Video Quality Representation Classification of Encrypted HTTP Adaptive Video Streaming

Ran Dubin[1], Ofer Hadar[1], Amit Dvir[2], Ofir Pele[2,3]
[1] Communication Systems Engineering, Ben-Gurion University of the Negev, Israel
[e-mail: ofer@bgu.ac.il, dubinr@post.bgu.ac.il]
[2] Center for Cyber Technologies, Department of Computer Science, Ariel University, Israel
[e-mail: amitdv@ariel.ac.il]
[3] Department of Electrical and Electronics Engineering, Ariel University, Israel
[e-mail: ofir.pele@g.ariel.ac.il]
*Corresponding author: Amit Dvir

Received January 16, 2017; accepted March 14, 2018; published August 31, 2018

Abstract

The increasing popularity of HTTP adaptive video streaming services has dramatically increased bandwidth requirements on operator networks, which attempt to shape their traffic through Deep Packet inspection (DPI). However, Google and certain content providers have started to encrypt their video services. As a result, operators often encounter difficulties in shaping their encrypted video traffic via DPI. This highlights the need for new traffic classification methods for encrypted HTTP adaptive video streaming to enable smart traffic shaping. These new methods will have to effectively estimate the quality representation layer and playout buffer. We present a new machine learning method and show for the first time that video quality representation classification for (YouTube) encrypted HTTP adaptive streaming is possible. The crawler codes and the datasets are provided in [43,44,51]. An extensive empirical evaluation shows that our method is able to independently classify every video segment into one of the quality representation layers with 97% accuracy if the browser is Safari with a Flash Player and 77% accuracy if the browser is Chrome, Explorer, Firefox or Safari with an HTML5 player.

Keywords: Machine Learning, Quality Representation Classification, HTTPS Video Streaming, Encrypted Traffic, YouTube

A preliminary version of this paper appeared in IEEE DMIAF, June 4-6, Santorini, Greece. In this version we added a testbed that has all four leading browsers with a HTML5 player (Chrome, Explorer, Firefox and Safari) whereas in the DMAIF paper we had only Safari with a Flash player; The new testbed is much larger and contains 500 streams of 100 movies, whereas the DMAIF paper had only 120 streams and 40 movies; We also extended the analysis of our algorithm with a different network conditions, different training size.

http://doi.org/10.3837/tiis.2018.08.014 **ISSN : 1976-7277**

1. Introduction

YouTube now occupies a market share of over 17% of the total mobile network bandwidth in North America [1]. In general, by 2020, their share of video traffic is expected to increase to 82% of the total IP traffic and approximately 135 exabytes per month [1]. Currently, YouTube and other video streaming websites use Hypertext Transfer Protocol (HTTP) Adaptive Streaming (HAS). Today, the de facto standard method for HAS is Dynamic Adaptive Streaming over HTTP (DASH) [2]. DASH is a Multi Bit Rate (MBR) streaming method designed to improve viewer Quality of Experience (QoE) [3].

In DASH, each video is divided into short segments, typically a few seconds long, and each segment is encoded several times, each time with a different quality representation. The user player Adaptation Logic (AL) algorithm is responsible for the automatic selection of the most suitable quality representation for each segment, based on several parameters such as the frequency of switching video quality level events [52], time-varying distortions [53], crowd sourcing [54], network conditions and the client playout buffer [4] – [8]. Thus, the quality representation in DASH can change between segments. In DASH, each quality representation is encoded at variable bit rates (VBRs). VBR varies the amount of output data per time segment and does not attempt to control the output bit rate of the encoder, so that the distortion will not vary significantly. DASH often uses the HTTP byte range mode. In this mode, the byte range of each segment request can differ.

A video quality representation classification of encrypted video streams can help ISPs to shape user traffic. ISPs face increasing competition, declining profitability and increasing client demands for network bandwidth (BW). Thus ISPs must optimize their network traffic, and video streaming has become the solution of choice. Network traffic is optimized by limiting user flows for a specific application. Shaping DASH encrypted traffic without severely reducing user QoE cannot be achieved without estimating the user playout buffer and classifying the user selected video quality representation.

Google encourages all website owners to switch from HTTP to HTTPS by integrating whether sites use secure, encrypted connections as a signal in their ranking algorithms [9]. Google has also started to encrypt their video services [9]. As a result, traditional Deep Packet Inspection (DPI) methods for data mining and classification, in general, and video quality representation classification, in particular, are not possible. Another challenge for quality representation classification comes from the development of new application layer network communication protocols such as SPDY [10] and HTTP2 [11]. These protocols include features such as header compression, request multiplexing/interleaving between two endpoints, and the ability to push content from the server side.

Studies have been conducted on YouTube in terms of server location [12], [13], YouTube vs. other video sharing services [12], Personal Computer (PC) vs. mobile user access patterns [14], QoE [15], traffic characterization and its DASH implementation, [16] and network analysis [17] – [21]. Many recent works have suggested methods for encrypted traffic classification and several surveys have presented detailed descriptions of state of the art methods [22] – [24]. Several have examined statistical features [25] – [35]. Some are not pertinent to video stream classification: the packet is often Maximum Transmission Unit (MTU) size in video streaming, as it consumes high bandwidth and re-transmission occurs often and Transmission Control Protocol (TCP) parameters such as server sent bit rate,

inter-arrival packet time, Round Trip Time (RTT) and packet direction are weak features in terms of video quality classification. To the best of our knowledge, this work is the first to classify quality representation of encrypted YouTube streams.

In this paper, we present a video stream quality representation classification for DASH. We classify the video quality representation, and each feature (group of packets) is classified by itself without any dependencies on past or future samples. For Safari with a Flash Player, our novel algorithm achieved 97% accuracy. When the browser was one of the four most popular browsers (Chrome, Explorer, Firefox or Safari) with an HTML5 player, a nearest neighbor algorithm combined with our feature extraction method achieved 77% accuracy.

A previous version of this work was published in [36]. In the revised version presented here, we have added a testbed that has all four leading browsers (Chrome, Explorer, Firefox, and Safari) with a HTML5 player whereas in [36] we only tested Safari with a Flash Player. The new testbed is much larger and contains 500 streams of 100 movies, whereas [36] only had 120 streams and 40 movies. We also extended the analysis of our algorithm to include different network conditions, different training dataset sizes, and provide a user buffer estimate using quality representation classification. The remainder of this paper is organized as follows: **Section 2** presents our new framework, **Section 3** presents the performance evaluation, and **Section 4** discusses our conclusions, limitations, and future work.

2. Video Quality Representation Classification

A DASH server stores a video which is segmented into fixed duration segments. Each segment is encoded into *m* representations (*m* can be different for different videos). The user can select to download the video in two different modes of operation: fixed or automatic (auto). In the fixed mode the client selects a single VBR quality representation layer. In our fixed scenario the user selects a single quality from the beginning tothe end of the video. In the auto mode, the client's video player application (via adaptation logic) selects the optimal representation layer to download. Each segment can be further divided into several smaller chunks (parts of a segment). The following sections describe our encrypted traffic quality representation classification method.

2.1 Preprocessing and Feature Extraction

Encrypted traffic generally relies on Secure Sockets Layer (SSL)/Transport Layer Security(TLS) for secure communication. These protocols are built on top of the TCP/IP suite. The TCP layer receives encrypted data from the above layer and divides the data into chunks if the packets exceed the Maximum Segment Size (MSS). Then, for each chunk it adds a TCP header, creating a TCP segment. Each TCP segment is encapsulated into an Internet Protocol (IP) datagram. Since TCP packets do not include a session identifier, we divided the traffic into flows based on a five-tuple representation: {protocol (TCP/UDP), src IP, dst IP, src port, dst port}. Next, for each flow we determined whether it was a YouTube flow or not,based on the Service Name Indication (SNI) field in the *Client Hello* message. If the "*googlevideos.com*" string wasfound in the SNI, the flow was passed on to the next module. Note that the YouTube flow identification can also be executed using machine learning techniques [37], [38].

To better understand encrypted YouTube streaming traffic properties, we examined YouTube traffic under different browsers. Examples of YouTube flows can been seen in **Fig. 1**. The figure depicts the bursty behavior of browsers, where there are periods of data

transmissions with silence (zero bits transmission) before and after. A period of data transmission is called a peak.

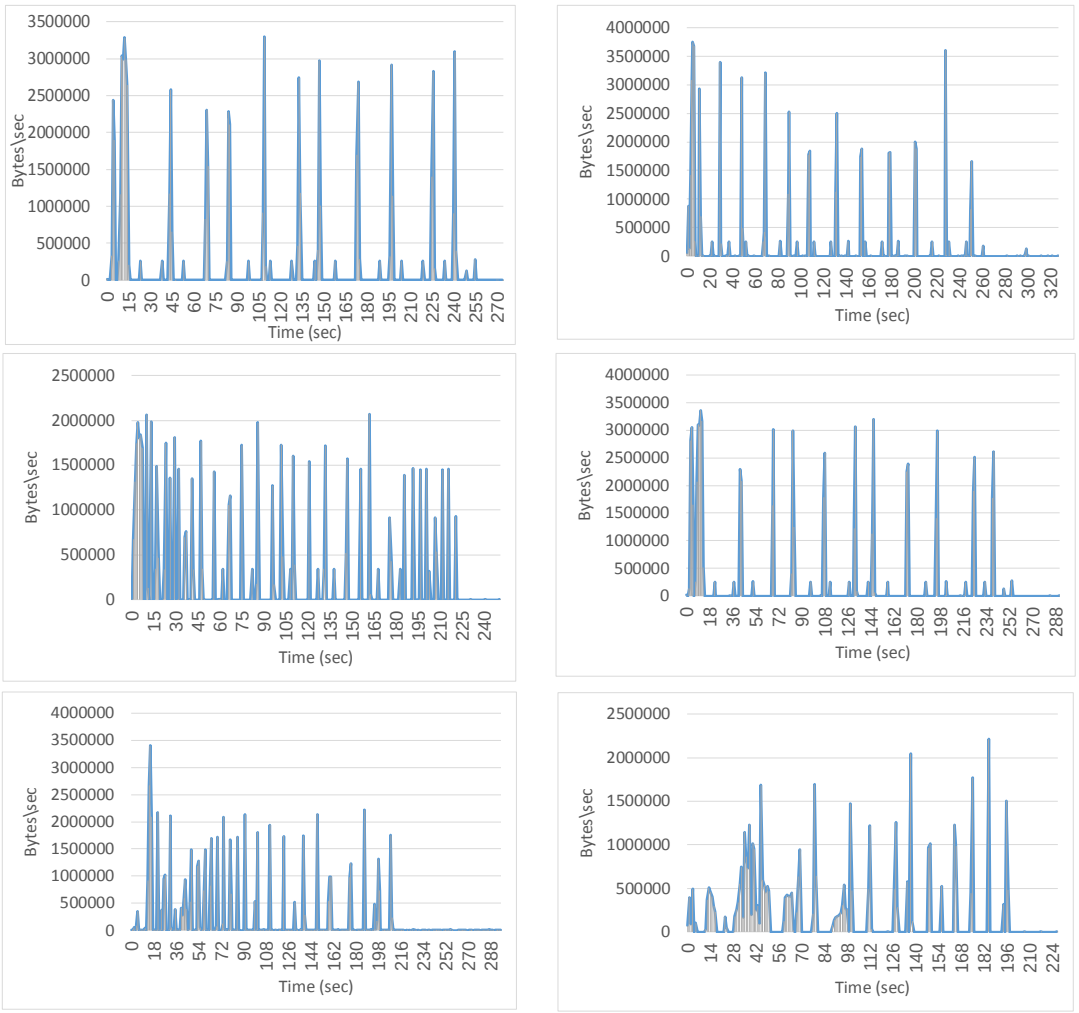


Fig. 1. Typical examples of traffic flows of auto mode downloads of the same movie using various browsers (Safari, Chrome, Firefox, and Explorer) and various players (Flash and HTML5 players) under various network conditions. The top two flows were downloaded with an ADSL connection and the four at the bottom were downloaded with a Wi-Fi connection. We used Wireshark [49] to collect data. All the flows have the same characteristics; i.e., peaks (of packets) with silences before and after. Note that the differences between flows may be caused by auto mode, network conditions, player algorithm, etc.

Our previous works also used these peak features to classify video titles [55] and operating systems, browsers, and applications [48]. Note that the bursty behavior of browser traffic was observed for YouTube traffic in [16], [40] – [41]. Because audio data and video data can be found in the same 5-tuple flow, we removed audio packets. We also removed the first and last peak in each flow. The first peak is large and bound by the TCP window while the player fills the buffer; thus the quality is not related to the peak volume [17]. Since our method constitutes the first step in traffic shaping and the last peak is not relevant for traffic shaping it was

removed. Finally, TCP re-transmissions were removed using a TCP stack [39] because re-transmissions are caused mostly by network conditions.

The feature extraction was carried out on the preprocessed traffic, after non-YouTube flows, most of the audio packets, the first and last peaks, and TCP re-transmissions were removed. We decided to encode the remaining peaks of the streams into features. This feature was dubbed *Bit Per Peak* (*BPP*) and is the sum of bytes in each peak:

$$BPP(peak) = \sum_{t=beginTime\ (peak)}^{endTime\ (peak)} bitsRecieved(t) - bitsReTransmitted(t)$$

where before *beginTime(peak)* and after *endTime(peak)* there are periods of silence (zerobit transmission).

2.2 Codebook Learning Algorithm

The classification solution is illustrated in **Fig. 2**. It has a training step and a testing step. In the training step, we first constructed our dataset based on YouTube video streaming captures (PCAP trace files). Each video was downloaded with the three following fixed qualities {360*P*, 480*P*, 720*P*}. The data were preprocessed and the *BPP* features were extracted. Afterwards, the training data were clustered using *k*-means++ [45] (step (3) in **Fig. 2**). The end product of these steps was a codebook that represents the training dataset.

For each quality and for each time index, we computed the average *BPP*. This yielded an average *BPP* vector (whose length is the maximum time index) for each quality. We then computed a representative string for each quality from these vectors and using the codebook from the *k*-means stage. In the classification stage we carried out the *BPP* extraction for each segment and then assigned a symbol (the one with the shortest distance from the average) to it from the codebook. Finally, we assigned a label by finding which center was the closest.

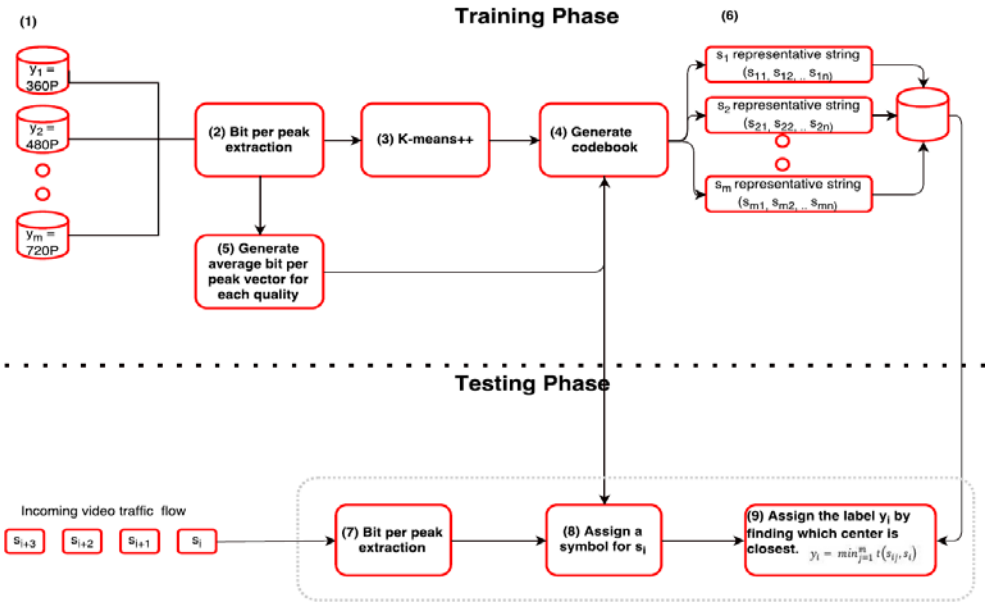


Fig. 2. Codebook learning algorithm diagram flow. The crawler code for this framework is provided in [43]

3. Performance Evaluation

In this section, an evaluation of the quality representation classification scheme is presented. First, the Safari with Flash Player dataset in **Section 3.1** is described. **Section 3.2** evaluates the performance on Safari with Flash Player. **Section 3.3** presents a simple method for user buffer estimation that implements our quality representation classification method and reports the accuracy of the estimate. Finally, the HTML5 dataset is described in **Section 3.4** and the results appear in **Section 3.5**.

3.1 Safari with Flash Player Dataset

The video titles used in this study are popular YouTube videos from different categories such as news, video action trailers, and GoPro videos. We show that for Safari with Flash Player, we can learn an accurate model for static or automatic quality modes simply by using a fixed training dataset. The training dataset contained 120 video streams of 40 unique video titles, each of which was separately downloaded with a fixed quality from the following qualities: {*360P, 480P, 720P*}. The dataset was manually downloaded using Wi-Fi networks from different public locations via the Internet. We provide the dataset and all video information including names, duration (between 01:18 – 08:15 minutes), and frame rate (24 or 30 frames per second) in [43].

We had three testing datasets:

1) *test-fixed-train-titles*: 120 video streams of 40 unique video titles (same titles as in the training phase) each of which was separately downloaded with a fixed quality from the following qualities: {*360P, 480P, 720P*}.
2) *test-adaptive-train-titles*: Five video streams of five unique video titles (titles taken from the training phase titles) each of which was downloaded with an adaptive quality representation (auto mode)
3) *test-adaptive-test-titles*: Five video streams of five unique video titles (new titles that were not in the training phase) each of which was downloaded with an adaptive quality representation (auto mode).

Note that all the test video streams were *different* from the ones that were used in the training phase (because of network conditions).

3.2 Evaluation and Comparison for the Safari with Flash Dataset

Our codebook learning algorithm clusters the bit rates into k bins. To choose this number, we used 5-fold cross validation on the training dataset. Cross validation is a well-established method for choosing hyperparameters of a machine learning method [55] - [56]. In the 5-fold cross validation method for choosing hyperparameters, the training data are randomly partitioned into five equal sized subsamples. Of the five subsamples, a single subsample is used as the validation data for testing the model, and the remaining four subsamples are used as training data for each of the hyperparameters we want to choose from. This process is repeated five times, where each time a different subsample is used for testing. Finally, the hyperparameter that achieves the best average identification rate is chosen and the entire training dataset is trained with it. **Fig. 3** shows the average cross validation identification rate on the training dataset for each of the k values: {5,10,13,14,15,16,17,20}. It can be seen that k

= 14 achieved the highest cross validation identification rate on the training dataset. The figure also shows that our codebook learning algorithm was not sensitive to the exact value of number of bins (*k*) chosen. In the following experiments we used *k = 14*.

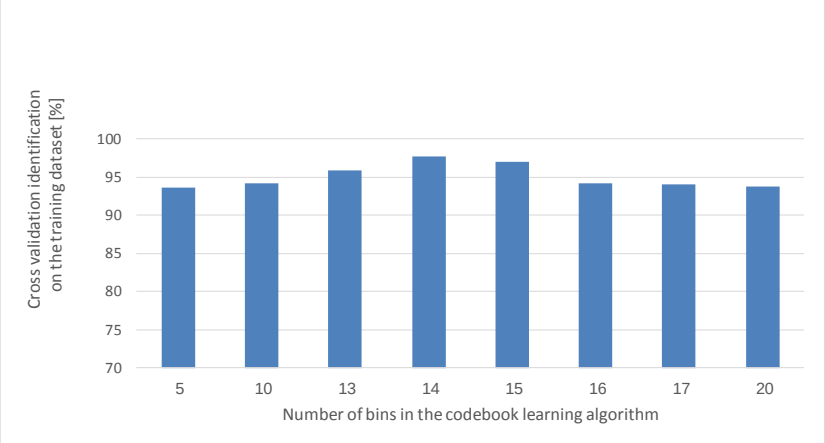


Fig. 3. Codebook learning algorithm cross validation identification rate on the training dataset for a different number of bins in the codebook learning algorithm

Fig. 4 shows the performance of our codebook learning algorithm for various training dataset sizes on the *test-fixed-train-titles* test dataset. Accuracy improved when not using the last peak. For each combination of method and training size, the mean of five different random training-testing splits and its normal confidence interval with a significance level of 95 percent is depicted. There weremajor gains in performance when the number of training video titles increased from 5 to 10. The gains were smaller when the number of training video titles increased from 10 to 40. Because the last peak size varied (because it corresponds to the stream leftovers) it decreased the identification rate. Finally, the cross validation identification rate (**Fig. 3**) emerged as a good estimate of the real testing error.

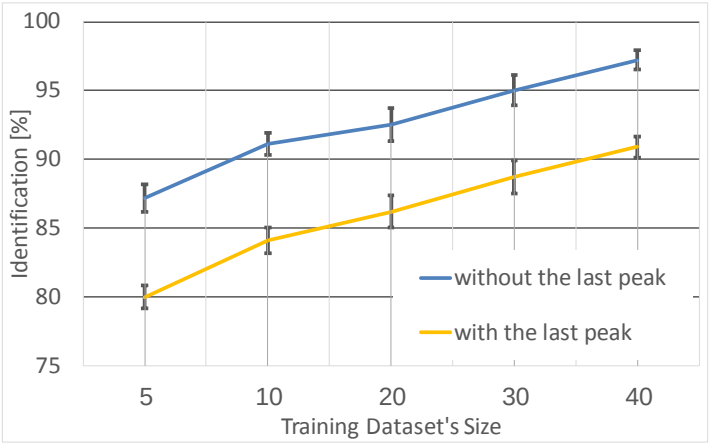


Fig. 4. Codebook learning algorithm testing identification rate on the *test-fixed-train-titles* dataset using various training dataset sizes and with/without the last peak

We next evaluated our codebook learning algorithm with our *BPP* feature on the various test datasets from **Section 3.1** (**Fig. 5a, 5d, 5e**) and compared it to several classification approaches: (1) Nearest neighbor using the average bit rate as a feature (**Fig. 5b**); (2) An adaptation of a network traffic malware fingerprinting algorithm [46] with our *BPP* feature

(**Fig. 5c**); (3) Our codebook learning algorithm with a time difference feature (**Fig. 5f**); (4) An adaptation of a network traffic malware fingerprinting algorithm [46] with a time differences feature (**Fig. 5g**). Our codebook learning algorithm with our *BPP* feature achieved the highest identification results in contrast to all the other algorithms, especially using time differences which obtained much lower identification results. **Fig. 5a** shows that our classification errors for the *test-fixed-train-titles* dataset were between close quality representations, and the average error rate was less than 4%. **Fig. 5b** shows that the nearest neighbor algorithm with an average bit rate feature had more than an 11% error rate. **Fig. 5c** shows that an adaptation of a network traffic malware fingerprinting algorithm [46] that used our *BPP* feature had more than a 19% error rate. Moreover, in some cases, it even confused 360P and 720P quality representations. **Fig. 5d** shows that our method was able to classify adaptive streams with high accuracy, achieving an error rate of less than 4%. **Fig. 5e** shows that our method accuracy remainedhigh when the video titles in the training set and the testing set differed, with an error rate of 12%. We examined why the mistaken classification of the 480P quality representation segments as 720P in adaptive streams was relatively high and found that when the quality representation switched from 360P to 480P there were high bit rate bursts causing the erroneous classification of these segments as 720P. Finally, we also evaluated the quality representation classification performance when using time differences as a feature. **Fig. 5f** and **Fig. 5g** show that this feature was unsuited to quality representation classification.

	360	480	720
360	0.97	0.03	0.00
480	0.01	0.97	0.02
720	0.00	0.03	0.97

(a) Codebook learning algorithm using *BPP* feature, *test-fixed-train-titles*

	360	480	720
360	0.96	0.04	0.00
480	0.08	0.84	0.08
720	0.00	0.15	0.85

(b) Nearest Neighbor using Average Bit Rate Feature(naïve), *test-fixed-train-titles*

	360	480	720
360	0.89	0.06	0.05
480	0.16	0.78	0.06
720	0.02	0.22	0.76

(c) Learning algorithm from [46] using *BPP* feature, *test-fixed-train-titles*

	360	480	720
360	1.00	0.00	0.00
480	0.00	0.94	0.06
720	0.00	0.04	0.96

(d) Codebook learning algorithm using *BPP* feature, *test-adaptive-train-titles*

	360	480	720
360	0.96	0.04	0.00
480	0.08	0.84	0.08
720	0.00	0.16	0.84

(e) Codebook learning algorithm using *BPP* feature, *test-adaptive-test-titles*

	360	480	720
360	0.66	0.26	0.08
480	0.15	0.60	0.25
720	0.03	0.35	0.62

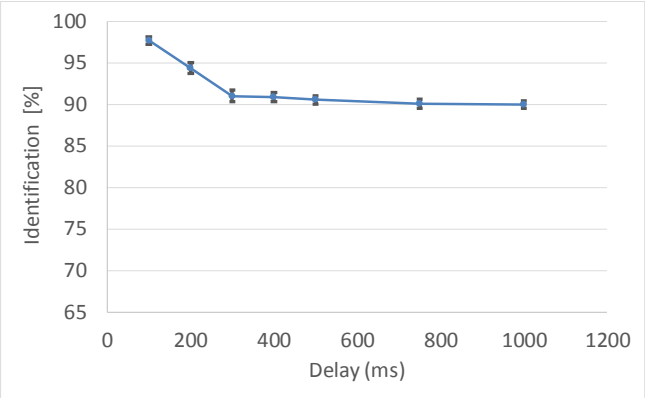
(f) Codebook learning algorithm using time differences feature, *test-fixed-train-titles*

	360	480	720
360	0.39	0.52	0.09
480	0.31	0.37	0.32
720	0.13	0.49	0.38

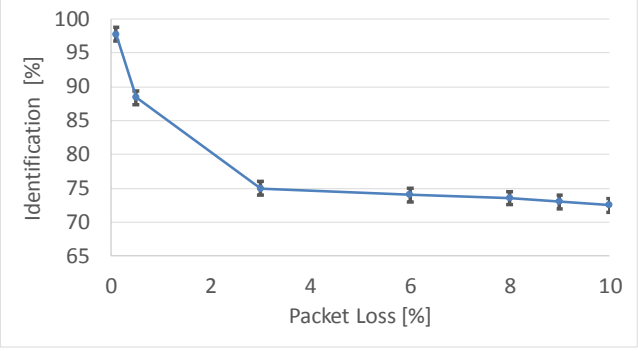
(g) Learning algorithm from [46] using time differences feature, *test-fixed-train-titles*

Fig. 5. Confusion matrices for various learning methods and various test datasets. Rows refer to the real quality, and the columns refer to the predicted quality.

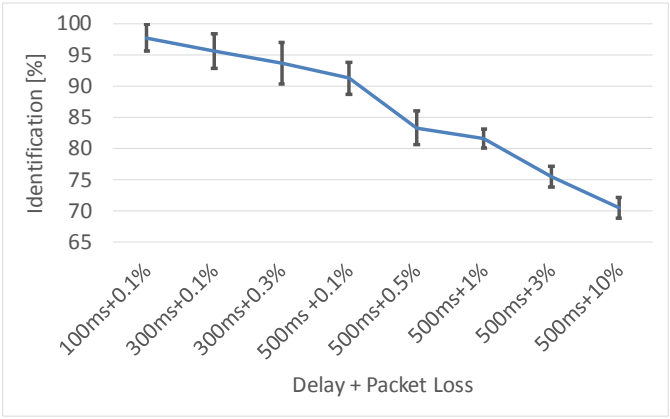
We also tested the classifier (trained under normal network conditions) on data to which packet loss and delays were added. Noise was added using the Clumsy network controller [50] based on the Nokia (Alcatel-Lucent) findings presented in [64]. In general, added noise can cause the client player to select lower qualities. Delays and packet losses can cause a new silent period (since the *BPP* feature is an aggregation of information from several packets) thus splitting peaks and changing *BPP* feature values. Packet losses and delays at test time indeed reduced accuracy, as can be seen in **Fig. 6**. The figure shows the means of five different random training-testing splits for each network condition and their normal confidence interval at a significance level of 95 percent. **Fig. 6a** illustrates our algorithm's robustness to network delays at test time. With 300 millisecond additional delays, there was a 6.3% decrease in classification accuracy. Adding more delay only caused a 0.7% decrease in the classification accuracy. **Fig. 6b** illustrates our robustness to packet loss at test time. The figure shows that a packet loss of 3% decreased classification accuracy by 20%. In the case of a 10% packet loss (when the video is practically halted) the classification accuracy decreased to 73%. **Fig. 6c** illustrates our algorithm's robustness to combinations of network delays and packet losses at test time. The figure shows that 500ms delays plus 10% packet losses decreased our classification to 70%. However, in real life scenarios it would be impossible to watch the video (overly low QoE). Thus overall, our algorithm emerged as robust to moderate changes in network conditions at test time.



(a) Codebook learning algorithm's identification rates for streams with different network delays



(b) Codebook learning algorithm's identification rates for streams with different percentages of packet loss events



(c) Codebook learning algorithm's identification rates for streams with different combinations of network delays and percentages of packet loss events.

Fig. 6. Codebook learning algorithm's identification rates under different network conditions

3.3 User Buffer Estimates implementing Quality Representation Classification

Estimating the user playout buffer level is crucial to evaluating user QoE, which is key to ISP optimization of service. We estimated the user buffer with our quality representation classification method. On the training set, for each peak index (where index 1 is the first peak of a video stream) and for each quality representation, the average duration which corresponds to the buffer size was calculated. Note that duration in this section refers to playout duration and not download duration. At testing time, the user buffer size was estimated from the duration that was measured on the training set for its peak index and its predicted quality representation. This user buffer estimation method was tested on Safari with a union of all Flash Player test datasets. The average distance between the estimates and real buffers per peak was less than 2% (0.035 seconds) and the STD was less than 4% (0.047 seconds). **Fig. 7** shows the average peak duration, which was much higher than 0.035, indicating that the resulting average difference of 0.035 seconds per feature in our buffer estimation method was very small, making our method very accurate.

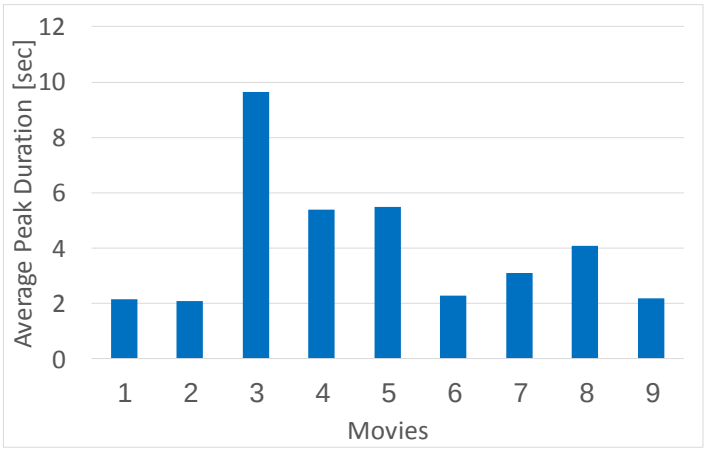


Fig. 7. Average peak duration of movies

3.4 HTML5 Player Browsers Dataset

As in the Safari with Flash Player dataset, the video titles used are popular YouTube videos from different categories such as news, video action trailers, and GoPro videos. The HTML5 dataset was much larger and more varied than the Safari with Flash Player dataset. The dataset contained 500 video streams of 100 unique video titles each of which was separately downloaded with automatic quality with an HTML5 player. The frame rate of each video was 24, 25 or 30 FPS. We used the Selenium web automation tool [47] to simulate a normal user video download. To collectthis dataset, we developed a system illustrated in **Fig. 8**. In step (1) the URL to download is selected. Step (2) initiates network recording using Tshark [49]. In step (3) we select which browser to use. In step (4) we connect the browser to a proxy web server which saves all the browser's requests and responds using the HTTP Archive (HAR) metadata format. We extract traffic features and label them using itags, where itags 242, 244 and 248 correspond to resolutions: 240P, 480P, 1080P respectively. Note that at testing time these itags are not available. The dataset is provided with all video information including names, duration (between 01:15 – 10:33 minutes), frame rate (24, 25 or 30 frames per second) in [44]; the crawler code is provided in [51].

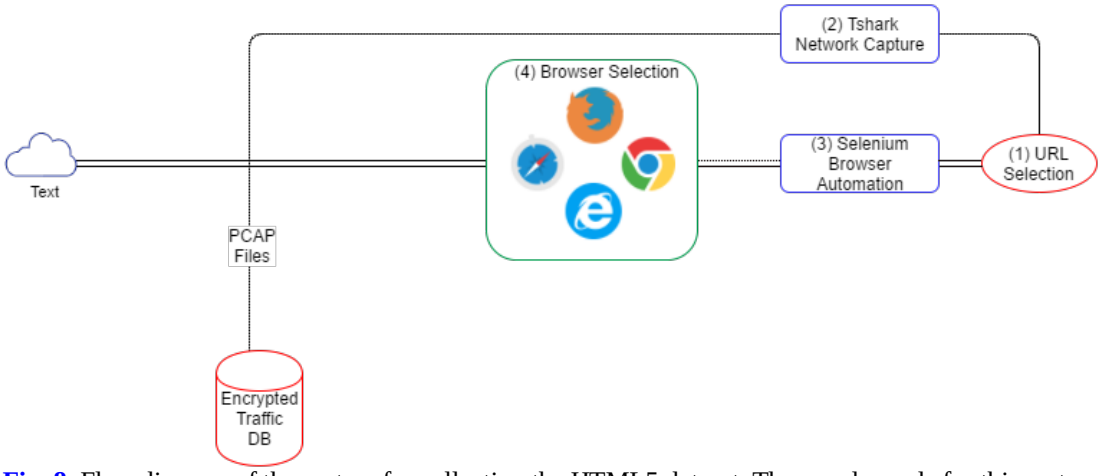


Fig. 8. Flow diagram of the system for collecting the HTML5 dataset. The crawler code for this system is provided in [51].

3.5 HTML5 Player Browsers Results

Using the codebook learning algorithm (**Sec 2.2**) for all leading browsers (Chrome, Explorer, Firefox, and Safari with HTML5 player) did not improve the results over a simple 1-Nearest-Neighbor algorithm. Using a 1-Nearest-Neighbor algorithm coupled with our feature extraction method achieved 77% accuracy in a leave- one- out validation study. That is, we iterated over the database and in each iteration one different sample was used as the testing sample, and all the other samples were used as the training dataset. In each iteration, the Euclidean distances of the test sample to all other samples are computed. The label of the test sample is the nearest neighbor sample's label. The average accuracy of our method was 77%.

The is lower than on the Safari with Flash dataset. The main difference between the HTML5 player and the Flash Player is that the HTML5 uses the HTTP byte range. In this mode, the byte range of each segment request can be different. This causes a higher variability between *BPP*s values and leads to a lower identification rate. However, the identification rate is still reasonable and, as can be seen in **Fig. 9**, the errors were mainly between close quality representations.

	240	480	1080
240	0.68	0.20	0.12
480	0.00	0.79	0.21
1080	0.00	0.17	0.83

Fig. 9. The confusion matrix for the 1-Nearest-Neighbor algorithm on the HTML5 Browsers dataset. Rows refer to the real quality, and the columns refer to the predicted quality.

4. Conclusion

We propose a novel framework for YouTube HTTP adaptive video streaming quality representation classification. Our solution was tested on both a dataset containing the Safari browser with Flash Player over HTTPS and the popular Chrome, Explorer, Firefox, and Safari browsers with a HTML5 player. On the Safari with Flash Player dataset, we achieved an average classification accuracy of 97% in the fixed mode and in the automatic quality representation switching mode. Using the quality classification ISP estimates the user buffer playout level with an average error of less than 4% (0.035 seconds). On the HTML5 browsers dataset our method accuracy was 77% and the errors were mainly between close quality representations. Our solution is more vulnerable to packet losses than to network delays.

The method presented in this paper is able to classify three different quality representations with reasonable accuracy. However, several factors can degrade performance. These include adding more quality representation layers or different frame rates. Changes in network conditions, as seen in the performance evaluation, and changes in the protocols can also reduce accuracy. Some of these changes can be resolved by online learning methods that continue to update their classification as new data arrive. Finally, quality representations and their changes are ingredients of user Quality of Experience (QoE). Clearly, other ingredients of QoE (e.g., content [55]) should also be classified to maximize ISP ability to shape DASH encrypted traffic with minimal reduction of user QoE.

The DASH encrypted traffic quality representation classification problem still faces many hurdles. The use of state-of-the-art network transport protocols such as HTTP2/SPDY and Quick UDP Internet Connections (QUIC) that have multiplexed connections should be investigated. TOR traffic morphing may also be a challenge to statistical classification. Adding features to strengthen robustness to packet losses is one of our future goals.

References

[1] Cisco, "The zettabyte era: Trends and analysis," 2015.

[2] ISO, "Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats," ISO/IEC 23009-1:2014, May 2014.

[3] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hofeld, and P. Tran-Gia. "A survey on quality of experience of http adaptive streaming," *IEEE Communication Surveys & Tutorials*, Vol. 17, No. 1, pp. 469-492, 2015. Article (CrossRef Link)

[4] C. M¨uller, S. Lederer, and C. Timmerer. "An evaluation of dynamic adaptive streaming over http in vehicular environments," in *Proc. of the 4th Workshop on Mobile Video*, pp. 37-42, NC, USA, Feb, 2012. Article (CrossRef Link)

[5] T. Hoßfeld, T. Zinner, P. Tran-Gia, and C. Timmerer, "Implementation and user-centric comparison of a novel adaptation logic for dash with SVC," in *Proc. of IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 1318-1323, Ghent, Belgium, May, 2013.

[6] C. M¨uller and C. Timmerer, "A VLC media player plugin enabling dynamic adaptive streaming over HTTP," in *Proc. of the 19th ACM international conference on Multimedia*, pp. 723--726, AZ, USA, Nov. 2011. Article (CrossRef Link)

[7] How-To Geek, Why YouTube in Chrome (and Firefox) is Draining Your Laptop's Battery and How to Fix It, 2016, Article (CrossRef Link).

[8] I. Ben Mustafa and T. Nadeem, "Dynamic traffic shaping technique for http adaptive video streaming using software defined networks", in *Proc. of Sensing, Communication, and Networking (SECON)*, pp. 178-180, Seattle, USA, June, 2015. Article (CrossRef Link)

[9] Google, "Google webmaster central blog: HTTPS as a ranking signal," 2014. Article (CrossRef Link)

[10] M. Belshe and R. Peon, "SPDY protocol," *Internet-Draft draft-mbelshe-httpbis-spdy-00, IETF*, Aug. 2012.

[11] M. Belshe, R. Peon and M. Thomson, "Hypertext Transfer Protocol Version 2," *RFC 7540, IETF*, 2015.

[12] V.K. Adhikari, S. Jain, and Z.L. Zhang, "YouTube traffic dynamics and its interplay with a Tier-1 ISP: An ISP perspective," in *Proc of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 431-443, NEW Delhi, India, Aug. 2010. Article (CrossRef Link)

[13] R. Torres, A. Finamore, J.R. Kim, M. Mellia, M.M. Munafo, and S. Rao, "Dissecting video server selection strategies in the YouTube CDN," in *Proc. of Distributed Computing Systems (ICDCS)*, 2011, pp. 248--257, Minnesota, USA, June, 2011. Article (CrossRef Link)

[14] A. Finamore, M. Mellia, M.M. Munafo, R. Torres, and S. Rao, "YouTube everywhere: Impact of device and infrastructure synergies on user experience," in *Proc. of the ACM SIGCOMM conference on Internet measurement conference*, pp. 345—360, ON, Canada, Aug. 2011. Article (CrossRef Link)

[15] T Hoßfeld, M Seufert, C Sieber, T Zinner, P Tran-Gia, "Identifying QoE optimal adaptation of HTTP adaptive streaming based on subjective studies," *Computer Networks* 81, 320-332. Article (CrossRef Link)

[16] J. Anorga, S. Arrizabalaga, B. Sedano, M. Alonso-Arce, and Jaizki Mendizabal, "Youtubes dash implementation analysis," in *Proc. of 19th International Conference on Circuits, Systems, Communications and Computers (CSCC)*, pp.61-66, Crete Island, Greece, July, 2015.

[17] G. Dimopoulos, "YouTube traffic monitoring and analysis," thesis, University of Catalonia

[18] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network - measurements, models, and implications," *Computer networks*, vol. 53, no. 4, pp. 501-514, 2009. Article (CrossRef Link)

[19] M. Cha, H. Kwak, P. Rodriguez, YY. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the worlds largest user generated content video system," in *Proc. of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 1-14, Kyoto, Japans, Aug., 2007. Article (CrossRef Link)

[20] X. Che, B. Ip, and L. Lin, "A survey of current YouTube video characteristics," *IEEE MultiMedia*, vol. 22, no. 2, pp.56-63, 2015. Article (CrossRef Link)
[21] S. Alcock and R. Nelson, "Application flow control in YouTube video streams," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 24-30, 2011. Article (CrossRef Link)
[22] A. Dainotti, A. Pescape, and K.C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35-40, 2012. Article (CrossRef Link)
[23] S. Valenti, D. Rossi, A. Dainotti, A. Pescap`e, A. Finamore, and M. Mellia, "Reviewing traffic classification," *Data Traffic Monitoring and Analysis*, pp. 123-147, Editors: Biersack, Ernst, Callegari, Christian, Matijasevic, Maja 2013. Article (CrossRef Link)
[24] Z. Cao, G. Xiong, Y. Zhao, Z. Li, and L. Guo, "A survey on encrypted traffic classification," in *Proc. of Int. Conf. on Applications and Techniques in Information Security*, pp. 73-81, Melbourne, Australia, Nov., 2014. Article (CrossRef Link)
[25] V. Paxson, "Empirically derived analytic models of wide-area TCP connections," *IEEE/ACM Transactions on Networking (TON)*, vol. 2, no. 4, pp. 316- 336, 1994. Article (CrossRef Link)
[26] R. Alshammari and A.N. Zincir-Heywood, "Unveiling Skype encrypted tunnels using gp," in *Proc. of IEEE Congress on Evolutionary Computation*, pp. 1- 8, Barcelona, Spain, July, 2010. Article (CrossRef Link)
[27] S. Zander, T. Nguyen, and G. Armitage, "Self-learning IP traffic classification based on statistical flow characteristics," in *Proc. of International Workshop on Passive and Active Network Measurement*, pp. 325- 328, MA, USA, Mar., 2005. Article (CrossRef Link)
[28] D. Zhang, C. Zheng, H. Zhang, and H. Yu, "Identification and analysis of skype peer-to-peer traffic," in *Proc. Of Internet and Web Applications and Services (ICIW)*, pp. 200-206, Barcelona, Spain, May, 2010. Article (CrossRef Link)
[29] I. Paredes-Oliva, I. Castell-Uroz, P. Barlet-Ros, X. Dimitropoulos, and J. Sole-Pareta, "Practical anomaly detection based on classifying frequent traffic patterns," *INFOCOM WKSHPS*, pp. 54-59, Orlando, FL, Mar., 2012. Article (CrossRef Link)
[30] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi, "Detailed analysis of Skype traffic," *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp.117- 127, 2009. Article (CrossRef Link)
[31] K.T. Chen, C.Y. Huang, P. Huang, and C.L. Lei, "Quantifying skype user satisfaction," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 399-410, 2006. Article (CrossRef Link)
[32] E. Hjelmvik and W. John, "Statistical protocol identification with SPID: Preliminary results," in *Proc. of Swedish National Computer Networking Workshop*, pp.399-410, Uppsala, Swedish, May, 2009.
[33] R. Bar-Yanai, M. Langberg, D. Peleg, and L. Roditty, "Realtime classification for encrypted traffic," in *Proc. of International Symposium on Experimental Algorithms*, pp. 373-385, Napoli, Italy, May, 2010. Article (CrossRef Link)
[34] A.M. White, A.R. Matthews, K.Z. Snow, and F. Monrose," Phonotactic reconstruction of encrypted VoIP conversations: Hookt on fon-iks," in *Proc. of IEEE Symposium on Security and Privacy*, pp. 3-18, California, USA, May, 2011. Article (CrossRef Link)
[35] C.V. Wright, L. Ballard, F. Monrose, and G.M. Masson, "Language identification of encrypted VOIP traffic: Alejandra y roberto or alice and bob?" *USENIX Security*, pp. 43-54, MA, USA, Aug., 2007.
[36] R. Dubin, O. Hadar, I. Richman, O. Trabelsi, A. Dvir, and O. Pele, "Video quality representation classification of Safari encrypted DASH streams," in *Proc. of Digital Media Industry \& Academic Forum (DMIAF)*, pp. 213-216, Santorini, Greece, June, 2016.
[37] P. Fu, L. Guo, G. Xiong, and J. Meng, "Classification research on SSL encrypted application," in *Proc. of International Conference on Trustworthy Computing and Services*, pp. 404-411, Beijing, China, Nov., 2013. Article (CrossRef Link)
[38] G. Lu Sun, Y. Xue, Y. Dong, D. Wang, and C. Li, "An novel hybrid method for effectively classifying encrypted traffic," in *Proc. of Global Telecommunications Conference (GLOBECOM)*, pp. 1-5, Florida, USA, Dec., Article (CrossRef Link)

[39] R. Dubin, O. Hadar, A. Noam, and R. Ohayon, "Progressive download video rate traffic shaping using TCP window and deep packet inspection," *WORLDCOMP*, pp. 10-17, Nevada, USA, July, 2012.
[40] P. Ameigeiras, J.J. Ramos-Muoz, J. Navarro-Ortiz, and J.M. Lpez-Soler, "Analysis and modelling of YouTube traffic," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 4, pp. 360-370, 2012. Article (CrossRef Link)
[41] A. Rao, A. Legout, Y.S. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proc. of the Seventh Conference on emerging Networking EXperiments and Technologies (CONEXT)*, pp. 1-12, Tokyo, Japan, Dec., 2011. Article (CrossRef Link)
[42] R. Dubin, A. Dvir, O. Pele, and O. Hadar, "I know what you saw last minute - the chrome browser case," *BlackHat*, London, Nov., 2016.
[43] The Crawler Code of the Codebook Algorithm and the Safari Dataset. Article (CrossRef Link)
[44] HTML5 Browsers Dataset and Video Information. Article (CrossRef Link)
[45] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. of Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027 -1035, Louisiana, USA, Jan., 2007.
[46] A. Shimoni and S. Barhom. "Malicious traffic detection using traffic fingerprint," Article (CrossRef Link)
[47] Selenium, Accessed: 2016-02-28. Article (CrossRef Link)
[48] J. Muehlstein, Y. Zion, M. Bahumi, I. Kirshenboim, R. Dubin, A. Dvir, O. Pele. "Analyzing HTTPS Encrypted Traffic to Identify User Operating System, Browser and Application," *CCNC* 2016. Article (CrossRef Link)
[49] Wireshark, Article (CrossRef Link)
[50] Clumsy network controller, version 0.2, Article (CrossRef Link)
[51] The HTML5 Crawler Code, Article (CrossRef Link)=0
[52] D. Rodriguez, Z. Wang, R. Rosa, and G. Bressan, "The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over HTTP," *EURASIP Journal on Wireless Communications and Networking*, vol. 216, no. 1, pp. 1-15, Dec. 2014. Article (CrossRef Link)
[53] C. Chen, L. Choi, G. de Veciana, C. Caramanis, R. Heath, and A. Bovik, "Modeling the time-varying subjective quality of HTTP video streams with rate adaptations," *IEEE Transactions on Image Processing*, vol. 23, no.5, pp. 2206-2221, May 2014. Article (CrossRef Link)
[54] R. Dubin, A. Dvir, O. Pele, O. Hadar, I. Katz, O. Mashiach, "Adaptation Logic for HTTP Dynamic Adaptive Streaming using Geo-Predictive Crowdsourcing," *Multimedia Systems*, pp. 1-13, Feb, 2016. Article (CrossRef Link)
[55] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. of Proceedings of the 14th international joint conference on Artificial intelligence (IJCAI)*, pp. 1137-1143, Quebec, Canada, Aug., 1995.
[56] J. Friedman, T. Hastie, and R. Tibshirani, "The elements of statistical learning," *Springer Series in Statistics*, Springer, 2001.
[57] D. C. Robinson, Y. Jutras, and V. Craciun, "Subjective video quality assessment of HTTP adaptive streaming technologies," *Bell Lab. Tech*. J., vol 16, Num 4, pp. 5-23, March 2012. Article (CrossRef Link)

Dr. Ran Dubin received the BSc, the MSc (cum laude) and the PhD degrees from the Ben Gurion University of the Negev, Israel, in 2010, 2013, and 2016, respectively, all in communication system engineering. Ran was assigned by the Israeli district courts to consult on the subject of Internet network class claims issues. Ran's previous industry experience was developing machine learning and big-data algorithms in the cyber protection and network communication fields

Prof. Ofer Hadar received the BSc, the MSc (cum laude) and the PhD degrees from the Ben Gurion University of the Negev, Israel, in 1990, 1992, and 1997, respectively, all in electrical and computer engineering. From August 1996 to February 1997, he was with CREOL at Central Florida University, Orlando, FL, as a visiting research scientist. From October 1997 to March 1999, he was a post - Science at the Technion - Israel Institute of Technology, Haifa. In 1999, Prof. Hadar joined the Communication Systems Engineering Department at Ben - Negev. Currently, he is Associate Professor and the head of the department. His research interests include image compression, advanced video coding, H.264, SVC, HECV, packet video, transmission of video over wireless networks, and image processing, data hiding and cyber in compressed video streaming. Since 2011, Prof. Hadar is an Associate Editor of the Optical Engineering journal. He was the guest editor (with his former Ph.D student Dr. Dan Grois) of a special section on video compression technology in Opt. Eng. 52(7), (July, 2013). On August 2015 Prof. Hadar started a Sabbatical leave for six months at the department of Electrical Engineering at UCLA, working with Prof. Rubin on optimizing video streaming over wireless networks. Prof. Hadar also works as a consultant for various Hi Tech companies in Israel, and is a Senior member of IEEE and SPIE. Recently Prof. Hadar established a startup company Coucou that focuses on developing algorithms against cyber-attack in video streaming.

Dr. Amit Dvir received the B.Sc., M.Sc., and Ph.D. degrees from Ben-Gurion University, Beer Sheva, Israel, all in communication systems engineering. He
is currently a Faculty Member with the Computer Science Department and the Cyber Center, Ariel University, Israel. From 2011 to 2012, he was a Post-Doctoral Fellow with the Laboratory of Cryptography and System Security, Budapest, Hungary.
His research interests include enrichment data from encrypted traffic.

Dr. Ofir Pele received the B.Sc. degree in computer science and life science and the M.Sc. and Ph.D. degrees in computer science from the Hebrew University of Jerusalem, Israel, in 2003, 2005, and 2011, respectively. From 2011 to 2013, he was a PostDoctoral Fellow with the Department of Computer and Information Science, University of Pennsylvania, working with the late Prof. B. Taskar. In 2013, he joined Ariel University as a Lecturer (Assistant Professor). His work focuses on novel algorithms that combine practical applicability and theoretical insights. He strongly believes in reproducible research and publishing open source code of his work. His work has been successfully used by several research groups, including groups from prestigious universities and research centers. His research interests include machine learning, computer vision, and their applications.