Encrypted Application Classification with Convolutional Neural Network

Kun Yang
High Speed Network Lab
New York University
New York, USA
ky13@nyu.edu

Lu Xu
High Speed Network Lab
New York University
New York, USA
lx643@nyu.edu

Yang Xu
School of Computer Science
Fudan University
Shanghai, China
xuy@fudan.edu.cn

Jonathan Chao
High Speed Network Lab
New York University
New York, USA
chao@nyu.edu

***Abstract*—Encrypted application classification (EAC) has become an emerging and challenging task for network monitoring and management, and statistical-based approaches are less impacted by encrypted streams. However, much effort is required from domain experts to handcraft statistical features. To solve this problem, this paper proposes an end-to-end encrypted application classification framework (E2E-EACF) based on one dimensional convolutional neural network (1D-CNN). Only encrypted payload (EncP) and inter-arrival time (IAT) are required by the framework to classify encrypted flows. Experimental results demonstrate that E2E-EACF can achieve more than 91.00% accuracy and 0.92 F1 score (the harmonic average of precision and recall) on a public dataset (WRCCDC), better than classical machine learning algorithms (e.g., decision tree and support vector machine).**

***Index Terms*—Machine Learning (ML), Encrypted Application Classification (EAC), Deep Learning (DL), Deep Packet Inspection (DPI), Convolutional Neural Network (CNN).**

I. Introduction

Encrypted application classification (EAC) has become an emerging and challenging task for network monitoring and management [1]. Much effort has been poured to address this issue; however, it is still far from being completely solved. The reasons can be mainly attributed to the follows: 1) New applications with unregistered or dynamic ports and new techniques (e.g., encryption and obfuscation) are continuously appearing [2], [3], resulting in traditional port-based classification approaches ineffective. 2) Deep packet inspection (DPI) based approaches [4] identify and classify traffic by comparing packets' payload with predefined rules. Consequently, they are very suitable for unencrypted traffic [5]. 3) Statistical-based approaches trying to find some statistical features, such as the maximum, minimum, and standard deviation of packets' length of each flow, which are less impacted by encryption techniques, seem to be viable. However, they require much effort and knowledge from domain experts to discover and summarize the features [6].

Motivated by the above reasons, this paper proposes an end-to-end encrypted application classification framework (E2E-EACF) based on one dimensional convolutional neural network (1D-CNN) to classify encrypted traffic into the category generated by the corresponding application as quickly as possible. The key contributions of this paper are summarized as follows:

1) E2E-EACF is focused on encrypted traffic analysis. All the experiments conducted in this paper exclude unencrypted traffic, e.g., UDP traffic (unencrypted), and DNS traffic, because they can be spoofed and is unreliable.

2) E2E-EACF can analyze raw packet streamings directly. Only encrypted payload (EncP, the spatial feature) and inter-arrival time (IAT, the temporal feature) are used to classify encrypted traffic without the need to handcraft any other features by human experts.

3) Experimental results show that E2E-EACF achieves more than 91.00% accuracy and 0.91 F1 on eight different kinds of encrypted traffic, which is better than classical machine learning algorithms. Moreover, the results show that combining the temporal feature with the spatial feature can further improve the performance. Any of them alone cannot achieve satisfying classification performance.

The remainder of this paper is organized as follows: Section II reviews the state-of-the-art works in this field. Section III explains the proposed E2E-EACF in detail. Section IV conducts experiments to evaluate E2E-EACF and analyze the results. Lastly, Section V concludes our work.

II. Related Works

This section focuses on the state-of-art works on EAC in recent years, and most of them are based on machine learning algorithms. Rezaei et al. [7] introduce a general framework for deep learning-based traffic classification, and present common deep learning methods and their applications in traffic classification tasks. The authors also discuss open problems and challenging issues in this field. Wang et al. [8] put forward an end-to-end approach based on CNN to classify traffic. The proposed approach is flow-based and uses the first 784 bytes of each flow as input features, which inspires our works. Lotfollahi et al. [9] propose a packet-based encrypted classification approach based on CNN and Stack Auto Encoder (SAE). They extract the first 1500 bytes of only one packet and take them as the inputs to train their classification models. However, they use some parts of header information and ignore temporal information hidden in a flow. Wang et al. [10] use three common deep learning algorithms: Multilayer

Annex to ISBN 978-3-903176-28-7

Perceptron, Stacked Autoencoder, and Convolutional Neural Networks, to classify encrypted traffic generated by different smart home devices under Software Defined Network (SDN). The authors want to better manage distributed smart home networks. Aceto et al. [11] propose to use common deep learning to design practical mobile traffic classifiers based on automatically-extracted features, which are able to cope with encrypted traffic. They evaluate the performance of the state-of-the-art deep learning classifiers based on an exhaustive experimental validation.

Even though many algorithms are proposed to classify encrypted traffic, they more or less use a part of header information as features. From the perspective of payload based classification, [8], [9], and [10] are the most similar works to our work. However, the main differences between our work and their works are: 1) Our work only uses encrypted application data to classify encrypted traffic without using any packet header information, and 2) Our work combines the spatial feature and the temporal feature to classify encrypted traffic without decrypting encrypted packet payload, and achieves higher accuracy than that of any of them alone (more details can be seen in subsection IV-C).

III. METHODOLOGY

Our classification is on per flow basis and our aim is to identify and classify encrypted flows as quickly as possible. Fig. 1 presents the proposed framework (E2E-EACF) in detail. The core of it is a classification module (CM) that can be simply recognized as a black box, which takes in raw packets and outputs the identified category of each flow. Inside the CM, there are two major components, online analysis component (OAC) and offline learning and updating component (OLUC). OAC is responsible for online analysis to cater to the real-time requirement, while OLUC aims to keep the online classification model up to date by regularly updating it with a new model, which constantly learns on the latest encrypted traffic.

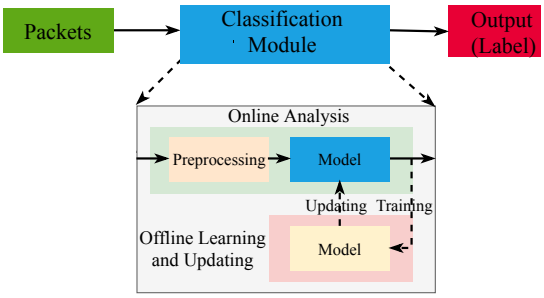

Fig. 1: End-to-end encrypted application classification framework (E2E-EACF).

A. Preprocessing

1) Traffic representation: In this paper, we consider combining a spatial feature (encrypted payload: EncP) and a temporal feature (IAT) to classify encrypted traffic. The main reasons are as follows: a) Encrypted payload, even after encryption process, still contains some useful information that is helpful to classify encrypted traffic. This assumption is validated in a latter section IV-C. "EncP" in this paper does not include any packets' header, including Ethernet header, IP header, TCP header, and even TLS header. IAT is another feature we use because this property is basically not affected by encryption [12]. On the other hand, the timing of packets cannot be easily modified because it has a direct impact on the performance of web services [12]. b) Any other information, such as header information, handshake packets, and flow-based features, is not used because this information can be spoofed and is unreliable. For example, IPs can be easily revised and spoofed by many public tools, e.g., ipspoof [13]. With a tool named Scapy [14], almost all content in the packet header can be modified.

2) Feature extraction: In general, feature extraction procedure mainly includes four steps, payload extraction (PE), IAT calculation (IAT-C), truncating/padding process (TPP), and normalization process (NP). PE is responsible for parsing packets, only extracting encrypted payload data (EncP), removing all header information (H) from each packet, and transforming EncP byte by byte from the hexadecimal values into the decimal values because our approach requires real numbers as an input. IAT-C is used to calculate the time difference between every two adjacent packets in each flow. The result is further transformed to decimal values in a fixed size (e.g., 4 bytes) and then appended to the extracted payload. Once the length of the extracted data (EncPs + IATs) exceeds the preset value len_{bytes} (e.g., 3,000), TPP truncates the extracted data to len_{bytes}. The corresponding flow of the extracted data is removed from the flow table and fed into the later parts. Otherwise, it keeps waiting until its length reaches the preset value or timeout. If the length reaches the preset value, truncation operation is executed. Otherwise, TPP pads zeros to the end of the extracted data until its length equals len_{bytes}. NP is applied to expedite the training process of the classification model. In this paper, Z-Score, one of the most commonly used normalization methods, is used to normalize the flow data.

3) Data labeling: This paper uses the value of server name indication (SNI) to label the samples popped from the flow table. The main reason is that the SNI value in the Client Hello packet of each TLS flow includes the domain name of the corresponding application.

B. Proposed classification approach

Once we obtain the samples represented by the features (EncP and IAT) mentioned in subsection III-A, 1D-CNN is used to identify and classify different encrypted traffic to the corresponding categories.

1) 1D-CNN: Fig. 2 shows the basic architecture and components of 1D-CNN. It includes input layer, convolutional layer (conv. layer), pooling layer, flatten layer, dense layer and output layer.

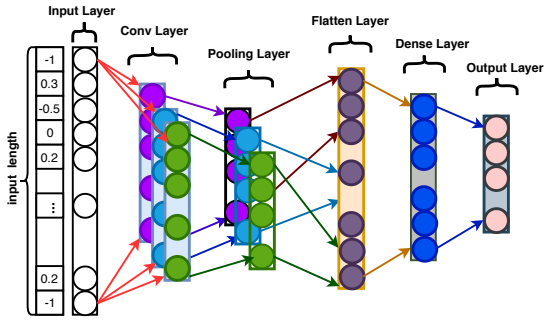

Fig. 2: 1D-CNN architecture.

Convolution operation (CO) in conv. layer is applied on the input data to extract abstract features (also known as feature maps). Let $x_i \in \mathbb{R}$ be the i_{th} sample in a dataset (X), which has n features (EncP+IATs), and $x_{i:s+j}$ represent the segment ($x_{i:s}$, $x_{i:s+1}$,..., $x_{i:s+j}$) of x_i with a length of $j+1$ features. In CO, a filter $w \in \mathbb{R}^k$ is involved and its size is $k * 1$. Therefore, in the 1D conv. layer, a feature z_s is produced from the segment of x_i ($x_{i:s+k-1}$) by

$$z_s = f(wx_{i:s+k-1} + b) \tag{1}$$

Here $b \in \mathbb{R}$ is the bias term and f is a non-linear activation function (e.g., leaky relu). This filter is applied to each possible window of segment of the input sample to generate a feature map

$$z = [z_1, z_2, ..., z_{n-k+1}] \tag{2}$$

with $\mathbf{z} \in \mathbb{R}^{n-k+1}$. Pooling layer is used to aggregate multiple low-level features and further reduce their number. Max pooling operation, $\max\{\mathbf{z}\}$, is applied on the feature map $\mathbf{z}$ to further expedite the learning and evaluation process of the classification model. After convolution and pooling operations, all the features in multi-channels (each CO generates one channel) are flattened to an one-dimensional vector that further forms flatten layer. Following the flatten layer is a dense layer that fully connects with the flatten layer, aiming to combine all the abstract features to assist the later classification process. The neural (c_i) in the dense layer can be produced by

$$c_i = f(w_i x_i + b_i) \tag{3}$$

where x_i is the input values from the previous layers, w_i and b_i are the weight and bias, respectively. Finally, all the results generated by the dense layer are passed to output layer that outputs the classification results by using softmax as activation function.

Once we obtain the classification model, then it can be used to identify and classify incoming flows to the corresponding category.

C. Offline learning and updating component

Keeping the classification model up to date is very important , and we use the following three steps to update the online classification model. Step 1: Using a local database (DB) to continually store the samples identified by the 1D-CNN model. Step 2: Using two conditions to determine if we need to update the online classification model. One is the number of samples n_{sams} stored in the DB and another is the interval $itvl$ between the previous update time and the current time. If one of them exceeds the preset threshold (e.g., $n_{sams} > 1,000$ or $itvl > 30$ minutes, which can be achieved from historical traffic), then OLUC is activated. All samples stored in the DB are used to train a cloned 1D-CNN Model, and its parameters (weights: W and biases: B) are copied from the current online model. Then these samples are removed from the DB. Step 3: Using the new model trained on the latest samples to replace and update the online model immediately.

IV. Evaluation

In this section, we evaluate the proposed approach and compare it with classical supervised machine learning approaches: decision tree (DT), support vector machine (SVM), k-nearest neighbors (KNN), logistic regression classifier (LR), and gaussian naive bayes (NB).

A. Dataset

In this paper, WRCCDC [15], a public dataset, is used to evaluate the proposed model. Firstly, the WRCCDC dataset filters all unencrypted traffic, such as UDP and DNS, and only keeps encrypted traffic. Then all the encrypted traffic is preprocessed with the procedures mentioned in subsection III-A. After the data preprocessing, we choose the top eight kinds of encrypted traffic, which comes from Google, Twitter, Youtube, Github, Facebook, Outlook, Slack, and Bing, respectively. The reason behind it is that these eight kinds of traffic account for more than 74.00% (31,280/42,179) flows and 75.00% bytes of all encrypted traffic, respectively, and each application has more than 700 samples (flows). Furthermore, to reduce the impact caused by data imbalance on classification algorithms [16], we randomly sample the flows (samples) in each category to keep each one around 700 samples.

B. Experiments

This section evaluates the proposed approach on the data obtained from subsection IV-A. The whole dataset is randomly split into three parts: Train set, Validation Set and Test Set, and the ratio is 8:1:1.

1) Parameter and architecture: In this paper, the architecture of our approach is similar to the general CNN architecture in Fig. 2. The major difference between them is that our architecture consists of four conv. layers, in which, the first two have 16 convolutional kernels, and the last two have 32. The size of each kernel is 3x1. A Max Pooling Layer with a pool size of 2x1 is added at the end of every two conv. layers. Moreover, we also use dropout after each pooling layer with a rate of 0.2. Then the output of the last dropout is flattened into a 1D vector. Finally, two dense layers are added in the model with 500 nodes and 300 nodes, respectively. In addition, Relu is chosen as the activation function; adam optimizer is used for optimization; cross-entropy is used as loss function; batch size is 64; learning rate is 0.0005, and epochs are 100. For DT,

SVM, KNN, LR and NB, all of them use default parameters in scikit-learn [17].

2) Results analysis: Based on the above architecture and parameters, we have obtained the following results. Fig. 3 shows the relation between input (sample) size (includes EncP and IAT) and classification accuracy of all models. The X axis is the input size for different models, and the Y axis is the corresponding classification accuracy. As shown in the figure, the larger input size, the higher accuracy, which is consistent with our expectation, i.e., a larger input size indicates the more information we used to train the model, and we hence can achieve higher accuracy than that of using a smaller size. For example, with 3000 bytes of the input size, the proposed model can achieve more than 91.00% accuracy, which is better than around 80.00% when 1000 bytes is used.

However, the accuracy does not increase further after the input size increases beyond 3000 bytes because the payload of some flows does not have more than 3000 bytes, which means that once the input size is more than 3000 bytes, zeros will be padded to the end of those flows, and it does not provide any more information for classification. Therefore, the accuracy does not increase anymore. Furthermore, for input size less than 1500 bytes, DT has the best result around 82.00% accuracy. The performance of the proposed model is a little bit worse than DT because insufficient information of each input sample is used to train the deep model, which causes the deep model not to have good generalization on Test Set. The performance of the other models (SVM, KNN, NB, and LR) are even worse than DT and the proposed model in all different input sizes.

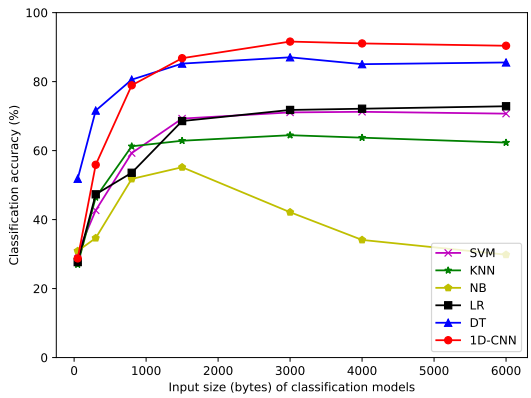

Fig. 3: Relation between input size and classification accuracy achieved by different approaches, including ours.

Considering time complexity and classification accuracy of the proposed approach, 3000 is a good input size. Under this condition, we have achieved the best accuracy, 91.61%, as seen in Fig. 3. What's more, only the first five packets are needed on average to accumulate 3000 bytes. However, this packet accumulation requirement does not mean that it will cause a significant delay. The proposed approach does not need to hold any packets while preparing for the input for the classification model. Instead, the network device can continue to forward packets upon their arrival, while storing and copying only the first few packets. Once the length of payloads and IATs obtained from a flow reaches the input size, classification is activated. According to the identified results, the network device can take a proper action to flow, e.g., changing its QoS priority or taking a security measure.

Fig. 4 shows the results of three metrics ($Precision$, $Recall$ and $F1$) achieved by our approach on Test Set. The X axis is for different categories, and the Y axis is the values of the metrics. We can see from the figure that our approach has a good performance on Google, Twitter, Outlook, Youtube and Facebook. For instance, we can achieve more than 94.00% $Precision$, 91.00% $Recall$ (except for Youtube, that is 88.14%) and 0.91 $F1$. The number of the flows generated by these five applications exceeds 90.06% of the total number of all eight encrypted flows. Moreover, only Google and Twitter account for more than 75.90% of the total number of all eight encrypted flows. Although the performance for the rest (Github, Slack, and Bing) is a little worse than that of the five applications, the total number of flows generated by these three applications is less than 10.00% of the total number of all eight encrypted flows. On the other hand, their performance in $Precision$, $Recall$ and $F1$ are still more than 82.00%, 88% and 0.85, respectively.

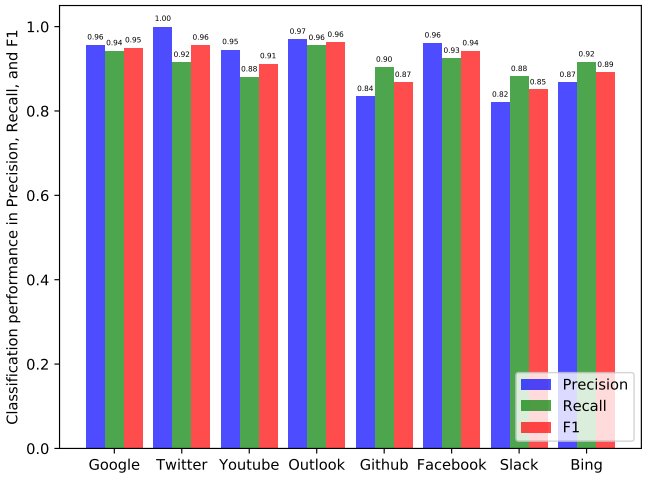

Fig. 4: The performance of Precision, Recall and F1 achieved by the proposed model on Test Set.

3) Analysis of the classification speed: The average time spent on identifying each flow in our approach is 291 μs. However, this time is much smaller than normal average flow duration. For instance, the work in [18] presents a workload characterization study of Outlook email traffic on campus (one of the applications used in our study). More than 70% of the average flow duration is more than 1 second. The work in [19] shows that about 90% of the average flow duration of Youtube (another application used in our study) is more than 15 seconds in the cellular network. Hence, we believe that the proposed model can be used to identify encrypted traffic.

C. Comparison with other works

This section talks about the main differences between our work and three related works in [8], [10] and [9], which are very similar to our work. We analyze and summarize the main differences in data object, features, handshake, and so on. Firstly, for data object, the majority of data samples used to build their models is unencrypted traffic, while all the data samples used to build our model is encrypted traffic. The promising result achieved by the approach in [8] is mainly due to the fact that a lot of packets generated by DNS and other protocols are included in the training data, and the distinct patterns the authors found in different applications are caused by those different protocols, e.g., DNS. Even though the authors in [9] filter DNS and mask IP addresses, they still use other header information (such as ports and TCP Flags). Hence the good results they obtained may come from this header information instead of payload. The work in [10] has the same problems as the work in [9]. On the other hand, our work does not use any header information, while all other works do (e.g., TLS header). Although the work in [9] claims that only payload is used as the features. However, the payload they referred is TCP payload that still includes TLS header.

Furthermore, none of them consider arrival time correlation among packets in each flow. In our scheme, we combine both the spatial feature (Payload: EncP) and the temporal feature (IAT) to classify encrypted traffic and achieve more than 91% accuracy when the input size is 3000 bytes. Fig. 5 shows the results of three experiments with different features, 1) Only using payload (EncP) as the feature, 2) Only using IAT as the feature, and 3) Using both of them as the features. There are two X axes in the figure; one is the input size, and the other is the number of IAT. The Y axis is the classification accuracy. From this figure, we can see that combining the spatial feature and the temporal feature can achieve better classification accuracy.

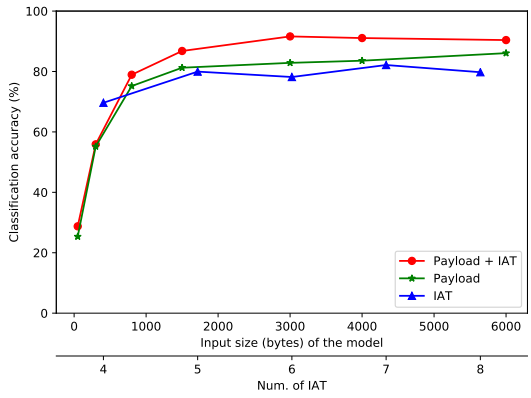

Fig. 5: Results achieved by the proposed model with different features on Test Set.

V. Conclusion and Future Works

This paper combines spatial feature (EncP) and temporal feature (IAT) to classify encrypted traffic based on 1D-CNN approach. Any other information, such as header information and handshake packets, is not used because this information can be easily spoofed and is unreliable. Extensive experimental results show that our approach has better performance than other approaches on the public dataset (WRCCDC). Our future work will further improve the accuracy of our approach and reduce its time complexity.

References

[1] Mirja Kühlewind, Brian Trammell, Tobias Bühler, Gorry Fairhurst, and Vijay Gurbani. Challenges in network management of encrypted traffic. *arXiv preprint arXiv:1810.09272*, 2018.

[2] Leon Böck, Emmanouil Vasilomanolakis, Max Mühlhäuser, and Shankar Karuppayah. Next generation p2p botnets: Monitoring under adverse conditions. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 511–531. Springer, 2018.

[3] IANA. IANA. https://www.iana.org/. Accessed: 2019-02-30.

[4] 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*, pages 1–27, 2018.

[5] Reham Taher El-Maghraby, Nada Mostafa Abd Elazim, and Ayman M Bahaa-Eldin. A survey on deep packet inspection. In *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, pages 188–197. IEEE, 2017.

[6] Blake Anderson and David McGrew. Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1723–1732. ACM, 2017.

[7] Shahbaz Rezaei and Xin Liu. Deep learning for encrypted traffic classification: An overview. *arXiv preprint arXiv:1810.07906*, 2018.

[8] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 43–48. IEEE, 2017.

[9] Mohammad Lotfollahi, Ramin Shirali Hossein Zade, Mahdi Jafari Siavoshani, and Mohammdsadegh Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *arXiv preprint arXiv:1709.02656*, 2017.

[10] Pan Wang, Feng Ye, Xuejiao Chen, and Yi Qian. Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. *IEEE Access*, 6:55380–55391, 2018.

[11] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management*, 2019.

[12] Saman Feghhi and Douglas J Leith. A web traffic analysis attack using only timing information. *IEEE Transactions on Information Forensics and Security*, 11(8):1747–1759, 2016.

[13] Sunny Behal and Krishan Kumar. Characterization and comparison of ddos attack tools and traffic generators: A review. *IJ Network Security*, 19(3):383–393, 2017.

[14] Scapy. Scapy. https://scapy.net/. Accessed: 2019-03-11.

[15] WRCCDC. WRCCDC. https://archive.wrccdc.org/pcaps/. Accessed: 2019-03-11.

[16] David Masko and Paulina Hensman. The impact of imbalanced training data for convolutional neural networks, 2015.

[17] Scikit-learn. Scikit-learn v0.20.3. https://scikit-learn.org/stable/tutorial/basic/tutorial.html. Accessed: 2019-03-11.

[18] Zhengping Zhang and Carey Williamson. A campus-level view of outlook email traffic. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*, pages 299–306. ACM, 2018.

[19] Pierdomenico Fiadino, Pedro Casas, Alessandro DAlconzo, Mirko Schiavone, and Arian Baer. Grasping popular applications in cellular networks with big data analytics platforms. *IEEE Transactions on Network and Service Management*, 13(3):681–695, 2016.