Classification of Encrypted Internet Traffic Using Kullback Leibler Divergence and Euclidean Distance

Vanice Canuto Cunha, Arturo Zavala, Pedro Inácio, Damien Magoni, Mário M. Freire

▶ **To cite this version:**

Vanice Canuto Cunha, Arturo Zavala, Pedro Inácio, Damien Magoni, Mário M. Freire. Classification of Encrypted Internet Traffic Using Kullback Leibler Divergence and Euclidean Distance. 34th International Conference on Advanced Information Networking and Applications, Apr 2020, Caserta, Italy. hal-02493390

HAL Id: hal-02493390
https://hal.science/hal-02493390

Submitted on 27 Feb 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Classification of Encrypted Internet Traffic Using Kullback Leibler Divergence and Euclidean Distance

Vanice Canuto Cunha [1,2], Arturo A. Z. Zavala [1], Pedro R. M. Inácio[2], Damien Magoni[3], and Mario M. Freire[2]

[1]Universidade Federal de Mato Grosso, Cuiabá, Brasil
[2]Instituto de Telecomunicações, Universidade da Beira Interior, Covilhã, Portugal
[3]LaBRI-CNRS, Université de Bordeaux, Talence, France

Abstract. The limitations of traditional classification methods based on port number and payload inspection to classify encrypted or obfuscated Internet traffic, often with randomized port numbers, have lead to significant research efforts focusing on classification approaches based on Machine Learning techniques using Transport Layer statistical features. However, these approaches also have their own limitations, leading to the study of a set of other alternative approaches, including statistics-based approaches. Statistical approaches can be an alternative to machine learning, because in real-time traffic classification with new types of data, the entire traffic classifier has to be retrained in order to adapt to the new change by combining the old training data with the new training data. This article investigates the classification of encrypted traffic using statistical methods applied to network traffic classification. We propose two statistical classifiers for encrypted Internet traffic based on Kullback Leibler divergence and Euclidean distance, which are computed using the flow and packet size obtained from some of the protocols used by applications. In our experiments, we evaluate the two classifiers based on statistical methods and compare them with a classifier based on Support Vector Machine (SVM). During our study, we were able to classify the traffic by using few features without compromising the performance of the classifier. The experimental results illustrate the effectiveness of our models used for traffic classification.

Keywords: traffic classification, encrypted Internet traffic, Kullback-Leibler divergence, Euclidean distance

1 Introduction

Internet traffic classification has been the focus of significant research efforts in the past two decades due to its importance for network management and security defense, since it may provide valuable information about the traffic metadata and the eventual underlying motivations [1], [2]. The study of statistical methods to classify network traffic is justified by the fact that many machine learning

techniques are supervised and then limited to the real time approach, as they require a new training model be created to a new classification each time new data is presented [1].

One of the major problems in classifying encrypted traffic is that the payload is encrypted, which makes difficult the analysis of the packet contents. Through statistical methods we can estimate, by means of empirical distributions, the behavior of the protocols, and with the help of divergences that show the similarities between two distributions, we can try to efficiently classify the applications that generate the Internet traffic under evaluation. This article proposes and evaluates the performance of two classifiers for encrypted Internet traffic using statistical methods applied to traffic flows: the Kullback Leibler (KL) divergence and the Euclidian distance. These two classifiers operate at a network flow level and make use of the relative frequency of the packet size to identify applications. KL divergence has previously been used for speaker identification/verification and image classification [18] and for detection of low-rate Distributed Denial of Service (DDoS) attacks [27]. Euclidean distance has previously been used for optimizing an artificial immune system algorithm used for flow-based Internet traffic classification [16]. Here, both KL divergence and Euclidean distance are used to build classifiers without the need to combine them with other methods.

2 Related Work

Significant research efforts have been carried out on the subjects of identification and classification of Internet traffic. Part of them rely on statistics, see, e.g, [5, 7]. Recent research still focuses on making improvements using machine learning, such as [1]. In [2], a classification module focusing on video streaming traffic, based on machine learning, is presented as a solution for networks that require real-time traffic analysis. In our work, we use real-world traces of encrypted traffic as well as Peer-to-Peer (P2P) applications such as edonkey, bittorrent, and gnutella. Some research works propose classifiers based on learning methods and corresponding signatures[13, 14]. A large number of research works, regardless of the classification method, make use of traffic flows or packet sizes (e.g, [5], [12].

Statistical tests such as Chi-Squared and Kolmogorov-Smirnov were used for traffic classification in [7], representing the classes through the corresponding signatures and the empirical distributions. The entropy was also used to measure and represent important differences regarding packet heterogeneity [8]. Exploring the heterogeneity of the packet sizes was accomplished by using samples obtained from a sliding window. For our work, we explored the characteristics of packet size, but using all the relative frequency flows and not just samples of the flow. Still in [8], Gomes et al proposed an online classifier to separate traffic generated by P2P and non-P2P applications and a new method to identify VoIP sessions [5].

Peng et. al propose in [12] a statistical classification approach that uses the Message Size Distribution (MSDC), which aims to identify the network flows precisely and the Message Size Sequence (MSSC) in real time. Such technique

provided very good detection results, making a decision after inspecting less than 300 packets with a 99.98% precision, but do not display recall and F-Measure values. In [15], Extreme Learning Machine (ELM) methods were used to classify Internet traffic. The Extreme Kernel learning machine (EKLM) approach, an ELM approach, was applied to the data. In particular, a Genetic Algorithm (GA)-based software was implemented for selecting the parameters used in the Extreme Kernel learning machine with Wavelet (WK-ELM) algorithm, where the Wavelet function was used. A precision rate over 95% was reached.

3 Statistical Methods

3.1 Kullback-Leibler Divergence

The KL divergence or relative entropy is a distance measured between two probability distributions and was introduced by the mathematicians S. Kullback and R.A. Leibler in 1951 [9], [10], [11], [26]. These researchers started with the assumption that two probability distributions differ more or less according to the possibility of discrimination between them by means of a statistical test.

The KL divergence is a special case of a wider class of divergences. By using this method, we can infer a similar behavior, or divergence between two distributions [26]. Considering that $D_{kl}[p||q]$ is a function, p_i and q_i are two probability distributions, we have:

$$D_{kl}[p||q] \quad = \quad \sum pi \log\left(\frac{1}{pi}\right) - \sum pi \log\left(\frac{1}{qi}\right). \tag{1}$$

Then, it can be assumed that the Kullback-Leibler divergence is represented by equation 2:

$$D_{kL}[p||q] \quad = \quad \sum_{i=1}^{N} pi \log\left(\frac{qi}{pi}\right), \tag{2}$$

where $D_K L[p||q] >= 0$ and $D_K L[p||q] = 0$ if and only if $p_i(x) = q_i(x)$, N defines the number of samples, i defines the number of the initial sample, pi defines the relative frequencies of the known class, qi defines the class relative frequency to be compared.

Note that, despite being also known as a divergence, it cannot be considered as a distance metric, since it does not meet the symmetry property, i.e., $D_K L[p||q] \neq D_K L[q||p]$. We mapped the behavior of some protocols through distributions (relative frequency) and used this mapping to classify traffic, assuming that each known distribution is pi and each unknown distribution is qi. KL divergence was used to implement one of the classifiers described in section IV, more specifically the Statistical Analysis module.

3.2 Euclidean Distance

To calculate the distance between two traffic classes, one must consider the probabilities of each traffic class, in our case, p_i defines the relative frequencies for

all the discrete values i (possible packet sizes) of the traffic class 1, representing the known traffic class, while q_i defines the relative frequencies for all the discrete values i (for possible packet sizes) of traffic class 2, representing the unknown traffic class. Therefore, the distance between both classes of traffic may be given by the following equation:

$$D_E[p,q] = \sqrt{\sum_{i=1}^{n} (p_i - q_k)^2}. \quad (3)$$

In this work, we use Euclidean distance to compare its results with the ones obtained with KL and to test the efficiency of this method, when mapping the behavior of the relative frequency for each protocol. The Euclidean distance is used to implement the second classifier described in section IV.

4 Traffic Classification Using Kullback-Leibler Divergence and Euclidean Distance

This section details our approach to classify Internet traffic. We describe the traffic features used for the classification, the classifier architecture incorporating the divergences and system modules, as well as the rules implemented for traffic classification.

4.1 Features

It is possible to create a new set of data from the original data, which contains important information obtained through the new features found [17]. This new set of data can present a smaller number of attributes than those found in the original set, bringing us as a benefit a lower dimensionality. In order to obtain this new set of data, we extract some features of the original database, such as packet sequence numbers, IP source and destination, packet size, time [19], where we apply a different view of the original data to reveal its important characteristics. We call the original base, the trace of traffic that has been stored after collection of network traffic. We consider each flow as a time series, which generates a standard characteristic: the relative frequency of packet size per application. We observe that the flow (source IP and destination IP) of the application would give us a feature that could be extracted, the relative frequency of packet size in each flow. We believe that the distribution of sizes is of great importance to characterize the traffic, since it represents each protocol in a unique way, forming a signature for each one of them. Figure 1 illustrates the relative frequency distribution of packets for some available applications in our new dataset. As we can see, each protocol behaves in a unique way and exhibits a unique distribution, which we call a signature. We can use such behavior to measure the divergences to known distributions and to start the classification. The X-axis of fig. 1 represents the number of buckets. Buckets were defined with

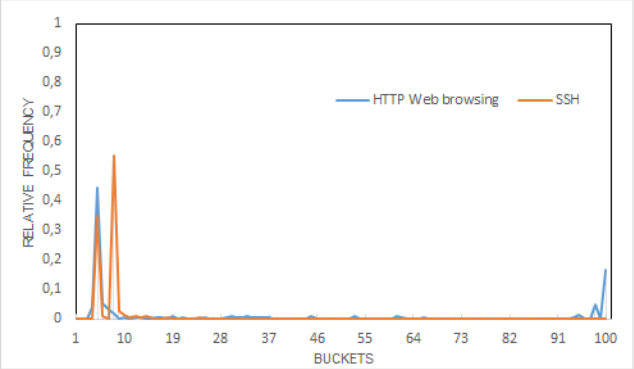


Fig. 1: Relative frequency of packet size per application, extracted through the sampling process, for SSH and HTTP Web Browsing.

the purpose of creating a histogram with intervals of 15 bytes. The maximum size of the histogram interval is 100. Buckets are required to calculate the relative frequencies of packets. The Y-axis represents the calculated relative frequency of HTTP Web browsing and SSH.

4.2 Classification Approaches

Divergence or distance is the measure of separation between two distributions, it indicates how similar or different two traces are. But for this we must insert some parameters so that, with that there is the comparison. The architecture for our classification system contains four modules, as shown in Fig. 2(a): packet capture and pre-processing, statistical analysis and stored signatures, classification and validation.

Packet Capture and Pre-Processing This step deals with the traffic capture and storage. Traffic generation was done in the laboratory in a controlled environment, and the capture was performed with the *tcpdump* and *windump* tools. The files were stored in the .pcap file format.

Then these file were converted to the .txt format. This conversion was done via command line with the command $tshark - rsrcfile.pcap - t$. After the files were separated into several files by application with the command $tcpdump - ettnnrfile.pcaphostIPorhostIP - wfile.pcap$.

First, we separated our dataset into individual trace files and collective trace files. We classify as individual trace files those in which we know by which applications they were generated. We separated into flow. We consider the application flow, source IP and destination IP, packet sequence and number of packets belonging to the same application and IP.

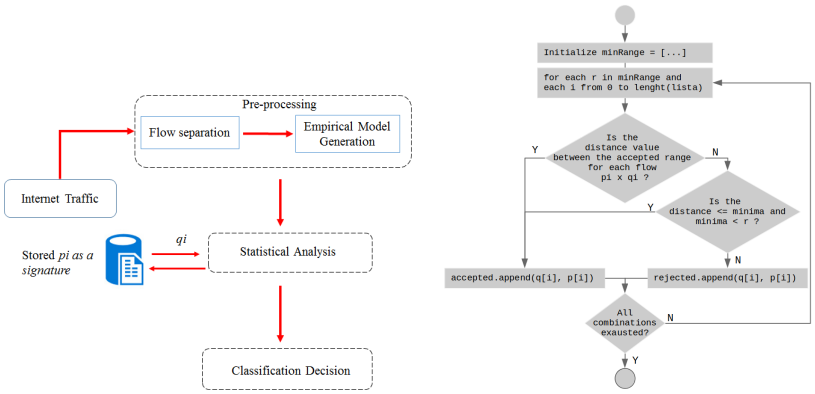


Fig. 2: Proposed classifier: (a) Architeture; (b) Flowchart of rules process.

Empirical model generation In this step, relative frequency files are generated that will feed the new base date. It was necessary to create file of relative frequencies corresponding to the known traces. These file were created by means of samples. To create the files with relative frequencies was developed a python script. For the generation of relative frequencies, values taken into account are the Maximum Transmission Unit (MTU), equivalent to 1514 bytes. This reference was crucial for the construction of the packet size histogram. Each byte packet was allocated in a sort of bucket. We take into account the size of 16 bytes to construct the histogram, so if we take the 1514 bytes and divide by 16 bytes, we will have an approximate value of 100 buckets, or intervals. Applying this logic to each flow, we are then able to calculate the empirical distributions - in our case, the relative frequencies.

Samples generation In this step, the program reads files from collective and individual folders and separate flows. The samples were generated from both individual and collective traces files. Each sample represents a relative frequency file generated for each protocol. The relative frequencies of the individual traces are stored in another dataset, as they will serve for signature identification purposes during the trace comparison phase. For flow grouping and separation, an application was built using the Python language, which allowed us to follow the entire flow and extract, with the construction of the histogram, the distributions of each flow. No feature was used to examine packages.

Statistical Analysis and Stored Signatures In this step the distance and divergence based on the relative frequencies of each flow are calculated. In order to know if two traffic traces belong to the same class, we use the empirical distribution, in our case the relative frequency that each trace has, and compare the two frequencies by applying the KL or Euclidean calculation, and obtaining

a distance as output. We begin the comparison phase between the distributions, where in this phase, the divergences are used, as shown in figure 2(a). In this step we consider the generated and separated files in the pre-processing module and rename these files in samples pi and qi. Now, pi for individual flow files and qi for collective flow files. Relative frequencies individual are stored as a signature and compared to Relative Frequencies collective using the distances of KL and Euclidean. For calculating distances, we choose which method we will apply, either Kullback-Leibler or Euclidean, and compare the relative frequency lists pi and qi generated. After this comparison, we have the values of the distances found between the two lists. This distance alone does not represent anything, therefore, in the classification step it is necessary to make use of heuristics. Note that, individual relative frequencies are stored as a signature and compared to collective relative frequencies using the KL and Euclidean distances.

Classification and Validation The classification of flows based on rules is made in this step. Note that, at this stage, we already have all the distances calculated between the flows. For divergences, the closer to 0 the most similar the two protocols are, the closer to 1, the most different the two protocols are likely to be. After the calculation, and having the values of the obtained distances, classification rules have to be created so that the classifier can make a decision. These rules are based on statistical heuristics after several iterations of distance calculation.

Initially, the rules for classification were based on dissimilarity values, where dissimilarity can be defined as follows: 0 if the attribute values match and 1 if they do not match. Since we will not always have distances equal to 0, or equal to 1, we need to find a cut-off threshold for the classification to be done more efficiently. For the cut-off, after debiting all distances, we created two lists to insert accepted and rejected flows according to established rules.

As we mentioned earlier, we need a threshold to know when the flow could be accepted or rejected. At first, we used a heuristic where we applied a range of intervals so that flow distances could be accepted if they were within the interval or rejected otherwise. After several tests, we found that in the rejected list there were flows that could have been accepted, but given the cut-off threshold, had been rejected. So we found that this methodology to accept or reject the flow would not work in our case.

Next, we defined another rule, this time making use of the average packet size found in each flow and also adding their standard deviation. Unfortunately, the values on the rejected list remained high. It is worth mentioning that when we checked the rejected lists, there were many flows that should have been accepted instead.

After several analysis, we chose to use all the distances after the calculation of all distances to calculate the distance between the calculated distances to insert in our list of accepted flows the distances were smaller, compared to the others.

We also decided to use the standard deviation (SD) of the distances calculated by the samples as the cut-off threshold, inserting in our distance list all flows that were in the interval $[average, SD]$. Using this technique, we were able to significantly reduce cases of false positives and false negatives, but it was not enough to deliver satisfactory results.

In order to create an efficient rule, we calculate the relative frequency of a given known protocol that is in our signature database, and compare it with all relative frequencies of unknown protocols. After this comparison, several distances will be generated. The first step is to select the minimum distance among the calculations to apply the classification rule.

The second step, after several and continuous tests with each protocol, was defining acceptance baseline values that would define five different minimum ranges. For each range value, the pre-calculated minimum distance of each protocol must necessarily be within 0 and the current range of the loop. Once the classification is over, the one yielding the best F-measure will be returned, as per auxiliary variables within the classification loop.

The MaxR variable was created to store the minimum range of the best result, while MAxMatrix variable was created to store its confusion matrix. The maxFmeasure variable, initially set as -1, will then store the best f-measure among the results. Classification rules are illustrated in the flowchart shown in Figure 2(b) .

5 Performance Evaluation

After the classification process of the samples, we checked and validated the results of the classification using the ground truth. For this, we created two new lists: in the first list we have the number of items classified as accepted and that were actually found in the mapping dictionary according to the IP source and IP destination corresponding to each protocol, and in the second list we have the number of items classified as rejected. Note that even though a distance is in the accepted list, the flow may not actually belong to the corresponding protocol, and even though it is in the rejected list, the flow might actually belong to the corresponding protocol.

5.1 Dataset and Ground Truth

For the study and analysis of the traffic, it was necessary to collect traffic traces and to build a database containing the traces to be analyzed. The dataset is composed by traces, generated by different Internet applications and services. Network traffic used in this analysis was captured next to the source where it was generated, obtained in a controlled environment, where only one application was running in a certain computer. All of the traces were collected by using the TCPDUMP and WINDUMP tools, implemented by the capture libraries: libpcap in the Linux platform, and Winpcap in the Windows platform. This way, we observed the properties both scenarios and established the ground truth

of the traffic records. In order to analyze the empirical distribution of the traffic packet size, an application was built to create the relative frequency distribution for each collected traffic. In this study, we chose services or applications that are widely used, heavy bandwidth consumers or raise more challenges from the perspective of traffic and network management, ending up with a set of services with varied characteristics. The classes considered for the traffic analysis were commonly used in the Internet, as listed below:

- Web browsing: browsing of general web pages, excluding media streaming.
- On-demand and live streaming: HTTP and Flash-based, RTSP, MMS, etc.
- P2P streaming: PPStream, TVUPlayer and SopCast.
- P2P file-sharing: BitTorrent, e-Donkey, and Gnutella.
- VoIP: Skype, Google Talk, Session Initiation Protocol (SIP) traffic.
- File transfer: FTP and SFTP transfers.
- Remote session: Telnet and SSH sessions.

Table 1: Dataset Characteristics.

Dataset	Volume (GB)	TCP (%)	UDP (%)
Dataset 1	8.80	78.35	21.59
Dataset 2	8.60	82.77	17.18
Dataset 3	7.97	77.13	22.84

In this work, "HTTP download" is used to refer to a long and continuous download of a large file using HTTP, while "Web navigation" is used to refer to the common activity of visiting Web pages through a Web Browser. The used dataset includes a total number of 92.317 traffic flows, corresponding to 35.317.091 packets and approximately 27.37 GB of traffic. The time to acquire all the datasets is 61h. Datasets were divided in three as described in table 1.

Ground truth is widely used by researchers who collect their own traffic traces to test the accuracy of their solutions. In our work it is possible to establish ground truth because traffic was collected in a controlled environment where each computer was running only one application. With this, we can map exactly the trace for a given application, calculate the relative frequency of the trace and compare it with the output of the classifier. The individual traces were used for the generation of our ground truth and the files with the collective traces were used for our tests.

5.2 Performance of the classifier

To address these issues, we used the confusion matrix, which allows us to obtain the performance and prediction of the classifier, based on the values of TP, TN, FP and FN. For the values of TP, the classifier added them to the list of accepted flows which, after the analysis and classification, are considered to belong to that flow. For FP values, we have the protocols erroneously classified as belonging

to the flow. For the values of TN, the classifier correctly understood that the protocol did not belong to the flow, inserting it in the accepted list. Finally, for FN, the classifier erroneously understood that the current protocol did not actually belong to the flow, inserting it in the rejected list. For performance evaluation, we use the classical performance metrics defined in machine learning textbooks: accuracy, precision, recall and F-measure.

Table 2 presents the accuracy, precision, recall and F-Measure for the Kullback-Leibler and Euclidean methods for all 8 classes tested. After evaluating the performance of the classification proposing the use of Kullback-Leibler divergence, we compared the performance with methods already found and tested in the literature[16], [22],[23]. For our comparison, we use of the Euclidian distance and Support Vector Machine (SVM). For testing with SVM we use the $sklearn-import-svm$ [24] function provided by the Python library and applied it to our dataset with the default parameters, except for the $Self$ parameter, which for our case we use C = 50. The largest problems encountered in setting up the SVM model were how to select the kernel function and its parameter values. When there are large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. We use four different kernels, which are linear, polynomial, sigmoid, and RBF kernels. For testing using SVM, we use the training module and change the Statistical Analysis KL module to Classification-SVM.

Table 2: Performance results with Kullback-Leibler divergence and Euclidean Distance.

Traffic Category\Protocol	Methods							
	Kullback-Leibler				Euclidean			
	Acc.	Rec.	Prec.	F-M.	Acc.	Rec.	Prec.	F-M.
Web browsing	99%	74%	100%	85%	99%	67%	100%	80%
HTTP download	99%	83%	100%	90%	99%	66%	100%	80%
Streaming On-demand and live	100%	100%	100%	100%	100%	100%	100%	100%
P2P streaming	99%	86%	100%	92%	99%	85%	100%	91%
P2P file-sharing	99%	76%	100%	86%	99%	73%	100%	84%
VoIP	100%	100%	100%	100%	100%	100%	100%	100%
File Transfer	100%	100%	100%	100%	100%	100%	100%	100%
Remote session	100%	100%	100%	100%	100%	100%	100%	100%

Legend: Acc: Accuracy, Rec: Recall, Prec:Precision, F-M: F-Measure

A summary of the results obtained from the classification using SVM is shown in table 3. The method using the SVM RBF kernel and the sigmoid kernel could not classify the dataset relative frequencies used in this article. Results achieved were too low or close to 0. For the category of Web browsing traffic, the SVM method has a classification capability, the Euclidean and KL methods presented very similar results, Kullback-Leibler having higher Recall and F- Measure values.

Table 3: Performance results with SVM Linear and Polynomial kernels.

Traffic Category\Protocol	Support Vector Machine - SVM							
	Linear				Polynomial			
	Acc.	Rec.	Prec.	F-M.	Acc.	Rec.	Prec.	F-M.
Web browsing	74%	74%	74%	74%	0%	0%	0%	0%
HTTP download	81%	82%	35%	4,9%	0%	25%	10%	15%
Streaming On-demand and live	57%	58%	57%	57%	0,90%	1%	9%	2%
P2P streaming	99%	99%	99%	99%	0,57%	10%	97%	10%
P2P file-sharing	98%	98%	97%	98%	0,0005%	0%	100%	0%
VoIP	100%	100%	100%	100%	0,22%	2%	100%	4%
File Transfer	0%	0%	0%	0%	0%	0%	0%	0%
Remote session	0%	0%	0%	0%	0%	0%	0%	0%

Legend: Acc: Accuracy, Rec: Recall, Prec:Precision, F-M: F-Measure

For the HTTP download category, the SVM method was unclear. Although the KL method had higher values than the SVM method, the values of Recall and F-Measure are still below acceptable values to state that for this type of application the method can classify this traffic category efficiently.

For on-demand and live streaming, the SVM method obtained results below the values found for the statistical methods. Analyzing the results, we realized that for this type of traffic, both KL and Euclidean methods, can classify efficiently and effectively. The results show that both methods are far superior than any of the SVM kernels.

For P2P streaming, linear, polynomial, and sigmoid SVM kernels have at least one metric considered reasonable for classification, but overall they are insufficient. This leads us to conclude that for video and video streaming traffic, the KL and Euclidean methods are efficient for identification and classification of these applications.

For P2P file-sharing, the KL and Euclidean methods gave excellent results for the precision and F-Measure metrics and good results for the Recall metric. The SVM linear kernel yielded results that are considered excellent.

For VoIP classification, the linear kernel SVM method presents significant results. KL and Euclidean methods gave excellent results for this type of traffic, reaching 100 % precision, Recall and F-measure results, which means that for this application the KL and Euclidean methods were very promising.

For the File Transfer traffic category the KL and Euclidean methods can also, given the obtained values, present an excellent classification capacity. For Remote session applications, the values achieved by the KL and Euclidean methods were 100% for the Accuracy because, although we have it mapped in our individual database, there are no corresponding files in the collective test database. The classifier states that all tested files do not contain Remote sessions, which in our analysis is correct, since we do not have any.

By making a comparative analysis between the statistical methods, we conclude that KL and Euclidean methods obtain the same results for most cases.

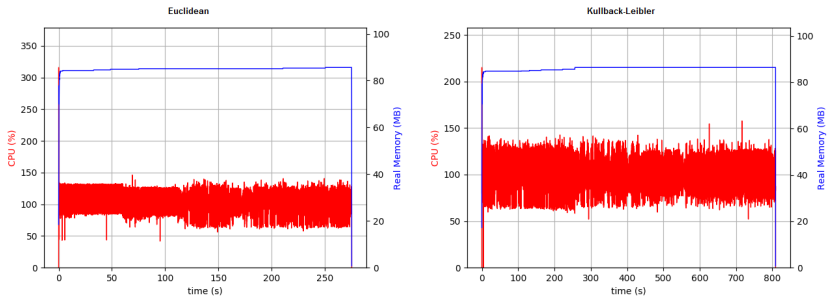
When comparing the F-Measure results obtained in both, we see that for the HTTP Download and Web Browsing traffic only, the values obtained by the KL method were higher, but we cannot say that this method is superior to the Euclidean one, given that for the other results, the values were the same or relatively similar. It is interesting to note that SVM is considered an excellent classifier in the literature, but when we do not have many features, we notice from the results that SVM does provide satisfying results. Therefore, building network traffic classifiers using statistical methods can still be considered a viable alternative for encrypted traffic.

5.3 Resource Usage

The experiments were executed on a 64-bit Linux desktop computer, equipped with an Intel (R) Core (TM) i7 CPU 2.93GHz, 6GiB system memory and a PCI Express Gigabit Ethernet Controller based on the RTL8111 chipset.

For computational analysis, the *psrecord* [25] tool is used. This tool records the core and memory activities of a process. In order to measure the computational performance of each method, we used the activity of the process that the method refers to. We did not use packet number or host/port, as this data was used only as information to convert the raw database into relative frequencies. When we look at fig. 3(a) and fig. 3(b), we find that memory usage remains stable whatever the processing consumption of the chosen classifier.

When the classifier is started, the CPU and memory time start at 0. If we observe the CPU usage over time, we see that KL and Euclidean classifiers both use a similar CPU percentage to process the information and compare the relative frequencies. Although not shown, the SVM classifier requires more CPU especially at the beginning of the process. Regarding the memory usage over time for each statistical classifier, they are very close, being slightly larger for the KL classifier, because KL is more demanded.



(a) (b)

Fig. 3: CPU and memory consumption from the beginning of the analysis of the trace to end of the classification (execution time) for the classifiers based on: (a) Euclidean distance; (b) Kullback-Leibler divergence.

6 Conclusion

Our results for Internet traffic classification were presented using Kullback-Leibler (KL) divergence, Euclidean distance and SVM. According to the results obtained, we can conclude that the SVM method with kernel set to default parameters is not effective to classify flows based only on packet size. SVM RBF was not efficient in any classification. Linear SVM was only efficient for classifying P2P streaming. The SVM sigmoid was efficient for classifying P2P streaming and VoIP. The SVM polynomial was efficient for classifying P2P streaming, P2P file sharing, and VoIP.

In contrast, the KL and Euclidian methods were able to classify all tested applications, standing out in the streaming and P2P classification, where for almost all cases it was efficient to identify them with a high precision.

We conclude that both the KL divergence and the Euclidean distance were efficient and that, in cases where the KL divergence was superior, it was not significantly better than the Euclidean distance. The performance of the statistical method is considered good as long as CPU usage remains almost constant when packets are being processed and analyzed. As for memory usage, it fluctuates according to the requests of each method.

Because these results are based on an approach that uses only packet sizes, it can be integrated with other methods to improve overall classification as it is a very promising classifier for high-speed network taps where traffic is encrypted. For future work, we intend to evaluate other statistical divergences indicators by performing new tests on encrypted traffic.

References

1. G. Sun, T. Chen, Y. Su, and C. Li, "Internet Traffic Classification Based on Incremental Support Vector Machines," Mob. Networks Appl., 23(4):789–796, 2018.
2. K. L. Dias, M. A. Pongelupe, W. M. Caminhas, and L. de Errico, "An innovative approach for real-time network traffic classification," Comput. Networks, 158:143–157, 2019.
3. N. Cascarano, L. Ciminiera, and F. Risso, "Optimizing deep packet inspection for high-speed traffic analysis," J. Netw. Syst. Manag., 19(1):7–31, 2011.
4. J. Zhang, Z. Li, Z. Pu, and C. Xu, "Comparing prediction performance for crash injury severity among various machine learning and statistical methods," IEEE Access, 6:60079–60087, 2018.
5. J. V. Gomes, P. R. M. Inácio, M. Pereira, M. M. Freire, and P. P. Monteiro, "Identification of peer-to-peer VoIP sessions using entropy and codec properties," IEEE Trans. Parallel Distrib. Syst.,24(10):2004–2014, 2013.
6. M. Zhang, W. John, K. C. Claffy, N. Brownlee, and U. C. S. Diego, "State of the Art in Traffic Classification : A Research Review," PAM '09 10th Int. Conf. Passiv. Act. Meas. Student Work., pp. 3–4, 2009.
7. M. Neto, J. V. Gomes, Mário M. Freire, and P. R. M. Inácio, Real-time traffic classification based on statistical tests for matching signatures with packet length distributions, 19th IEEE Workshop on in Local and Metropolitan Area Networks (LANMAN), 2013. IEEE, 2013, pp. 1–6.

8. V. P. Gomes et al., "Analysis of Peer-to-Peer traffic using a behavioural method based on entropy," Conf. Proc. IEEE Int. Performance, Comput. Commun. Conf., pp. 201–208, 2008.
9. T. M. Cover and J. A. Thomas, "Elements Of Information Theory", John Wiley and Sons, 2012.
10. C. Delpha, D. Diallo, and A. Youssef, "Kullback-Leibler Divergence for fault estimation and isolation: Application to Gamma distributed data," Mech. Syst. Signal Process., 93:118–135, 2017.
11. D. J. Galas, G. Dewey, J. Kunert-, N. A. Sakhanenko, B. Seattle, and H. Sciences, Expansion Of The Kullback-Leibler Divergence a New Class of Information Metrics, Axioms, 6(2):8, 2017.
12. L. Peng, H. Zhang, Y. Chen, and B. Yang, "Imbalanced Traffic Identification Using An Imbalanced Data Gravitation-Based Classification Model", Computer Communications, 102:177–189, 2017.
13. A. Tongaonkar, R. Torres, M. Iliofotou, R. Keralapura, and A. Nucci, "Towards self adaptive network traffic classification," Comput. Commun.,56:35–46, 2015.
14. D. Li, G. Hu, Y. Wang, and Z. Pan, "Network traffic classification via non-convex multi-task feature learning," Neurocomputing, 152:322–332, 2015.
15. F. Ertam and E. Avcı, "A new approach for internet traffic classification: GA-WK-ELM," Meas. J. Int. Meas. Confed.,95:135–142, 2017.
16. B. Schmidt, A. Al-fuqaha, A. Gupta, and D. Kountanis, "Optimizing an artificial immune system algorithm in support of flow-Based internet traffic classification," Appl. Soft Comput. J.,54:1–22, 2017.
17. H. Shi, H. Li, D. Zhang, C. Cheng, and W. Wu, "Efficient and robust feature extraction and selection for traffic classification," Comput. Networks, 119:1–16, 2017.
18. M., Pedro J and Ho, Purdy P and V., Nuno, A Kullback-Leibler Divergence Based Kernel For SVM Classification In Multimedia Applications, Advances in neural information processing systems, 1385-1392, 2004.
19. T, Pang-Ning; S, Michael; K, Vipin, Introduction To Data Mining, Person, Addison Wesley, Pearson Education, 2006. lSBN 0-321-420.
20. J. V. Gomes, P. R. M. Inácio, M. Pereira, M. M. Freire, and P. P. Monteiro, "Exploring behavioral patterns through entropy in multimedia peer-to-peer traffic," Comput. J., 55(6):740–755, 2012.
21. I. Syarif, A. Prugel-Bennett, and G. Wills, SVM parameter optimization using grid search and genetic algorithm to improve classification performance, Telkomnika (Telecommunication Comput. Electron. Control)., 14(4):1502–1509, 2016.
22. S. Tavara, Parallel computing of support vector machines: A survey, ACM Computing Surveys. 2019.
23. T. Marwala, Support Vector Machines. In Handbook of Machine Learning, Wold Scientific, pp. 97–112, 2018.
24. scikit-learn user guide Release 0.21.2. Available in https://scikit-learn.org/stable/modules/svm.html. Accessed in 06/05/2019.
25. Library PSrecord 1.1. Available in https://pypi.org/project/psrecord/ . Accessed in 06/05/2019.
26. S. Parveen, S. K. Singh, U. Singh, and D. Kumar, A Comparative Study of Traditional and Kullback-Leibler Divergence of Survival Functions Estimators for the Parameter of Lindley Distribution, Austrian J. Stat., 48(5):45–53, Jul. 2019.
27. Y. Xiang, K. Li, and W. Zhou, Low-rate DDoS attacks detection and traceback by using new information metrics, IEEE Transactions on Information Forensics and Security, 6(2):426–437, Jun. 2011.