traffic classification," *Computer Networks*, vol. 119, pp. 1–16, 2017.
- [225] C.-N. Lu, C.-Y. Huang, Y.-D. Lin, and Y.-C. Lai, "High performance traffic classification based on message size sequence and distribution," *Journal of Network and Computer Applications*, vol. 76, pp. 60–74, 2016.
- [226] J. Kim, J. Hwang, and K. Kim, "High-performance internet traffic classification using a markov model and kullback-leibler divergence," *Mobile Information Systems*, vol. 2016, 2016.
- [227] Y. Wang and R. Nelson, "Identifying network application layer protocol with machine learning," *Proc. Passive and Active Network Measurement (PAM 09)*, Seoul, Korea, 2009.

- [228] A. W. Moore and K. Papagiannaki, “Toward the accurate identification of network applications,” in *International Workshop on Passive and Active Network Measurement*, pp. 41–54, Springer, 2005.
- [229] P. Wang, S.-C. Lin, and M. Luo, “A framework for qos-aware traffic classification using semi-supervised machine learning in sdns,” in *2016 IEEE International Conference on Services Computing (SCC)*, pp. 760–765, IEEE, 2016.
- [230] H. Jiang, A. W. Moore, Z. Ge, S. Jin, and J. Wang, “Lightweight application classification for network management,” in *Proceedings of the 2007 SIGCOMM workshop on Internet network management*, pp. 299–304, ACM, 2007.
- [231] T. Chen and Y. Zeng, “Classification of traffic flows into qos classes by unsupervised learning and knn clustering,” *KSII Trans. on Internet and Information Systems*, vol. 3, no. 2, pp. 134–146, 2009.
- [232] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song, “Flowqos: per-flow quality of service for broadband access networks,” tech. rep., Georgia Institute of Technology, 2015.
- [233] J. Park, H.-R. Tyan, and C.-C. J. Kuo, “Internet traffic classification for scalable qos provision,” in *2006 IEEE International Conference on Multimedia and Expo*, pp. 1221–1224, IEEE, 2006.
- [234] W. Zai-jian, Y.-n. Dong, H.-x. Shi, Y. Lingyun, and T. Pingping, “Internet video traffic classification using qos features,” in *2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, IEEE, 2016.
- [235] J. Yang, S. Zhang, X. Zhang, J. Liu, and G. Cheng, “Characterizing smartphone traffic with mapreduce,” in *2013 16th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 1–5, IEEE, 2013.
- [236] T. Bujlow, T. Riaz, and J. M. Pedersen, “A method for classification of network traffic based on c5.0 machine learning algorithm,” in *2012 international conference on computing, networking and communications (ICNC)*, pp. 237–241, IEEE, 2012.
- [237] A. Kortebi, Z. Aouini, C. Delahaye, J.-P. Javaudin, and Y. Ghamri-Doudane, “A platform for home network traffic monitoring,” in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 895–896, IEEE, 2017.

- [238] T. Iwai and A. Nakao, “Adaptive mobile application identification through in-network machine learning,” in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–6, IEEE, 2016.
- [239] M. Grajzer, M. Koziuk, P. Szczechowiak, and A. Pescapé, “A multi-classification approach for the detection and identification of ehealth applications,” in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–6, IEEE, 2012.
- [240] S. Valenti and D. Rossi, “Identifying key features for p2p traffic classification,” in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2011.
- [241] S. K. Baghel, K. Keshav, and V. R. Manepalli, “An investigation into traffic analysis for diverse data applications on smartphones,” in *2012 National Conference on Communications (NCC)*, pp. 1–5, IEEE, 2012.
- [242] A. Pektaş, “Proposal of machine learning approach for identification of instant messaging applications in raw network traffic,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 6, no. 2, pp. 97–102, 2018.
- [243] Y. Choi, J. Y. Chung, B. Park, and J. W.-K. Hong, “Automated classifier generation for application-level mobile traffic identification,” in *2012 IEEE Network Operations and Management Symposium*, pp. 1075–1081, IEEE, 2012.
- [244] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi, “Tracking down skype traffic,” in *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*, pp. 261–265, IEEE, 2008.
- [245] J. Kampeas, A. Cohen, and O. Gurewitz, “Traffic classification based on zero-length packets,” *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 1049–1062, 2018.
- [246] G. Aceto, D. Ciunzo, A. Montieri, and A. Pescapé, “Traffic classification of mobile apps through multi-classification,” in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2017.
- [247] I. Tsompanidis, A. H. Zahran, and C. J. Sreenan, “Mobile network traffic: A user behaviour model,” in *2014 7th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 1–8, IEEE, 2014.
- [248] L. Vassio, I. Drago, and M. Mellia, “Detecting user actions from http traces: Toward an automatic approach,” in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 50–55, IEEE, 2016.

- [249] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, “Iot sentinel: Automated device-type identification for security enforcement in iot,” in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2177–2184, IEEE, 2017.
- [250] O. Salman, L. Chaddad, I. H. Elhajj, A. Chehab, and A. Kayssi, “Pushing intelligence to the network edge,” in *2018 Fifth International Conference on Software Defined Systems (SDS)*, pp. 87–92, IEEE, 2018.
- [251] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, “Classifying iot devices in smart environments using network traffic characteristics,” *IEEE Transactions on Mobile Computing*, 2018.
- [252] B. Copos, K. Levitt, M. Bishop, and J. Rowe, “Is anybody home? inferring activity from smart home network traffic,” in *2016 IEEE Security and Privacy Workshops (SPW)*, pp. 245–251, IEEE, 2016.
- [253] K. Gopalratnam, S. Basu, J. Dunagan, and H. J. Wang, “Automatically extracting fields from unknown network protocols,” in *First Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML06)*, 2006.
- [254] S. Zhao, S. Chen, Y. Sun, Z. Cai, and J. Su, “Identifying known and unknown mobile application traffic using a multilevel classifier,” *Security and Communication Networks*, vol. 2019, 2019.
- [255] C. R. Kalmanek, S. Misra, and Y. R. Yang, *Guide to reliable internet services and applications*. Springer Science & Business Media, 2010.
- [256] Y. Xie, H. Deng, L. Peng, and Z. Chen, “Accurate identification of internet video traffic using byte code distribution features,” in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 46–58, Springer, 2018.
- [257] A. Canovas, J. M. Jimenez, O. Romero, and J. Lloret, “Multimedia data flow traffic classification using intelligent models based on traffic patterns,” *IEEE Network*, vol. 32, no. 6, pp. 100–107, 2018.
- [258] T. D. Nguyen, S. Marchal, M. Miettinen, N. Asokan, and A. Sadeghi, “Dïot: A self-learning system for detecting compromised iot devices,” *CoRR*, vol. *abs/1804.07474*, 2018.
- [259] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici, “Detection of unauthorized iot devices using machine learning techniques,” *arXiv preprint arXiv:1709.04647*, 2017.

- [260] S. Siby, R. R. Maiti, and N. Tippenhauer, “Iotscanner: Detecting and classifying privacy threats in iot neighborhoods,” *arXiv preprint arXiv:1701.05007*, 2017.
- [261] H. Kawai, S. Ata, N. Nakamura, and I. Oka, “Identification of communication devices from analysis of traffic patterns,” in *2017 13th International Conference on Network and Service Management (CNSM)*, pp. 1–5, IEEE, 2017.
- [262] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Z. Yang, “Automatic device classification from network traffic streams of internet of things,” in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pp. 1–9, IEEE, 2018.
- [263] P. Robyns, E. Marin, W. Lamotte, P. Quax, D. Singelée, and B. Preneel, “Physical-layer fingerprinting of lora devices using supervised and zero-shot learning,” in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pp. 58–63, ACM, 2017.
- [264] R. Das, A. Gadre, S. Zhang, S. Kumar, and J. M. Moura, “A deep learning approach to iot authentication,” in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.
- [265] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. R. Karger, “Infranet: Circumventing web censorship and surveillance.,” in *USENIX Security Symposium*, pp. 247–262, 2002.
- [266] “Protocol obfuscation - emule wiki.” http://wiki.emule-web.de/Protocol_obfuscation. (Accessed on 10/02/2019).
- [267] B. Wiley, “Dust: A blocking-resistant internet transport protocol,” *Technical report*. <http://blanu.net/Dust.pdf>, 2011.
- [268] Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov, “Censorspoof: asymmetric communication using ip spoofing for censorship-resistant web browsing,” in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 121–132, ACM, 2012.
- [269] Z. Weinberg, J. Wang, V. Yegneswaran, L. Briesemeister, S. Cheung, F. Wang, and D. Boneh, “Stegotorus: a camouflage proxy for the tor anonymity system,” in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 109–120, ACM, 2012.
- [270] H. Mohajeri Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, “Skypemorph: Protocol obfuscation for tor bridges,” in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 97–108, ACM, 2012.

- [271] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail,” in *2012 IEEE symposium on security and privacy*, pp. 332–346, IEEE, 2012.
- [272] “pluggable-transport/obfsproxy - pluggable transport for obfuscated traffic.” <https://gitweb.torproject.org/pluggable-transport/obfsproxy.git/tree/doc/obfs2/obfs2-protocol-spec.txt>. (Accessed on 10/02/2019).
- [273] “Flash proxies.” <https://crypto.stanford.edu/flashproxy/>. (Accessed on 10/02/2019).
- [274] P. Winter, T. Pulls, and J. Fuss, “Scramblesuit: A polymorph network protocol to circumvent censorship,” *arXiv preprint arXiv:1305.3199*, 2013.
- [275] A. Houmansadr, T. J. Riedl, N. Borisov, and A. C. Singer, “I want my voice to be heard: Ip over voice-over-ip for unobservable censorship circumvention.,” in *NDSS*, 2013.
- [276] “obfs3-protocol-spec.txt\obfs3\doc - pluggable-transport/obfsproxy - pluggable transport for obfuscated traffic.” <https://gitweb.torproject.org/pluggable-transport/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>. (Accessed on 10/02/2019).
- [277] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Protocol misidentification made easy with format-transforming encryption,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 61–72, ACM, 2013.
- [278] “doc/meek tor bug tracker & wiki.” <https://trac.torproject.org/projects/tor/wiki/doc/meek>. (Accessed on 10/02/2019).
- [279] J. Gardiner and S. Nagaraja, “Blindspot: Indistinguishable anonymous communications,” *arXiv preprint arXiv:1408.0784*, 2014.
- [280] J. Lv, C. Zhu, S. Tang, and C. Yang, “Deepflow: Hiding anonymous communication traffic in p2p streaming networks,” *Wuhan University Journal of Natural Sciences*, vol. 19, no. 5, pp. 417–425, 2014.
- [281] S. Li, M. Schliep, and N. Hopper, “Facet: Streaming over videoconferencing for censorship circumvention,” in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pp. 163–172, ACM, 2014.
- [282] “Github - yawning/obfs4: The obfourscator (courtesy mirror).” <https://github.com/Yawning/obfs4>. (Accessed on 10/02/2019).

- [283] Y. Tang, P. Lin, and Z. Luo, “psobj: Defending against traffic analysis with pseudo-objects,” in *International Conference on Network and System Security*, pp. 96–109, Springer, 2015.
- [284] Y. Tang, P. Lin, and Z. Luo, “Obfuscating encrypted web traffic with combined objects,” in *International Conference on Information Security Practice and Experience*, pp. 90–104, Springer, 2014.
- [285] Y. Li, R. Dai, and J. Zhang, “Morphing communications of cyber-physical systems towards moving-target defense,” in *2014 IEEE International Conference on Communications (ICC)*, pp. 592–598, IEEE, 2014.
- [286] W. B. Moore, H. Tan, M. Sherr, and M. A. Maloof, “Multi-class traffic morphing for encrypted voip communication,” in *International Conference on Financial Cryptography and Data Security*, pp. 65–85, Springer, 2015.
- [287] K. Kohls, T. Holz, D. Kolossa, and C. Pöpper, “Skypeline: Robust hidden data transmission for voip,” in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 877–888, ACM, 2016.
- [288] R. McPherson, A. Houmansadr, and V. Shmatikov, “Covertcast: Using live streaming to evade internet censorship,” *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 3, pp. 212–225, 2016.
- [289] D. Barradas, N. Santos, and L. Rodrigues, “Deltashaper: Enabling unobservable censorship-resistant tcp tunneling over videoconferencing streams,” *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 5–22, 2017.
- [290] F. Li, A. Razaghpanah, A. M. Kakhki, A. A. Niaki, D. Choffnes, P. Gill, and A. Mislove, “lib erate(n): A library for exposing (traffic-classification) rules and avoiding them efficiently,” in *Proceedings of the 2017 Internet Measurement Conference*, pp. 128–141, ACM, 2017.
- [291] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, “A systematic approach to developing and evaluating website fingerprinting defenses,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 227–238, ACM, 2014.
- [292] A. Iacovazzi and Y. Elovici, “Network flow watermarking: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 512–530, 2016.
- [293] L. Dixon, T. Ristenpart, and T. Shrimpton, “Network traffic obfuscation and automated internet censorship,” *IEEE Security & Privacy*, vol. 14, no. 6, pp. 43–53, 2016.

- [294] T. A. Ghaleb, “Techniques and countermeasures of website/wireless traffic analysis and fingerprinting,” *Cluster Computing*, vol. 19, no. 1, pp. 427–438, 2016.
- [295] C. V. Wright, S. E. Coull, and F. Monroe, “Traffic morphing: An efficient defense against statistical traffic analysis.,” in *NDS*, vol. 9, Citeseer, 2009.
- [296] C. H. Rowland, “Covert channels in the tcp/ip protocol suite,” *First Monday*, vol. 2, no. 5, 1997.
- [297] K. Ahsan and D. Kundur, “Practical data hiding in tcp/ip,” in *Proc. Workshop on Multimedia Security at ACM Multimedia*, vol. 2, 2002.
- [298] Y. Huang, B. Xiao, and H. Xiao, “Implementation of covert communication based on steganography,” in *2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 1512–1515, IEEE, 2008.
- [299] S. Burnett, N. Feamster, and S. Vempala, “Chipping away at censorship firewalls with user-generated content.,” in *USENIX Security Symposium*, pp. 463–468, Washington, DC, 2010.
- [300] L. Invernizzi, C. Kruegel, and G. Vigna, “Message in a bottle: Sailing past censorship,” in *Proceedings of the 29th Annual Computer Security Applications Conference*, pp. 39–48, ACM, 2013.
- [301] D. Barradas, N. Santos, and L. Rodrigues, “Effective detection of multimedia protocol tunneling using machine learning,” in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pp. 169–185, 2018.
- [302] H. Liu, Z. Wang, and F. Miao, “Concurrent multipath traffic impersonating for enhancing communication privacy,” *International Journal of Communication Systems*, vol. 27, no. 11, pp. 2985–2996, 2014.
- [303] Y. Hu, X. Li, J. Liu, H. Ding, Y. Gong, and Y. Fang, “Mitigating traffic analysis attack in smartphones with edge network assistance,” in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.
- [304] Y. Gokcen, V. A. Foroushani, and A. N. Z. Heywood, “Can we identify nat behavior by analyzing traffic flows?,” in *2014 IEEE Security and Privacy Workshops*, pp. 132–139, IEEE, 2014.
- [305] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” tech. rep., Naval Research Lab Washington DC, 2004.

- [306] S. Saleh, J. Qadir, and M. U. Ilyas, “Shedding light on the dark corners of the internet: A survey of tor research,” *Journal of Network and Computer Applications*, vol. 114, pp. 1–28, 2018.
- [307] G. He, M. Yang, J. Luo, and X. Gu, “A novel application classification attack against tor,” *Concurrency and Computation: Practice and Experience*, vol. 27, no. 18, pp. 5640–5661, 2015.
- [308] E. Hodo, X. Bellekens, E. Iorkyase, A. Hamilton, C. Tachtatzis, and R. Atkinson, “Machine learning approach for detection of nontor traffic,” in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, p. 85, ACM, 2017.
- [309] M. AlSabah, K. Bauer, and I. Goldberg, “Enhancing tor’s performance using real-time traffic classification,” in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 73–84, ACM, 2012.
- [310] M. AlSabah and I. Goldberg, “Performance and security improvements for tor: A survey,” *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 32, 2016.
- [311] S. Matic, C. Troncoso, and J. Caballero, “Dissecting tor bridges: a security evaluation of their private and public infrastructures,” in *Network and Distributed Systems Security Symposium*, pp. 1–15, The Internet Society, 2017.
- [312] B. Qu, Z. Zhang, L. Guo, X. Zhu, L. Guo, and D. Meng, “An empirical study of morphing on network traffic classification,” in *7th International Conference on Communications and Networking in China*, pp. 227–232, IEEE, 2012.
- [313] B. Qu, Z. Zhang, X. Zhu, and D. Meng, “An empirical study of morphing on behavior-based network traffic classification,” *Security and Communication Networks*, vol. 8, no. 1, pp. 68–79, 2015.
- [314] X. Fu, B. Graham, R. Bettati, and W. Zhao, “On effectiveness of link padding for statistical traffic analysis attacks,” in *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, pp. 340–347, IEEE, 2003.
- [315] A. Iacovazzi and A. Baiocchi, “Protecting traffic privacy for massive aggregated traffic,” *Computer Networks*, vol. 77, pp. 1–17, 2015.
- [316] A. Iacovazzi and A. Baiocchi, “Padding and fragmentation for masking packet length statistics,” in *International Workshop on Traffic Monitoring and Analysis*, pp. 85–88, Springer, 2012.

- [317] A. Iacovazzi and A. Baiocchi, “Investigating the trade-off between overhead and delay for full packet traffic privacy,” in *2013 IEEE International Conference on Communications Workshops (ICC)*, pp. 1345–1350, IEEE, 2013.
- [318] A. Iacovazzi and A. Baiocchi, “Optimum packet length masking,” in *2010 22nd International Teletraffic Congress (ITC 22)*, pp. 1–8, IEEE, 2010.
- [319] A. Iacovazzi and A. Baiocchi, “Internet traffic privacy enhancement with masking: Optimization and tradeoffs,” *IEEE transactions on parallel and distributed systems*, vol. 25, no. 2, pp. 353–362, 2013.
- [320] L. Wang, K. P. Dyer, A. Akella, T. Ristenpart, and T. Shrimpton, “Seeing through network-protocol obfuscation,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 57–69, ACM, 2015.
- [321] A. Houmansadr, C. Brubaker, and V. Shmatikov, “The parrot is dead: Observing unobservable network communications,” in *2013 IEEE Symposium on Security and Privacy*, pp. 65–79, IEEE, 2013.
- [322] M. Yang, J. Luo, Z. Ling, X. Fu, and W. Yu, “De-anonymizing and countermeasures in anonymous communication networks,” *IEEE Communications Magazine*, vol. 53, no. 4, pp. 60–66, 2015.
- [323] M. Rigaki and S. Garcia, “Bringing a gun to a knife-fight: Adapting malware communication to avoid detection,” in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 70–75, IEEE, 2018.
- [324] Q. Bai, G. Xiong, and Y. Zhao, “Find behaviors of network evasion and protocol obfuscation using traffic measurement,” in *International Conference on Trustworthy Computing and Services*, pp. 342–349, Springer, 2014.
- [325] Z. Cao, G. Xiong, and L. Guo, “Mimichunter: A general passive network protocol mimicry detection framework,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, pp. 271–278, IEEE, 2015.
- [326] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, “Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting,” *Computer Networks*, vol. 53, no. 1, pp. 81–97, 2009.
- [327] R. W. Shirey, “Internet security glossary, version 2,” 2007.
- [328] W. Stallings, *Cryptography and Network Security, 4/E*. Pearson Education India, 2006.

- [329] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, “Statistical identification of encrypted web browsing traffic,” in *Proceedings 2002 IEEE Symposium on Security and Privacy*, pp. 19–30, IEEE, 2002.
- [330] S. Chen, R. Wang, X. Wang, and K. Zhang, “Side-channel leaks in web applications: A reality today, a challenge tomorrow,” in *2010 IEEE Symposium on Security and Privacy*, pp. 191–206, IEEE, 2010.
- [331] M. Gruteser and D. Grunwald, “Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis,” *Mobile Networks and Applications*, vol. 10, no. 3, pp. 315–325, 2005.
- [332] T. Jiang, H. J. Wang, and Y.-C. Hu, “Preserving location privacy in wireless lans,” in *Proceedings of the 5th international conference on Mobile systems, applications and services*, pp. 246–257, ACM, 2007.
- [333] Y. Fan, B. Lin, Y. Jiang, and X. Shen, “An efficient privacy-preserving scheme for wireless link layer security,” in *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pp. 1–5, IEEE, 2008.
- [334] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall, “Improving wireless privacy with an identifier-free link layer protocol,” in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pp. 40–53, ACM, 2008.
- [335] K. Bauer, D. McCoy, B. Greenstein, D. Grunwald, and D. Sicker, “Physical layer attacks on unlinkability in wireless lans,” in *International Symposium on Privacy Enhancing Technologies Symposium*, pp. 108–127, Springer, 2009.
- [336] W. Hu, D. Willkomm, M. Abusubaih, J. Gross, G. Vlantis, M. Gerla, and A. Wolisz, “Dynamic frequency hopping communities for efficient ieee 802.22 operation,” *IEEE Communications Magazine*, vol. 45, no. 5, pp. 80–87, 2007.
- [337] A. Sheth, S. Seshan, and D. Wetherall, “Geo-fencing: Confining wi-fi coverage to physical boundaries,” in *International Conference on Pervasive Computing*, pp. 274–290, Springer, 2009.
- [338] I. Martinovic, P. Pichota, and J. B. Schmitt, “Jamming for good: a fresh approach to authentic communication in wsns,” in *Proceedings of the second ACM conference on Wireless network security*, pp. 161–168, ACM, 2009.
- [339] S. Lakshmanan, C.-L. Tsao, R. Sivakumar, and K. Sundaresan, “Securing wireless data networks against eavesdropping using smart antennas,” in

2008 The 28th International Conference on Distributed Computing Systems, pp. 19–27, IEEE, 2008.

- [340] F. Zhang, W. He, and X. Liu, “Defending against traffic analysis in wireless networks through traffic reshaping,” in *2011 31st International Conference on Distributed Computing Systems*, pp. 593–602, IEEE, 2011.
- [341] N. Al Khater and R. E. Overill, “Network traffic classification techniques and challenges,” in *2015 Tenth International Conference on Digital Information Management (ICDIM)*, pp. 43–48, IEEE, 2015.
- [342] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we need hundreds of classifiers to solve real world classification problems?,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133–3181, 2014.
- [343] M. Crotti, F. Gringoli, and L. Salgarelli, “Impact of asymmetric routing on statistical traffic classification,” in *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, pp. 1–8, IEEE, 2009.
- [344] M. Grzenda, “Towards the reduction of data used for the classification of network flows,” in *International Conference on Hybrid Artificial Intelligence Systems*, pp. 68–77, Springer, 2012.
- [345] W. De Donato, A. Pescapé, and A. Dainotti, “Traffic identification engine: an open platform for traffic classification,” *IEEE Network*, vol. 28, no. 2, pp. 56–64, 2014.
- [346] “Intelligent networking is all about app development - cisco.” <https://www.cisco.com/c/en/us/solutions/enterprise-networks/intelligent-network.html>. (Accessed on 10/02/2019).
- [347] “Huawei leaps into ai; announces powerful chips and ml framework.” <https://medium.com/syncedreview/huawei-leaps-into-ai-announces-powerful-chips-and-ml-framework-f9aa6ec87bcb>. (Accessed on 10/02/2019).
- [348] “Machine learning and endpoint security - palo alto networks.” <https://www.paloaltonetworks.com/resources/whitepapers/machine-learning-endpoint-security>. (Accessed on 10/02/2019).
- [349] “Artificial intelligence for smarter cybersecurity — ibm.” <https://www.ibm.com/security/artificial-intelligence>. (Accessed on 10/02/2019).

- [350] L. Grimaudo, M. Mellia, and E. Baralis, “Hierarchical learning for fine grained internet traffic classification,” in *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 463–468, IEEE, 2012.
- [351] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, “Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection,” *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
- [352] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, “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, July 2017.
- [353] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi, “A multi-level internet traffic classifier using deep learning,” in *2018 9th International Conference on the Network of the Future (NOF)*, pp. 68–75, Nov 2018.
- [354] “Vpn-nonvpn dataset (iscxvpn2016), howpublished = <https://www.unb.ca/cic/datasets/vpn.html>.”
- [355] “Tor-nontor dataset (iscxtor2016).” <https://www.unb.ca/cic/datasets/tor.html>.
- [356] “Tflearn — tensorflow deep learning library.” <http://tflearn.org/>.
- [357] “Tensorflow.” <https://www.tensorflow.org/>.
- [358] “scikit-learn: machine learning in python scikit-learn 0.21.3 documentation.” <https://scikit-learn.org/stable/>. (Accessed on 09/24/2019).
- [359] Buyun Qu, Zhibin Zhang, Le Guo, Xingquan Zhu, Li Guo, and Dan Meng, “An empirical study of morphing on network traffic classification,” in *7th International Conference on Communications and Networking in China*, pp. 227–232, Aug 2012.
- [360] B. Qu, Z. Zhang, X. Zhu, and D. Meng, “An empirical study of morphing on behavior-based network traffic classification,” *Security and Communication Networks*, vol. 8, no. 1, pp. 68–79, 2015.
- [361] L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, “Mobile traffic anonymization through probabilistic distribution,” in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pp. 242–248, Feb 2019.

- [362] Xinwen Fu, B. Graham, R. Bettati, and Wei Zhao, “On effectiveness of link padding for statistical traffic analysis attacks,” in *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, pp. 340–347, May 2003.
- [363] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail,” in *Proceedings of the 2012 IEEE Symposium on Security and Privacy, SP '12*, (Washington, DC, USA), pp. 332–346, IEEE Computer Society, 2012.
- [364] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, “Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset,” *Future Generation Computer Systems*, vol. 100, pp. 779 – 796, 2019.
- [365] “dpkt, howpublished = <https://dpkt.readthedocs.io/en/latest/>, note = Accessed: 2019.”
- [366] “tensorflow, howpublished = <https://www.tensorflow.org/>, note = Accessed: 2019.”
- [367] M. Huh, A. Liu, A. Owens, and A. A. Efros, “Fighting fake news: Image splice detection via learned self-consistency,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 101–117, 2018.
- [368] “Tensorflow.” <https://www.tensorflow.org/>. (Accessed on 09/24/2019).
- [369] L. Van Der Maaten, “Accelerating t-sne using tree-based algorithms,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.
- [370] “Github - jakobovski/free-spoken-digit-dataset: A free audio dataset of spoken digits. think mnist for audio..” <https://github.com/Jakobovski/free-spoken-digit-dataset>. (Accessed on 01/18/2020).