Early Online Classification of Encrypted Traffic Streams using Multi-fractal Features

Erik Areström
Linköping University, Sweden

Niklas Carlsson
Linköping University, Sweden

***Abstract*—Timely and accurate flow classification is important for identifying flows with different service requirements, optimized network management, and for helping network operators simultaneously operate networks at higher utilization while providing end users good quality of experience (QoE). With most services starting to use end-to-end encryption (HTTPS and QUIC), traditional Deep Packet Inspection (DPI) and port-based approaches are no longer applicable. Furthermore, most flow-level-based approaches ignore the complex non-linear characteristics of internet traffic (e.g., self similarity). To address this challenge, in this paper, we present and evaluate a classification framework that combines multi-fractal feature extraction based on time series data (which captures these non-linear characteristics), principal component analysis (PCA) based feature selection, and man-in-the-middle (MITM) based flow labeling. Our detailed evaluation shows that the method is able to quickly and effectively classify traffic belonging to the six most popular traffic types (video streaming, web browsing, social networking, audio communication, text communication, and bulk download) and to distinguish between video-on-demand (VoD) and live streaming sessions delivered from the same services. Our results show that good accuracy can be achieved with only information about the timing of the packets within a flow.**

I. Introduction

In an increasingly competitive market, network operators must compete based on both the price and the quality of experience (QoE) that they offer their end users. Typically, network operators can reduce their costs per user (and hence also the prices that they can offer their customers) by operating some networks at higher bandwidth utilization. However, without careful flow scheduling and traffic-aware prioritization, this can easily result in reduced QoE. For optimized network management it is therefore important to have the ability to quickly and accurately classify flows based on the end-to-end services that they deliver.

Flow classification is a relatively well explored research topic and have, for example, been used by network providers to prioritize real-time streaming and interactive services at times when the more elastic demands of peer-to-peer networks have used up much of the bandwidth [1], [2], [3]. Deployments of these methods have traditionally used Deep Packet Inspection (DPI) [4], [5], in which the classification is done by analyzing the payloads of the packets. However, over the past few years, flow classification has been significantly complicated by the majority of flows today being delivered over end-to-end encrypted (HTTPS) connections [6], preventing access to payload information.

Recently there has therefore been an upswing in research trying to extract application level information from the encrypted traffic, including to determine the application class itself. Much of this work applies machine learning algorithms on high-level network and transport layer features such as port numbers, packet size statistics, and flow durations [4], [7], [8], [9], [10]. However, these types of summary statistics (easily obtained using Netflow [11] and similar tools) cannot capture the complex non-linear characteristics of internet traffic (e.g., self-similarity [12], [13], [14]) and may not allow classification until after the flow has completed (e.g., if using the flow duration as a feature). Shi et al. [15] therefore recently proposed to combine multi-fractal feature extraction based on time series data (which captures these non-linear characteristics) and principal component analysis (PCA) based feature selection methods for traffic classification. In this paper, we present enhanced variations of these methods and a man-in-the-middle (MITM) based evaluation framework that allows us to consider encrypted traffic (while they only considered unencrypted traffic), apply the methods on encrypted data from different application types (rather than protocols), and focus on early classification. Detailed evaluations and the impact of various model choices are presented for six of the most popular traffic types (video streaming, web browsing, social networking, audio communication, text communication, and bulk download) as well as to distinguish between video-on-demand (VoD) and live streaming sessions delivered from the same services (and IP addresses). The high accuracy for the second of these use cases are particularly encouraging as the three services considered (YouTube, Twitch, and SVT play) all use HTTP-based Adaptive Streaming (HAS) for both their VoD and live contents, and our technique only needs access to timing information of the packets within a flow.

The remainder of this paper is organized as follows. Section II describes how we collected and labeled the datasets used for training and evaluation. Section III describes our multi-fractal feature based model, how it was used to classify flows, and provides step-by-step results to help understand the classification process. Section IV presents evaluation results using the same dataset. Section V then uses another example use case (with complementing datasets) to demonstrate how the approach also can be used to distinguish between VoD and live streaming flows, even when delivered using the same HAS services. Finally, Sections VI and VII discuss related works and present our conclusions, respectively.

II. Collecting labeled time-series data

For training and evaluation, we need labeled time-series data for example flows[1] of each traffic category of interest. However, such labeling is significantly complicated when working with encrypted traffic. For this reason, we use a trusted proxy approach.

A. Trusted proxy approach

All the data collected needs to be available both in its encrypted and non-encrypted form so that a ground truth can be established. This is done by using a mitmproxy [16]. In particular, all traffic to and from the smartphone is setup to pass through the trusted proxy, which also decrypt and (re)encrypt the traffic before forwarding it to its destination. Packet captures are then split into short flows using the SplitCap [17] tool. Finally, for each flow, we create a per-millisecond time series of the packet arrivals during the first 20 seconds of the flow. In particular, by going through all packets in each flow, we create a 20,000 time-slot long time series array in which a time-slot during which a packet arrived (from the server) is one, and zero otherwise.

B. Data generation

Using the trusted-proxy approach to label real-world encrypted user data would require the users' consent. As this may not be feasible in practice, in this paper, data is instead generated by automatic instrumentation of mobile applications. More specifically, for each application of interest, we run a Python script (on a laptop) that forwards commands to the smartphone via the Android Debug Bridge (ADB), over the Universal Serial Bus (USB), so to control that application. Every script was uniquely created to capture example use cases of the traffic that each application may generate. However, to ensure fair comparison of how well different services can be classified early during the flow, each script generated a series of shorter 20 second connections followed by 10-second pauses (without traffic).

C. Example applications

Example applications to be evaluated were selected to represent six dominating traffic categories, each with different Quality of Experience (QoE) expectations and related Quality of Service (QoS) requirements. More specifically, we selected to consider the following six classes: *video streaming*, *web browsing*, *social networking*, *audio communication*, *text communication*, and *bulk download*. These six categories together represent over 80% of the total mobile access traffic in North America [18]. While *video streaming* is an obvious choice to include due to the prospect of available bandwidth savings and QoE optimizations recently presented for this category (e.g., [19]), other sub-categories are likely to see similar optimization frameworks developed. Of this reason we keep *web browsing* and *social networking* separate.

[1]Following standard convention, we define a flow as a sequence of packets sent between a source IP-port pair and a destination IP-port pair.

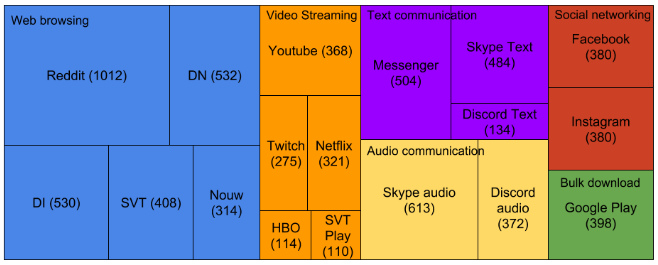


Fig. 1. Summary of the number of sessions per type.

Figure 1 summarizes the applications that were used for each category and the number of samples for each application (with the area of each rectangle proportional to the number of samples for that application). In total, we collected data for 7,154 sessions (or approx. 60 hours) of labeled data. We next describe how sample flows of each category were generated.

Video streaming: Videos from five different video services were used, including YouTube, Netflix, Twitch, HBO, and SVT play. Flows were generated by starting a randomly selected video, found using each service search function. To generate random samples, random keywords were selected from the 10,000 most commonly occurring words in the English language, as determined by frequency analysis of Google's Trillion Word Corpus [20].

Web browsing: Flows were generated by having the smartphone visit a sub-site of one of six different example websites: http://reddit.com (link collector), http://nouw.com (popular local blogging platform), http://svt.se/nyheter (popular local news), http://dn.se (popular local news), and http://di.se (popular local news). The visited sub-sites were chosen randomly from a list found by crawling the sites.

Social networking: We used Facebook and Instagram as examples. After entering the service, the script generated additional traffic by slowly and continuously "swiping" the screen vertically for the vertical length of three screens, causing new content to load throughout the full 20 second duration.

Audio communication: For this category, a phone call using one of two services (Discord audio and Skype audio) was made to a client that automatically accepted the call. For the 20 second call duration, the background sounds from a nearby radio is transmitted.

Text communication: We used the three services Messenger, Skype text, and Discord text. For each session, traffic was generated by sending and receiving four messages at random times during the 20 second session, where each message contains one to three words randomly selected from the list of the 10,000 most common English words.

Bulk download: For this category, we downloaded random applications from Google Play [21]. Here, random applications were identified using the search function together with random keywords from the 10,000 most common English words.

III. Multi-fractal feature based model

We next describe, step-by-step, how a model is created based on the multi-fractal features of a set of time series.

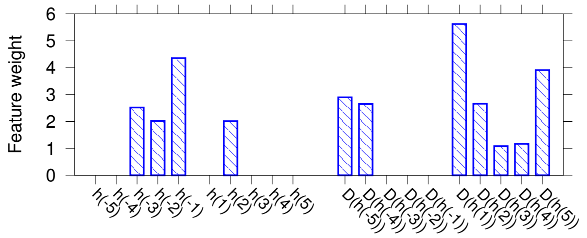


Fig. 2. Feature selection results using NCA.

A. Feature extraction

First, for each time series, we extract two sets of multi-fractal features, consisting of the estimated Holder exponents $h(q)$ and the Hausdorff dimensions $D(h(q))$, respectively, of the linearly-spaced moments q. More specifically, the features are extracted from the time series by computing the wavelet coefficients for different scales of the signal by using the Discrete Wavelet Transform (DWT) with the Symlet wavelet of order six.[2] From these, the time- or space-localized suprema of the coefficients, called the wavelet leaders, are calculated. The wavelet leaders are then used to form a multi-resolution structure function which in turn enables estimation of the scaling exponents by regression. Finally, from these, the Holder exponents $h(q)$ and the Hausdorff dimensions $D(h(q))$ are derived for the (integer) moments $q = -5$ to $q = 5$, $q \neq 0$. Consequently, each time series has 20 features.

B. Feature selection

Not all features are significant or add to the model. To remove non-contributing features, we performed a Neighborhood Component Analysis (NCA). An optimized value for the feature regularization parameter λ, was found by applying the limited memory BFGS optimization algorithm [22], and features with a feature value below 0.01 were removed.

Figure 2 shows example NCA results were we used half of the dataset for training and the feature selection regularization parameter λ was calculated to be $2.8 \cdot 10^{-4}$. Here, 11 features had a feature weight value higher than the threshold and were included in the model.

C. Support Vector Machine

A multi-class Support Vector Machine (SVM) [23] classifier was built based on multiple binary classifiers, were the values for the hyperparameters of the model were obtained by applying a Bayesian optimization algorithm (with the "expected-improvement" acquisition function). The SVM classifier was implemented in the Statistics and Machine Learning Toolbox (SMLT) in Matlab and the "fitcecoc" function in SMLT was used for multi-class model fitting. Table I shows example results where we used half of the samples for each class as training data. Based on these optimizations, the resulting model was then built using a radial basis kernel function with a kernel scale of 6.203, a box constraint (cost of misclassification) of 432.09, and a one-versus-one encoding. We note that the box constraint is the cost of misclassification (which will happen as the datasets are not perfectly separable), used to control how strict the separation of data needs to be, and the use of one-versus-one encoding simply means that binary classifiers were made for each pair of classes.

[2]Section IV evaluates and discusses the use of other wavelets.

TABLE I
THE MODEL HYPER PARAMETERS.

Parameter	**Value**
Kernel function	Radial basis function
Kernel scale	6.203
Box constraint	432.09
Multi-class encoding	One versus one

TABLE II
SUMMARY STATISTICS ACROSS TEN EXAMPLE EVALUATIONS.

Metric	**F1-score**	**Precision**	**Recall**
Average	0.960	0.959	0.961
Stdev	0.002	0.002	0.003
Min	0.956	0.956	0.955
Median	0.961	0.961	0.964
Max	0.960	0.959	0.961

D. Basic classification

Having built a model (e.g., as exemplified above), a new flow can quickly be classified by (i) monitoring the packet arrivals, (ii) creating a time series, (iii) extracting the time series multi-fractal spectrum, and (iv) feeding the spectrum into the model which classifies the flow. We next present the evaluation of such a classification model.

IV. Model evaluation

To evaluate the model, half of the samples were used to build the model and the other half was used for evaluation. For each classified flow in the evaluation set we then compared the model-based classification (using only the time series info) with the ground truth labels. This process was repeated for ten random sample sets and summary metrics were reported for all flows and broken down for each class.

A. Baseline comparison

Table II reports summary statistics (e.g., average, standard deviation, min, median, and max) for all flows and Table III breaks down the average results observed per class, as observed across the ten experiments. Here, we report the *precision* (i.e., the number of flows correctly classified as a class divided by the total number of flows classified as that class), the *recall* (i.e., the number of flows correctly classified as a class divided by the total of flows actually belonging to that class), and the *F1-score* (that incorporates both precision p and recall r according to the equation $F1 = \frac{2pr}{p+r}$).

First, note that the average F1-score is high (0.960) with small variations. The high F1-score shows that the model is effective and the small variation (e.g., as indicated by small min-max difference and the small standard deviation) suggests that the average results are significant to two decimals (e.g., the 95% confidence interval using the Student's t distribution is [0.959,0.961]).

TABLE III
PER-CLASS BREAKDOWN OF F1-SCORE, PRECISION, AND RECALL. (AVERAGES OVER 10 SAMPLES.).

Class	**F1-score**	**Precision**	**Recall**
Audio communication	0.986	0.988	0.986
Bulk download	0.994	0.989	0.996
Text communication	0.958	0.957	0.959
Social media	0.900	0.892	0.910
Video	0.958	0.962	0.954
Web	0.952	0.960	0.944

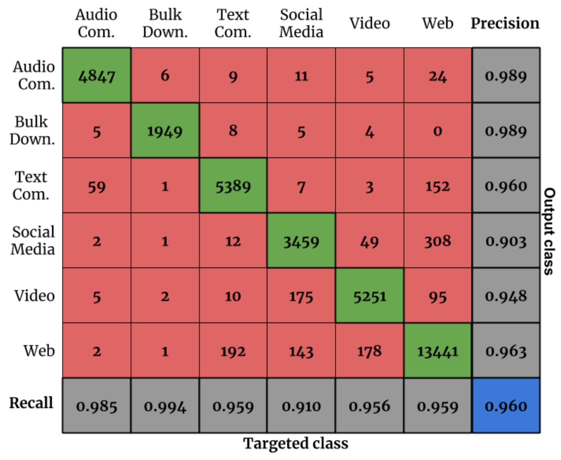
Looking closer at the per-class breakdown (Table II), we note that the *audio communication* and *bulk download* classifiers got very high F1-scores (0.986 and 0.994, respectively). Also the *text communication* (0.958), *video* (0.958), and *web* (0.952) classifiers performed very well. The worst performing classifier was the *social media* classifier, which sometimes got mixed up with *web* and *video*, both of which the *social media* sessions contain elements of. (These and other cross-category misclassifications are captured by the confusion matrix in Figure 3, where we have aggregated the results across all ten sample experiments.[3]) Yet, also this classifier performed well, achieving an average F1-score of 0.900.

To provide some intuition for how the classification is achieved Figure 4 visualizes the 11-dimensional dataset in 3-dimesnions using t-distributed Stochastic Neighbor Embedding (t-SNE). As part of this method we perform a Principal Component Analysis (PCA) on the 11 selected features to identify the principal components that explain most of the variance in the dataset, and then embedded data points into such a 3-dimensional space so that points in the 11-dimensional space also are close in this 3-dimensional space. We note that there are some significant clustering of data points associated with the same classes, suggest that these points likely are closeby in the 11-dimensional used by the SVM classifier.

B. Early detection

In the above evaluation we used the full 20-second traces. While 20 seconds provide fairly quick detection, an important question is how soon a purely time-series-based method, such as the one explored here, actually can provide accurate classification. Clearly, a shorter time-series duration would allow earlier predications, but would have less packets to base those predictions on. Table IV presents summary statistics when we use the first 20, 15, 10, 5, 2.5, 2, and 1 seconds of each flow for the training and evaluation. To provide a more fair comparison we kept the number of time-slots fixed at 20,000 (instead reducing the slot durations as we tried to make earlier predictions). We note that the model achieved an F1-score of 0.814 already after 5 seconds, suggesting that good accuracy can be achieved early in a flow (e.g., after only 5 seconds). Of course, such early classification can easily be combined with re-classification later in the flow (e.g., at 10 and 20 seconds), so to further improve the overall accuracy.

[3]The small differences observed between Table III and Figure 3 are due to the average of ratios not necessarily being equal to the ratio of sums.



	Audio Com.	Bulk Down.	Text Com.	Social Media	Video	Web	**Precision**
Audio Com.	4847	6	9	11	5	24	0.989
Bulk Down.	5	1949	8	5	4	0	0.989
Text Com.	59	1	5389	7	3	152	0.960
Social Media	2	1	12	3459	49	308	0.903
Video	5	2	10	175	5251	95	0.948
Web	2	1	192	143	178	13441	0.963
Recall	0.985	0.994	0.959	0.910	0.956	0.959	0.960

Output class
Targeted class

Fig. 3. Confusion matrix (based on an aggregate over all 10 samples).

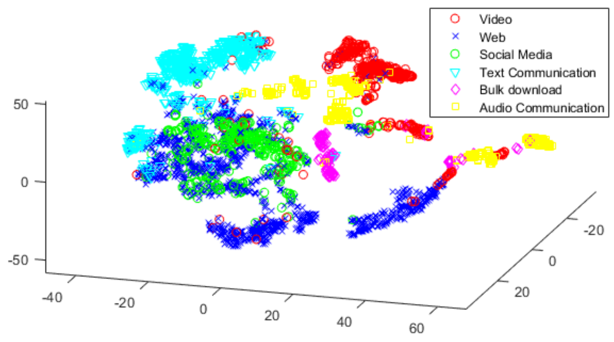


Fig. 4. t-SNE visualization of the chosen multi-fractal features.

C. The impact of added packet delay variation

To evaluate the impact of packet delay variations, the evaluation set was modified through random perturbations of the packet arrival times. In particular, for each of the packets in the time-series of the evaluation set, we drew a random number x from the normal distribution $\mathcal{N}(0, \sigma)$, and then "moved" the arrival time of this packet int(x) timeslots. Four our evaluation, we used $\sigma = 0$, 10, 25, 50, 100, 250, 500, and 1000. Table V presents the resulting F1-scores from these experiments. We note that the F1-score remains above 0.8 even for perturbations as large as with $\sigma = 500$ (in which case 31.8% of the packet arrivals are perturbed by more than ± 0.5 seconds).

D. Impact of other features and model choices

We have evaluated the model with many other features and model choices. Here, we briefly discuss some of our findings. First, the Symlet wavelet of order six typically gives the best result. For example, for our default experiments, a Symlet wavelet of order six had the highest F1-score (0.958) of all wavelets considered, which include all Daubechies wavelets of orders one-to-eight, all Symlets of orders one-to-eight, and all Coiflets of orders one-to-eight. As a comparison, we note that the best Daubechies wavelet (of order six) had an F1-score of 0.938, and the best Coiflet (of order five) had an F1-score of 0.943. Second, we have not found any added value from (i) adding the number of packets for each time slot (same combined F1-score of 0.958), (ii) adding the number of

TABLE IV
IMPACT OF THE DURATION ON THE F1-SCORE. (20,000 TIMESLOTS.)

Duration	**F1-score**	**Precision**	**Recall**
20 seconds	0.958	0.958	0.958
15 seconds	0.892	0.891	0.894
10 seconds	0.844	0.838	0.851
5 seconds	0.814	0.823	0.805
2.5 seconds	0.631	0.594	0.673
2 seconds	0.409	0.404	0.415
1 second	0.214	0.202	0.228

TABLE V
F1-SCORES WHEN PACKET ARRIVAL INSTANCES ARE PERTURBED ACCORDING TO A NORMAL DISTRIBUTION $\mathcal{N}(0, \sigma)$.

σ	10	25	50	100	250	500	1000
F1-score	0.952	0.942	0.925	0.927	0.891	0.834	0.695

transmitted bytes for each time slot ($F1$=0.949), (iii) adding the time since last packet in each time slot ($F1$=0.823), and (iv) multiplying all timeslots with bytes transmitted thus far ($F1$=0.951). Third, further improvements are possible if combining the multi-fractal features studied here with more traditional baseline features such as the total number of packets and the total number of bytes transmitted during the full time window (e.g., first 20 seconds). Figure 5 shows the results when adding these two factors to the model. Overall, we observe a slight increase in F1-score from 0.958 to 0.968.

V. ANOTHER USE CASE: VOD VS LIVE

Thus far we have evaluated the approach on classes that typically also could be classified on a per-service basis. In this section, we take the evaluation one step further and classify flows within the same set of services. In particular, we use the same methodology framework as explained in Sections II and III on sites that deliver both video on demand (VoD) and live streaming, often from the same IP addresses.

A. Data generation and collection

A dataset is first generated using the same setup as described in Section II, this time using only the three streaming services YouTube, Twitch and SVT play. In fact, for the VoD data we used the set of sessions collected above, and for the live streaming we randomly selected live streams from each application's list of current live streams. Table VI shows the breakdown of the 616 samples used for each class.

B. Model and feature selection

NCA resulted in a reduction from 20 to 7 multi-fractal features and the optimized SVM model resulted in a model with the same kernel function (radial basis function), but now with a kernel scale of 1.0213 and box constraint of 121.39.

C. Evaluation

Figure 6 shows the evaluation results. Here, the right column shows the precision of each classifier, the bottom row shows the recall of each classifier, and the cell in the bottom right shows the overall accuracy. The model managed an F1-score of 0.893, with similar performance of both classes. For a binary classifier this is a good result; showing that classification is possible with satisfying, but not perfect, accuracy.

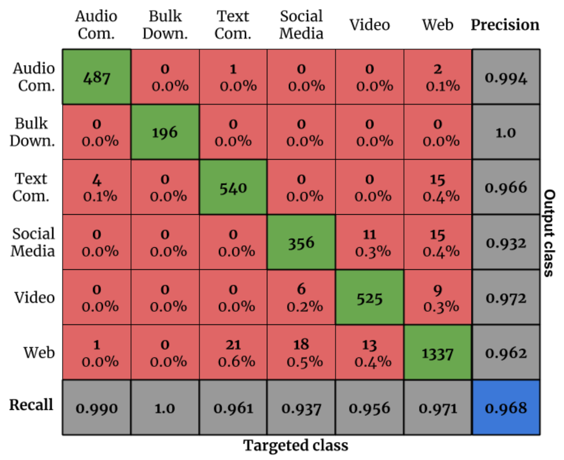


Fig. 5. Confusion matrix when evaluating the framework with the addition of two non-fractal features (based on the first example sample).

TABLE VI
NUMBER OF COLLECTED SAMPLES FOR EACH CLASS.

Class	**Samples**	**Class traffic composition**
Live	616	Youtube: 214, Twitch: 214, SVT play: 188
VoD	616	Youtube: 214, Twitch: 214, SVT play: 188

VI. RELATED WORK

Online flow classification has been used for many different application areas (e.g., security and management [24], accounting [25], and providing QoS guarantees [26]), efficient online performance has been demonstrated using many techniques (e.g., supervised techniques based on Naïve Bayes [27], and automated and semi-automated clustering techniques [10], [2], [28]), and solutions have been based on a wide range of features (ranging from simple flow-based metrics [3] to statistical analysis of specific properties [29], for example). The majority of this work ignores the burstiness of current internet traffic and do not capture its complex non-linear characteristics (e.g., self-similarity [12], [13], [14]).

The work closest to ours is the work by Shi et al. [15], who first demonstrated the value of using multi-fractal features for traffic classification. In their work, they show that multi-fractal features can be used to differentiate between traffic associated with different protocols. In this paper, we extend the framework for training and evaluation of encrypted traffic, and then show that the method effectively classifies encrypted traffic associated with delivery of different application types (rather than protocols) and demonstrate that the method is effective for early classification.

Encrypted traffic has also been classified by others. Pan et al. [30] shows that characteristics in the SSL/TLS handshake process can be used to discriminate between 12 websites and Muehlstein et al. [31] used 53 features calculated based on transport layer attributes to discriminate between YouTube, Facebook and Twitter. Other researchers have shown that, even when the data is encrypted, it is possible to identify the operating system, browser and the application of unknown hosts [31], the language used on an encrypted VoIP channel [32], what Netflix videos users watch [33], and even to estimate the QoE and buffer conditions of video streaming clients [34], [35].

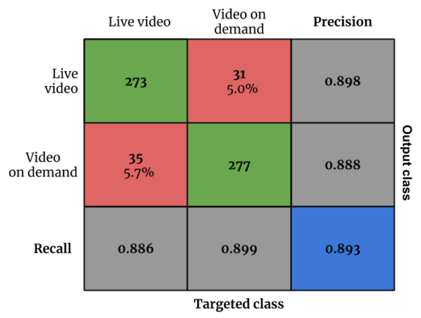


Fig. 6. The resulting confusion matrix from evaluating the model.

VII. Conclusion

This paper presents and evaluates a classification framework that combines multi-fractal feature extraction based on time series data (that captures the non-linear characteristics of internet traffic), PCA-based feature selection, and MITM-based flow labeling. Our detailed evaluation shows that the method is able to quickly and effectively classify traffic belonging to the six most popular traffic types (video streaming, web browsing, social networking, audio communication, text communication, and bulk download) and to distinguish between VoD and live streaming sessions delivered from the same services. The high accuracy for the second of these use cases are particularly encouraging as the three streaming services considered (YouTube, Twitch, and SVT play) use HAS-based streaming for both their VoD and live contents. Furthermore, since the method only requires access to timing information of the packets within a flow, it is significantly more future-proof than approaches that rely on DPI (already not possible due to high HTTPS/QUIC usage) and port numbers (easily obscured and typically the same across classes as the majority of traffic is HTTP-based anyway today, although often delivered over TLS or QUIC).

Acknowledgments

The authors are very thankful to Peter Alvarsson and Eric Henziger for their feedback and help with this work. This work was funded in part by the Swedish Research Council (VR).

References

[1] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson, “Identifying and discriminating between web and peer-to-peer traffic in the network core,” in *Proc. WWW*, 2007.
[2] L. Bernaille, R. Teixeira, and K. Salamatian, “Early application identification,” in *Proc. ACM CoNEXT*, 2006.
[3] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “BLINC: Multilevel traffic classification in the dark,” in *Proc. ACM SIGCOMM*, 2005.
[4] A. W. Moore and K. Papagiannaki, “Towards the accurate identification of network applications,” in *Proc. PAM*, 2005.
[5] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, “ACAS: automated construction of application signatures,” in *Proc. ACM SIGCOMM workshop on Mining network data*, 2005.
[6] A. P. Felt, R. Barnes, A. King, C. Palmer, C. Bentzel, and P. Tabriz, “Measuring HTTPS adoption on the web,” in *Proc. USENIX Security Symposium*, 2017.
[7] M. Soysal and E. G. Schmidt, “Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison,” *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.
[8] 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.
[9] 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 *Proc. ACM IMC*, 2004.
[10] J. Erman, M. Arlitt, and A. Mahanti, “Traffic classification using clustering algorithms,” in *Proc. ACM SIGCOMM Workshop on Mining Network Data*, 2006.
[11] B. Claise, “Cisco systems Netflow services export version 9,” RFC 3954, Tech. Rep., 2004.
[12] W. E. Leland, W. Willinger, M. S. Taqqu, and D. V. Wilson, “On the self-similar nature of Ethernet traffic,” *ACM SIGCOMM CCR*, vol. 25, no. 1, pp. 202–213, 1995.
[13] A. Feldmann, A. C. Gilbert, and W. Willinger, “Data networks as cascades: Investigating the multifractal nature of internet WAN traffic,” *ACM SIGCOMM CCR*, vol. 28, no. 4, pp. 42–55, 1998.
[14] A. Feldmann, A. C. Gilbert, W. Willinger, and T. G. Kurtz, “The changing nature of network traffic: Scaling phenomena,” *ACM SIGCOMM CCR*, vol. 28, no. 2, pp. 5–29, 1998.
[15] H. Shi, H. Li, D. Zhang, C. Cheng, and W. Wu, “Efficient and robust feature extraction and selection for traffic classification,” *Comput. Netw.*, vol. 119, no. C, pp. 1–16, 6 2017.
[16] mitmproxy. [Online]. Available: https://mitmproxy.org/
[17] Splitcap. [Online]. Available: https://www.netresec.com/?page=SplitCap
[18] Sandvine, “2016 Global internet phenomena report, Latin America and North America,” Tech. Rep., 2016.
[19] V. Krishnamoorthi, N. Carlsson, and E. Halepovic, “Slow but steady: Cap-based client-network interaction for improved streaming experience,” in *Proc. IEEE/ACM IWQoS*, 2018.
[20] P. Norvig. Natural language corpus data: Beautiful data. Visited on 01/06/2018. [Online]. Available: http://norvig.com/ngrams/
[21] Google Inc. Google Play. [Online]. Available: https://play.google.com/
[22] D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization,” *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.
[23] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
[24] R. Bendrath and M. Mueller, “The end of the net as we know it? deep packet inspection and Internet governance,” *New Media & Society*, 2011.
[25] A. Ramachandran, S. Seetharaman, N. Feamster, and V. Vazirani, “Fast monitoring of traffic subpopulations,” in *Proc. ACM IMC*, 2008.
[26] N. Cascarano, L. Ciminiera, and F. Risso, “Optimizing deep packet inspection for high-speed traffic analysis,” *Journal of Network and Systems Management*, vol. 19, no. 1, pp. 7–31, 2011.
[27] A. Moore and D. Zuev, “Internet traffic classification using Bayesian analysis techniques,” in *Proc. ACM SIGMETRICS*, 2005.
[28] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, “Robust network traffic classification,” *IEEE/ACM Trans. Netw.*, vol. 23, pp. 1257–1270, 2015.
[29] S. Zander, T. Nguyen, and G. Armitage, “Automated traffic classification and application identification using machine learning,” in *Proc. IEEE LCN*, 2005.
[30] W. Pan, G. Cheng, and Y. Tang, “WENC: HTTPS encrypted traffic classification using weighted ensemble learning and Markov chain,” in *Proc. IEEE Trustcom/BigDataSE/ICESS*, 2017.
[31] J. Muehlstein, Y. Zion, M. Bahumi, I. Kirshenboim, R. Dubin, A. Dvir, and O. Pele, “Analyzing HTTPS encrypted traffic to identify user's operating system, browser and application,” in *Proc. IEEE CCNC*, 2017.
[32] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson, “Language identification of encrypted VoIP traffic: Alejandra y roberto or alice and bob?” in *Proc. USENIX Security Symposium*, 2007.
[33] A. Reed and M. Kranich, “Identifying HTTPS-protected Netflix videos in real-time,” in *Proc. ACM CODASPY*, 2017.
[34] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, “BUFFEST: predicting buffer conditions and realtime requirements of HTTP(S) adaptive streaming clients,” in *Proc. ACM MMSys*, 2017.
[35] M. H. Mazhar and Z. Shafiq, “Real-time video quality of experience monitoring for HTTPS and QUIC,” in *Proc. IEEE INFOCOM*, 2018.