Unveiling the end-user viewport resolution from encrypted video traces

Othmane Belmoukadam, Chadi Barakat

Unveiling the end-user viewport resolution from encrypted video traces

Othmane Belmoukadam
Université Côte d'Azur, Inria, France
othmane.belmoukadam@inria.fr

Chadi Barakat
Université Côte d'Azur, Inria, France
chadi.barakat@inria.fr

***Abstract*—Video streaming is without doubt the most requested Internet service, and main source of pressure on the Internet infrastructure. At the same time, users are no longer satisfied by the Internet's best effort service, instead, they expect a seamless service of high quality from the side of the network. As result, Internet Service Providers (ISP) engineer their traffic so as to improve their end-users' experience and avoid economic losses. Content providers from their side, and to enforce customers privacy, have shifted towards end-to-end encryption (e.g., TLS/SSL). Video streaming relies on the dynamic adaptive streaming over HTTP protocol (DASH) which takes into consideration the underlying network conditions (e.g., delay, loss rate, and throughput) and the viewport capacity (e.g., screen resolution) to improve the experience of the end user in the limit of the available network resources. In this work, we propose an experimental framework able to infer fine-grained video flow information such as chunk sizes from encrypted YouTube video traces. We also present a novel technique to separate video and audio chunks from encrypted traces based on Gaussian Mixture Models (GMM). Then, we leverage our dataset to train models able to predict the class of viewport (either *SD* or *HD*) per video session with an average 92% accuracy and 85% F1-score. The prediction of the exact viewport resolution is also possible but shows a lower accuracy than the viewport class.**

***Index Terms*—Video streaming, controlled experiments, video chunk size, viewport resolution, YouTube encrypted traces, machine learning.**

I. Introduction

The Internet as we know it today, has revolutionized people's life, from shopping to ordering food, sharing moments with family and friends, ending with instant messaging. Everything is one click away from anybody anywhere on the globe. Video traffic is unarguably the major contributor to the global Internet traffic and the main source of pressure on the Internet infrastructure. By 2023, video traffic is expected to account for 73% of the global mobile data traffic [1]. Moreover, due to the COVID-19 pandemic, lockdown forced people around the globe to restrict their mobility and increase their video traffic, for remote working, entertainment, and education. Recently, researchers have leveraged data from ISPs, IXPs and educational networks, to show that video traffic has increased by 15-20% almost within a week [2]. A statement by the European Union raised concerns about the Coronavirus lockdown putting strain on broadband delivery systems. As result, mainstream content video providers, such as Netflix, reduced their video resolution to the standard definition during the pandemic [3]. Later, some providers started again to upgrade their services back to high definition or 4K around [4].

Meanwhile, in light of the rapid growth of video traffic, Internet Service Providers (ISPs) feel more pressure to optimize their networks and meet the expectations of their end users. They give high importance to video traffic engineering which requires the ability to infer the context of the video streaming such as the characteristics of the terminal on which the video is played out and the resolution of the streamed video. However, this is getting more difficult because of the end-to-end encryption adopted by major video streaming platforms (e.g., YouTube, Netflix and Amazon) [5]. For example, to prioritize or load balance video traffic efficiently, ISPs need information on end-users' Quality of Experience (QoE) rather than just capturing the network Quality of Service (QoS). But, video QoE is dependent on the content itself (the video bitrate and resolution) and on the application-level QoS metrics such as start-up delay, duration of stalls and resolution switches [6]–[8]. It also depends on the resolution of the viewport on which the video is played out [9]. All this information is unfortunately hard to obtain from encrypted video traffic, making its inference an important challenge to surmount.

In this paper, we present a data-driven methodology unveiling the end user viewport resolution from YouTube encrypted video traces. To that aim, we leverage video chunk sizes and inband network-level traffic features such as throughput and download/upload packet inter-arrival times to train machine learning models able to distinguish between *HD* and *SD* viewports and infer the resolution of the viewport. More specifically, our contributions are the following:

- We present a controlled experimental framework to perform video streaming experiments at large scale and collect YouTube video metadata. We leverage the Chrome Web Request API to read the clear HTTP text messages [10] and obtain ground truth on the video streams and the dynamics of their chunks.
- We provide a descriptive analysis of a large YouTube video catalog, highlighting the diversity of its video content, with a focus on the encoding bitrate w.r.t. video category and format (i.e., mp4 versus webm).

[1]This work is supported by the French National Research Agency undergrant BottleNet no. ANR-15-CE25-0013 and by Inria within the Project LabBetterNet.

[2]An earlier version of this work appeared in the proceedings of the 16-th International Conference on Network and Service Management CNSM, 2-6 November 2020.

- We stream up to 5000 unique YouTube videos, collect encrypted traces and clear HTTP messages, and show that chunk sizes and inband network-level traffic features carry an interesting signature of the viewport resolution.
- We propose a novel approach to separate video and audio chunks from encrypted video traces based on a Gaussian Mixture Model (GMM). Then, we validate our work on inferring video chunk sizes by comparing similarities and differences with respect to the real video chunk size distribution derived from the clear HTTP messages.
- We train different machine learning algorithms to classify the viewport resolution. We prove the pertinence of this classification taking as input video chunk statistics and inband network-level traffic features that can be derived from passive captures of encrypted video traffic.

Overall, the paper is organized as follows. In the second section, we discuss video streaming development throughout the years and motivate our study. Later, in the third section, we summarize the mainstream contributions related to deep packet inspection, video QoE modeling and inference from encrypted traffic. In Section IV, we present our experimental framework, followed by a descriptive study of today's video content based on an open source YouTube catalog. In Section V, we present our methodology to extract video chunk sizes from YouTube encrypted traces. Later, in Sections VI and VII, we highlight the viewport signature carried by a set of inband network traffic features and chunk size statistics, and train and evaluate a classifier able to classify the viewport resolution from encrypted traces. Last, we conclude and present our future work in Section VIII.

II. BACKGROUND AND MOTIVATION

Video transmission over HTTP has changed over the years. It started with videos downloaded completely by the clients before being played out, to videos progressively streamed to the clients at a fixed resolution. Recently, adaptive streaming over HTTP has been widely adopted to automatically tune the streamed video resolution as a function of the available network resources. For instance, Dynamic Adaptive Streaming over HTTP (DASH) protocol [11], [12] allows adapting the video quality to the available bandwidth and the client terminal characteristics (e.g., viewport resolution). In plain, DASH divides the video into segments (e.g., 2 - 10 seconds) with each segment available in different quality versions. Details on the video availability at the server are stored in the Media Presentation Description (MPD) template which describes the video segments in terms of coding standard and bitrate and is shared with the client at the beginning of every video session. The choice between the different video representations is done by the DASH client taking into consideration the network state and the terminal capacity with as objective the smoothest possible playout without excess bandwidth usage.

In parallel, previous studies highlight key video QoE metrics such as stalls, average resolution, and quality fluctuations from measurements of the encrypted traffic and by using machine learning [13]. As a matter of fact, statistics show that up to 90% of end-users abandon their operator after experiencing network quality degradation without giving any feedback, resulting in huge economic losses. Moreover, users watching videos start abandoning the session after 2 seconds of join time, while 80% of them leave the session when the join time exceeds 60 seconds. As result, both operators and service providers feel more pressure to be proactive and enhance their services as much as possible by assessing the end-user QoE.

Therefore, when it comes to video streaming and user experience, studies mainly rely on the fact that application-level metrics are proved to be tightly correlated to the network-level QoS [14]–[16]. However, they overlook an important information regarding the resolution of the viewport on which the video is played out and its impact on the user QoE. Cermak et al. [9] answered partially the question concerning the bandwidth needs for acceptable video experience on a set of screen resolutions. Authors in [17] argue that any representation exceeding the resolution of the viewport brings the maximum level of QoE. Moreover, in a previous work [18], we also observed that the DASH transmission process may result in download resolutions exceeding the screen resolution (i.e., a viewport in full-screen mode), hence resulting in a waste of bandwidth. Such waste can be of particular concern to end-users paying their subscription at the byte level and to operators who can invest it on other flows in need of it. In fact, the literature is missing a solution to infer the viewport resolution from passive captures of encrypted video traffic, which is the main focus of our paper.

III. STATE OF THE ART

When assessing the user experience, also known as QoE, different approaches exist in the literature to target specific services or aspects. For instance, authors in [19] describe YoMoApp, a video streaming crowdsourcing application, that aims at the collection of Youtube's QoS metrics and users' feedback. Besides, Meteor [20] links network measurements to multiple services' QoE, while RTR-NetTest [21] provides user-friendly meters per each network QoS metric. Researchers in [22] address QoE from the hardware perspective, paying more attention to performance issues and their impact on QoE. Meanwhile, social media applications also embed QoE feedback related to network conditions (e.g., Messenger, Skype)

For video QoE in particular, researchers aim at linking the QoE to application-level Quality of Service (QoS) measurements, such as the initial join time (time for video to start playing out), the number of video interruptions (stalls) and the playout quality variations (resolution switches) [14]. Meanwhile, recent studies hinted at a strong correlation between the network-level and application-level QoS metrics opening up the door for a stream of approaches linking the network conditions (e.g., throughput, delay and jitter) directly to the QoE. For instance, authors in [16] perform QoE forecasting using machine learning models taking as input network-level QoS metrics (e.g., throughput and delay) collected by end-users' devices with the help of active measurements. The same trend is depicted in [23], [24] with mobile applications

inferring the QoE from passive measurements performed on the end-users' devices. Aggarwal et al. [25] use TCP-level flow features collected from a mobile core network to produce models able to detect video stalls (interruptions) for adaptive video streaming over HTTP. The Mean Opinion Score (MOS) is also used to estimate the subjective measure of user satisfaction, while modern standards (e.g., ITU) focus on protocols or general class of applications without being specific to any Internet service [26], [27].

On another related topic, traffic identification from encrypted traces is an active field of study. Methods based on Deep Packet Inspection (DPI) offer solutions to inspect and take actions based on the payload of the packets rather than just the packet header. Machine Learning (ML) is widely exploited in the DPI field, as multiple ML based solutions have been proposed over the last years [28]–[30]. ML algorithms proved their efficiency, learning from big data and statistical properties of the traffic flow. However, these algorithms pass by a heavy training phase and might struggle in terms of processing complexity if run in real-time. Another well-known DPI technology is OpenDPI, which is freely available and includes the latest DPI technology combined with other techniques making it one of the most accurate classifiers nowadays [31]. Khalife et al. [32] attempt to reduce the OpenDPI computational overhead by examining different sampling techniques. Two sampling techniques are proposed and compared *(i)* per-packet payload sampling, and *(ii)* per-flow packet sampling. Enhancing DPI performance is as active as inventing new DPI technologies, so several approaches have been proposed including behavioural [33], statistical [34], port based [35] and DFI (Deep Flow Identification) based approaches [36]. Other approaches apply either software based optimization focused on enhancing DPI algorithms, e.g., [37], or rely on hardware based optimisation [38].

In this context of end-to-end encryption, and knowing the reality behind video QoE, researchers leverage encrypted traffic by passively monitoring the network and capturing traffic statistics that are then transformed onto video QoE indicators using machine learning. For instance, one can find work on inferring video interruptions, video quality and quality variations by observing network-level traffic statistics [13]. Others use a large number of video clips to identify specific Netflix videos leveraging only the information provided by the TCP/IP headers [39]. Dimopoulos et al. [13] propose to use the size of video chunks as input to machine learning, however, their method requires access to the end-user device to collect real values about these chunks, instead of inferring them from the encrypted packet traces. They also provide the first heuristic to automatically extract chunk size information from encrypted traffic based on identifying long inactivity periods along the video streaming session. Silhouette [40], a video classification method, uses Application Data Units (ADUs) and network statistics to detect and infer properties about video flows. Their method leverages downlink/uplink packet characteristics to identify chunk requests and corresponding information sent by the server (chunk size), however, it only incorporates static thresholds making it unable to differentiate between video and audio chunks.

Previous studies incorporate techniques targeting video flow identification and the inference of application-level quality of service. In the light of advanced and diverse equipment, their limitation is in overlooking the impact of the end-user display which is an important factor to determine the end-user QoE. In a previous contribution [41], we take into consideration the relationship that exists between screen resolution, video resolution and end-user QoE, and propose a resource allocation problem that maximizes the overall QoE over a set of users sharing the same bottleneck link. In the present work, we complete the puzzle by proposing techniques to infer the end-user viewport characteristics from the encrypted traffic, which together with the information on the video flow such as the streaming resolution and the application level QoS, can provide the ISP with a fine-grained estimation of the user QoE for a more efficient video traffic management. To the best of our knowledge, this is the first attempt to infer the end-user viewport resolution and viewport class (*SD* or *HD*) using inband network traffic features and chunk sizes inferred from encrypted video traces. Further, our approach to separate video and audio chunks from encrypted traces is novel and proves its efficiency when compared to ground truth collected from clear HTTP messages through the Chrome Web API [10].

IV. Experimental setup

We play different YouTube videos using different viewports, and under different network conditions emulated using Linux traffic control (*tc*). Each experiment consists of a unique YouTube video, browser viewport, and enforced network bandwidth. For every video session, we leverage the Chrome Web Request API to read the clear HTTP text messages and establish ground truth on the requested chunks and the application-level quality of service. Moreover, we dump the encrypted client-server traffic to pcap files using *tcpdump*.

A. Overall experimental framework

Our overall experimental setup, described in Fig. 1, consists of a local *mainController* running on MacBook Air machine of 8 GB RAM. Videos are visualized on a Dell screen 27' of 2560 x 1440 resolution. The local *mainController* stores the YouTube video catalog and the viewport list, and provides a random combination of video ID and viewport for every new experiment as illustrated in Fig. 1. We consider a list of default standard viewports such as the current YouTube small media player mode (400x225) along with other default *SD* viewports (e.g., 240x144, 640x360 and 850x480). These latter viewports represent the current player dimensions adopted by streaming platforms for several watching modes. We also account for larger viewports by considering the standard 1280x720 and 1920x1080 (*HD*). In fact, as of march 2021, stats show that up to 70% of desktop screens worldwide are of resolution less than or equal to 1920x1080 [42]. Since video resolution pattern is a function of both network conditions and terminal display capacity, we study the viewport importance while

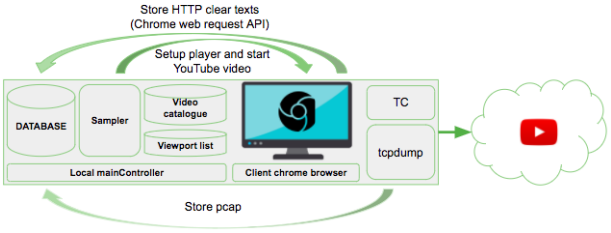
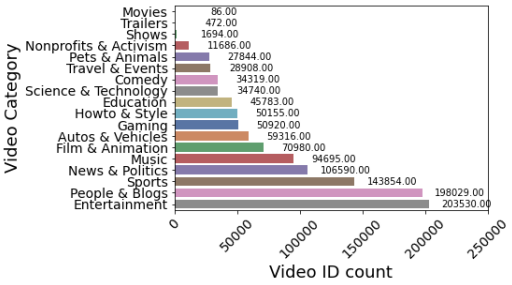
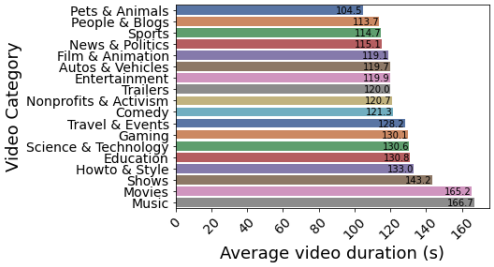


Fig. 1: Experimental framework description



(a) Total number of unique videos



(b) Average video duration

Fig. 2: YouTube catalogue overview per video category

degrading the network bandwidth. To that aim, we use Linux traffic control *tc* and enforce different bandwidth settings such as 3, 6, 9, 15 and 20 Mbps. We also stream with no bandwidth limitation on Ethernet to emulate the best case scenario.

B. YouTube catalogue

We use the open source YouTube catalogue proposed in [43] to identify the videos to stream. The catalog was built using the YouTube API where YouTube was searched with specific keywords obtained from the Google Top Trends website. Later, the authors in [44] rely on Google's *getvideoinfo* API to return the video metadata for each video identifier in the catalog. The catalogue includes around 1 Million unique video identifiers, each of which is characterized by an encoding bitrate that differs from one video to another depending on its content (e.g., high motion, slow motion, static, music video). Overall, 99% of the videos featured by the YouTube catalog support resolutions up to 1080p, the remaining 1% support higher resolutions (e.g., 2160p and 2880p). Due to the small proportion of the latter resolutions, we limit our study to videos available in multiple resolutions up to 1080p.

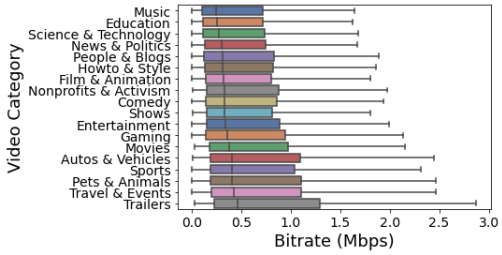
1) Catalog overview: In Fig. 2(a), we highlight the cumulative unique videos per category. The Entertainment and People & Blogs categories are the largest ones with 203,530 and 198,029 unique videos, respectively, followed by the sports category with almost 150K unique videos. On the other hand, the least represented categories in the catalogue are Movies and Trailers with 86 and 472 unique videos, respectively. In terms of average video duration (Fig. 2(b)), the Music videos are the longest with an average duration of 166 seconds, followed by videos classified by YouTube as Movies with an average of 165 seconds (only 86 unique videos). At the bottom of the list, we find the Pets & Animals and People & Blogs categories with 104 and 113 seconds, respectively.

2) Encoding bitrate: We now zoom in on the encoding bitrate per category and coding standard. In Fig. 3(a), we plot the distribution of encoding bitrate per video category using box-plots ranked by median value. As illustrated, the Trailers category is the one highlighting the highest median encoding bitrate up to 0.4 Mbps. The Trailers category is further characterized by the largest span in the video bitrate. This makes sense as movie trailers feature the latest video resolution technology for attracting more viewers. In a second place, one can find Travel & Events videos with a median encoding bitrate slightly less than the first category. At the bottom, we find the Music, Education and Science & Technology videos with almost the same bitrate distribution and a median around 0.2 Mbps.

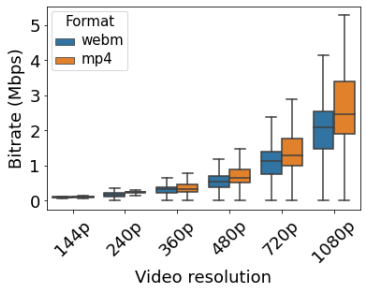
Next, we study the encoding standard and its implication on the bitrate. A video encoding standard is a description of a bit stream structure and a decoding method to enable video compression. It highlights a set of tools for compression and defines the output structure that an encoder should produce. In this scope, we study the YouTube catalog so as to understand the video formats and therefore the encoding standards that are supported by YouTube. Overall, YouTube videos are available in two major video formats encoded by the H.264 and Google's VP9 standards. The individual video formats are "mp4" and "webm", respectively. In our YouTube catalogue, 82% of the videos are available in both mp4 and webm. The remaining 18% of videos are only available in mp4.

We plot in Fig. 3(b) the bitrate distribution for the main video resolutions, ranging from 144p to 1080p, and for both video formats. The two formats have slightly different bitrates for the same resolution with a noticeable advantage for the "webm" format on the "mp4" one, making it the preferred format by Google to handle the video content bulk. In plain, for the highest video resolution of 1080p we consider, the "mp4" format is characterized by a median bitrate of 2.6 Mbps compared to 2.2 Mbps for the "webm" videos and a 75th percentile of 3.3 Mbps compared to 2.5 Mbps. The figure also highlights a clear positive correlation between the video resolution and the bitrate, as expected.

Our observations on the bitrate are in line with the prior study in [45] on the correlation between the two formats for encoding multimedia content and the user experience. Indeed, the authors in [45] compare the two formats from the



(a) Video bitrate per category



(b) Bitrate vs Resolution w.r.t. video format

Fig. 3: YouTube video bitrate insights

perspective of the Mean Opinion Score (MOS), highlighting an advantage of the H.264 (mp4) when the network conditions are favorable, while the VP8 codec (webm) behaves better in highly error-prone networks.

V. Analysis of Video Streaming Traffic

In adaptive video streaming, the client decides on the resolution of the next chunk to download based on underlying network conditions and viewport characteristics. So each viewport is supposed to exhibit a different video resolution pattern depending on the available network resources; the pattern of chunk sizes is the main illustration of such specific behavior. However, as most of the video traffic is encrypted, the information on the viewport resolution is not visible to any entity between the client and the server. Our intuition is to exploit the specificity of the chunk size pattern to infer the viewport resolution from encrypted traffic. The problem is that when the bandwidth starts getting scarce either due to congestion or to in-network shaping, clients are automatically forced by DASH to request lower video resolutions, thus reducing the effect of the viewport and increasing the difficulty to infer its resolution. To highlight these aspects, we therefore investigate the extent to which screens impact the video transmission pattern while varying the network bandwidth.

A. Inferring video chunk sizes

Overall, we stream up to 5K YouTube unique videos in series randomly selected from the 1 Million catalogue in [43]. In general, chunks of a video are fetched using separate HTTP requests. So on one hand, we infer the chunk sizes from the encrypted YouTube traces. In parallel, we extract the real chunk sizes from the clear text HTTP messages accessed from within the Chrome browser using the Chrome Web Request API [10]. Our chunk size inference method is inspired by [40], [44] and works as follows. At first, we use the source IP of our host and the list of destination IPs to isolate the different flows corresponding to the video sessions (CDNs identified by the URLs ending with googlevideo.com). The CDN identifiers can be collected from the clear HTTP text messages and their corresponding IP addresses can be resolved. Then, for each video streaming session (source and destination already known), we look at the size of uplink packets. Large sized uplink packets correspond to chunk requests while small packets correspond to transport level acknowledgments by TCP/QUIC. Instead of using thresholds as depicted in [29], we use K-means clustering to segregate the uplink packet sizes into two clusters; the first cluster represents the request packets and the second cluster represents the acknowledgment packets. Once the uplink request packets are identified, we sum up the data downloaded between any two consecutive request packets, and consider it equal to the downloaded chunk size following the first request between the two. Overall, we leverage clustering for the sake of generality and to make sure our approach can be reused with other type of ACKs mainly in the context of other transport protocols.

Up to this point, and as the case for other existing methodologies, the calculated chunk sizes mix between audio and video chunks, whereas we are only interested in video chunks. In fact, the chunks located between consecutive request packets can be of either of the two types, audio or video, and so need to be separated. To overcome this limitation, we leverage Gaussian Mixture Models (GMM) applied to the chunk size. The GMM clustering method is based on the maximum likelihood principle, able to find clusters of points in a dataset that share some common characteristics. Unlike K-means, GMM belongs to the soft clustering subset of unsupervised algorithms, it provides probabilities that tell how much a data point is associated with a specific cluster. Another key property of GMM is that clusters do not need to be topologically separated as with K-means, they can overlap and still be identified as long as they follow some Gaussian property for the distribution of their points. In general, video chunks should have larger sizes than audio chunks, we rely on this property to identify the two Gaussian distributions and classify the chunks between audio and video. Each distribution has three unique values, mean γ, covariance $\sum$ modeling the spread around the mean, and a probability π defining how big or small one cluster is compared to the other one, the sum of probabilities of the two clusters is naturally equal to 1.

For our case, we fit a GMM of two components with the chunk sizes inferred according to K-means. For a clear visual illustration, we plot in Fig. 4 the two clusters rendered by the GMM method, over a 2D space of chunk sizes (MB) and download time (s). In plain, the audio cluster (in blue) shows chunks smaller than 750 KBytes with no more than 2 seconds of download time. Video chunks however (in orange) can be of larger sizes and download times compared to audio ones.

Now we test the accuracy of our method by comparing its

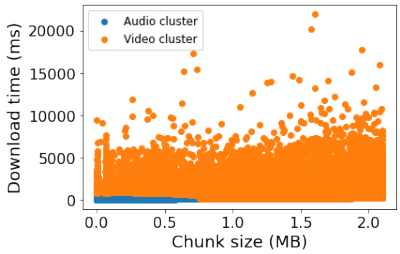


Fig. 4: Audio/video clusters as produced by GMM

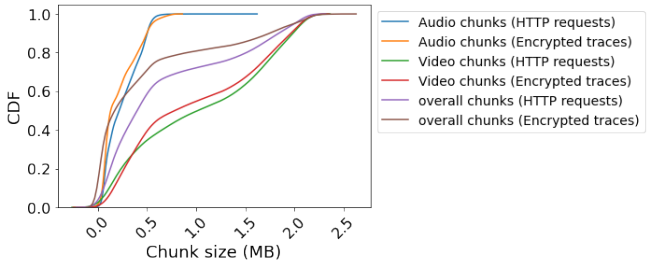


Fig. 5: Chunk size CDF

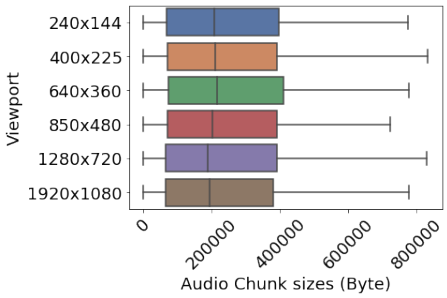


Fig. 6: Audio bitrate distribution

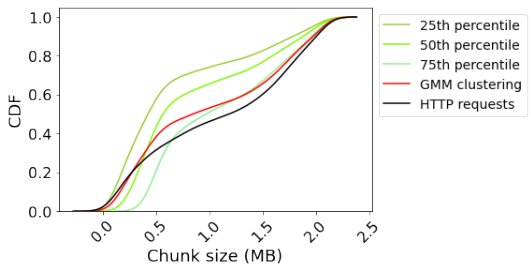


Fig. 7: Threshold/GMM/HTTP inference of video chunks

output to the ground truth collected directly from within the browser by analyzing the clear HTTP requests. These requests include the *itag*, *range* and *mime* (Multi-purpose Internet Mail Extensions) parameters, which can be then used to infer the corresponding resolution, the codec and the size of the chunk using open source documentation [46], [47]. We use this ground truth to check whether our GMM method provides video chunk sizes that respect the distribution of the size of real video chunks as seen in the browser. Fig. 5 compares the chunk sizes as estimated by our method from the encrypted traffic traces and the chunk sizes obtained from the clear text HTTP traces for the same video sessions. The overall distribution of the encrypted chunk sizes extracted using our method exhibits the same shape as the one obtained with HTTP requests. Further, the two distributions produced by our method for audio and video chunks are very close to those of the HTTP requests. We can also notice how the video chunks, understandably, have larger sizes than the audio chunks. This helps better characterizing the video chunks within a trace of encrypted video traffic, with this result particularly useful in our case to further understand the interplay between viewport, network resources and chunk resolution pattern.

B. Audio chunk size distribution

We illustrate in Fig. 6 the audio chunk sizes accessed from within the Chrome browser using the Chrome Web Request API [10] w.r.t. the viewport size considered in the experiments. Overall, we notice that regardless of the viewport size, the audio chunk size distribution is almost the same, which discards any impact of the viewport and confirm the use of a standard audio quality. In plain, the audio chunk size distribution is characterized by a median encoding bitrate of 200 Kbytes. Moreover, the variation of the audio chunk size is almost the same through all viewports, with the 25th percentile and 75th percentile equal to 100 and 400 Kbytes, respectively.

C. Threshold based audio/video chunk separation

Above, we leveraged the GMM clustering to separate audio and video chunks from each other. Another feasible solution easier to deploy would be to use static thresholds applied to chunk size. Here, we compare, clustering and threshold based techniques for the sake of chunk segregation efficiency.

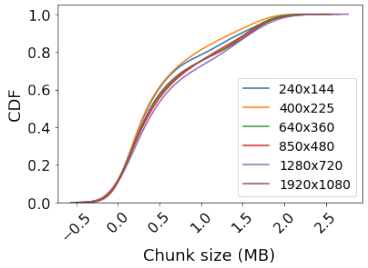
We leverage the audio chunk size distribution illustrated in Fig. 6 to derive threshold values able to separate the two types of chunks based on their sizes. In plain, we use three threshold values representing the 25th, 50th and 75th percentiles of audio chunk sizes. For each threshold, the set of audio chunks include every chunk with a size less than the threshold while the others are considered to be video chunks. In Fig. 7, we plot the video chunk size distribution per different separation methods; *(i)* the three variants of the threshold based method, *(ii)* the video chunk sizes inferred using the GMM clustering, *(iii)* and the real video chunk size distribution as inferred from the HTTP requests. We can observe that the overall distribution of the video chunk sizes is well captured by both the threshold based and the clustering based methods, with as expected the higher the threshold the more the shift of the distribution towards larger video chunks. In plain, the 75th percentile threshold provides the closest distribution to the real one, yet, the GMM method by using the maximum likelihood principle is able to capture the real video chunk size distribution in a close manner. To note here that a main advantage of the GMM clustering method is in its automatic learning property, which avoids one from tuning manually the threshold value.

D. Video resolution pattern

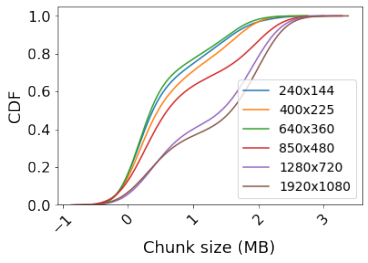
The DASH client automatically switches between video resolutions according to the viewport and underlying network performance. The video resolution pattern as requested from the server is thus determined by the network conditions, and normally has to take into consideration the viewport size, which is defined as the number of pixels, both vertically and horizontally, on which the video is displayed. It is indisputable that the network conditions, for instance the bandwidth, reduces the impact of the screen in scenarios of bandwidth shortage as DASH will end up downloading chunks of lower resolution than the viewport capacity. In this section, we present experimental results supporting these statements and highlight in particular the reduction of the effect of the viewport as the available bandwidth decreases.

We artificially change the available bandwidth (as highlighted in Sec. IV), and stream for each bandwidth setting hundreds of YouTube videos using different viewports. Each time, we use random sampling to select the video ID and the viewport. We plot in Fig. 8 the CDF of the video chunk size per viewport for three bandwidth settings: 3 Mbps, 15 Mbps, and no control. As expected, the video resolution pattern is driven by the network bandwidth and the viewport size. In Fig. 8(a), the bandwidth is limited to 3 Mbps, all viewports thus exhibit the same pattern by streaming the same video resolution, which therefore results in the same cumulative distribution of chunk sizes. However in Fig. 8(b), we set the bandwidth to 15Mbps, the effect of the viewport starts appearing as the distribution of chunk sizes differs from one screen to another. However, and even at this high bandwidth, the two large viewports 1280x720 and 1920x1080 illustrate close distributions, which can be explained by the same reason of bandwidth shortage. Finally, when no restriction is imposed on the bandwidth, full high definition viewport (1920x1080) starts differentiating itself from the others. We further notice in Fig. 8(c) that 40% of chunk requests on small screens (e.g., 240x144, 400x225 and 640x360) correspond to a chunk size smaller than 200 KBytes compared to 300 KBytes for medium screens. For 1280x720 viewports (HD), chunk sizes are bigger with 40% of them smaller than 1 MBytes (resp. smaller than 1.6 MBytes for 1920x1080 Full HD viewports).

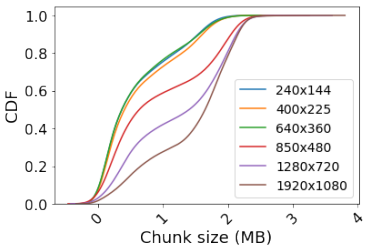
To illustrate further this result, we plot the network throughput as measured over the encrypted traces and compare it to the available bandwidth for different viewports. For each video, we get the CDN URL from the HTTP logs and use the DNS Lookup of CDN URL to identify the video flow corresponding to using the CDN IP. Then, we leverage the downlink packet timestamps and a time bin of 1s to return a vector of throughput values per video session. The vector is used to derive throughput statistics (e.g., average, percentiles) per video session. In Fig. 9, we plot the CDF of the throughput values of the video sessions for different viewports with different bandwidth settings. We notice that with an enforced bandwidth of 3 Mbps (Fig. 9(a)), all viewports end up experiencing the same throughput which correlates with chunk



(a) Chunk size per viewport with a 3Mbps bandwidth



(b) Chunk size per viewport with a 15Mbps bandwidth



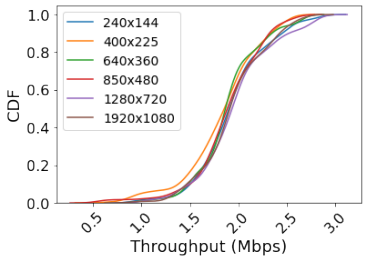
(c) Chunk sizes per viewport with unlimited bandwidth

Fig. 8: Network and viewport impact on chunk sizes

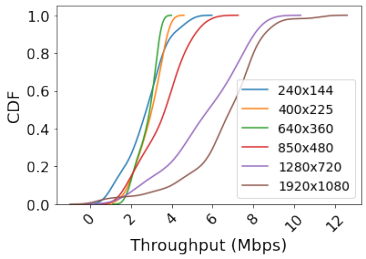
size results. Moreover, regardless of the available bandwidth, a subset of viewports form one cluster exhibiting the same throughput pattern (e.g., 240x144, 400x225 and 640x360).

VI. Traffic correlation to viewport

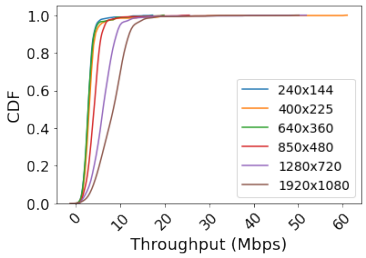
For each video session, we get an array of video chunk sizes over which we calculate different statistical features that we use for viewport classification. We study here the correlation between this array and the viewport. Our feature set contains the maximum, the average and the standard deviation along with the $10th$ to $90th$ percentiles (in steps of 10) of the chunk size array. This forms a set of 12 features describing statistically the evolution of chunk sizes over a video session. In addition to chunk size related statistics, we also consider the same statistical features, but this time for the downlink throughput (in bps, averaged over time bins of 1s), and the uplink and downlink packet interarrival times (in seconds). With these features, we believe that we get a fine-grained description of the DASH transmission process and capture any effect of viewport resolution. Overall, according to feature analysis, viewports such as 240x144, 640x360 and 850x480



(a) Throughput per viewport for a 3Mbps bandwidth



(b) Throughput per viewport for a 15Mbps bandwidth



(c) Throughput per viewport for a unlimited bandwidth

Fig. 9: Throughput per viewport for multiple network settings

are more likely to exhibit close chunk size and throughput distributions forming one viewport class (*SD*). On the other hand, the 1280x720 and 1920x1080 represent another cluster, called *HD* showing similar properties. To take advantage of this overlapping, we equally consider a relaxed definition of the viewport classification problem to either *SD* or *HD*.

Before building our classifier, we start by illustrating the correlation between our feature set and the viewport class. Fig. 10 points to most relevant features by ranking them according to their Pearson correlation coefficient with the viewport class. The figure shows only those features having a correlation coefficient at least equal to 0.4. The x_th_csize represents the x_th percentile of the video chunk size over a video session and the y_th_dltp stands for the y_th percentile of the downlink throughput. Overall, the chunk size percentiles show a more important correlation with the viewport capacity, especially when it comes to lower percentiles. This is because the video resolution pattern is not only influenced by the available network resources, but also by the user display capacity. Downlink throughput percentiles come in second place with a correlation coefficient of more than 0.4.

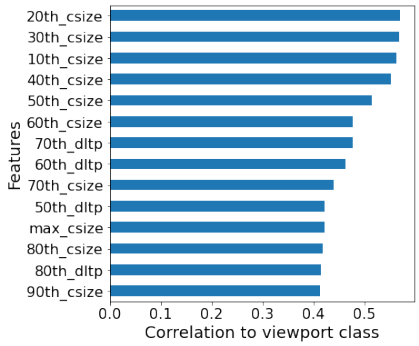


Fig. 10: Features correlation to viewport class, *SD* (0), *HD* (1)

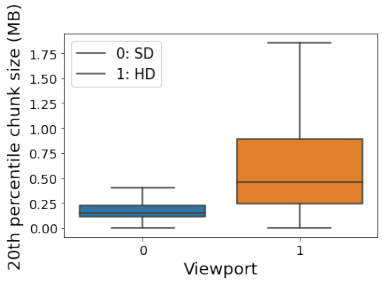


Fig. 11: $20th$ chunk size percentile (most relevant chunk size percentile in Fig. 10), *SD* (0), *HD* (1)

To shed further light on the previous results, we show boxplots of the most important traffic features w.r.t. the two viewport classes. We plot in Fig. 11 the distribution of the $20th$ chunk size percentile for all video sessions and for both viewport classes. Overall, we notice a small overlapping portion, the smaller the overlap the easier to differentiate between *SD* and *HD* viewports. In plain, 50% of video sessions have their $20th$ chunk size percentile less than or equal to 230 KBytes, whereas high definition viewports score almost twice the value for the same percentile. In terms of downlink throughput, we plot in Fig. 12 the distribution of the $70th$ download throughput percentile for all video sessions as it scores 0.46 in terms of the correlation coefficient. In general, and as expected, larger screens are characterized by larger throughput values. Moreover, the boxplots show that half of our video sessions have a $70th$ download throughput percentile around 7 Mbps compared to 4 Mbps for small definition viewports. All these results point to the presence of a correlation pattern between encrypted traffic features and viewport resolution at the client, a pattern that we exploit next to build our classifier of viewport resolution.

VII. Viewport Classification by Machine Learning

In this section, we discuss the performance of the ML model built using our dataset. We start by predicting the viewport class (*SD* or *HD*) using inband network features and chunk size stats ($F_{inband+chunk}$) extracted from the YouTube encrypted traces. Later, we highlight the performance of our methodology in the context of multi-label classification where the

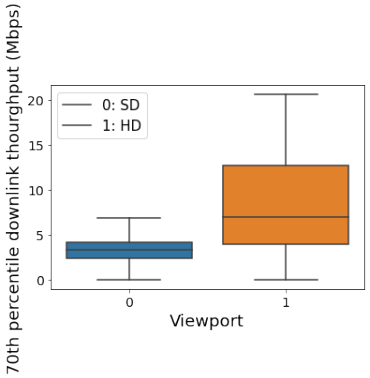


Fig. 12: $70th$ downlink throughput percentile (most relevant throughput percentile in Fig. 10), *SD* (0), *HD* (1)

viewport resolution precisely is targeted. Our goal is to provide the ISP with a mean to infer viewport resolution insights despite the end-to-end encryption of the video flows. Such inference can help the ISP to get an idea about the bandwidth requirements of her customers and their level of Quality of Experience (QoE) with the obtained network service, which can also help taking network management decisions (e.g., resource allocation, priority queuing) to improve such QoE [41].

We build a dataset matching $F_{inband+chunk}$ to viewport capacity and use it to train different supervised ML classification algorithms. We randomly pick videos from the catalog available in [43], then stream them under different network conditions emulated locally using the Linux *tc* utility. Each experiment consists of enforcing the bandwidth, playing out the selected video under the enforced QoS, collecting clear HTTP messages using the Chrome Web Request API and dumping the traffic in *pcap* files using tcpdump. The *pcap* files are used to calculate the feature set $F_{inband+chunk}$.

A. Viewport class classification

As we have seen before, the effect of the viewport is maximum in a bandwidth unlimited scenario. As bandwidth decreases, the different viewports converge to the same video resolution pattern, therefore we expect any viewport inference model to become less accurate as both classes (*SD*/*HD*) start overlapping. To assess the extent of such limitation, we test our model in different scenarios each featuring a different bandwidth configuration. We use Random Forest (available in python Scikit-Learn library [48]) because of its observed outperformance in our case compared to other classifiers such as Support Vector Machine, Decision Tree and Multi-Layer Perceptron (see later for comparison). To find the best tuning of the Random Forest algorithm, we apply at first a random search of best hyper-parameters values, then after reducing the space of search, we apply a grid search to get a fine-grained fitting of major parameters [49].

Fig. 13 highlights the accuracy of two classification models trained with our dataset. In plain, for each bandwidth setting on the x-axis, we highlight two Random Forest models trained on two different datasets, the blue model is trained with video samples conducted with one specific enforced bandwidth (the corresponding x-axis value), the orange one is a model trained with the aggregate set of video sessions obtained over all enforced bandwidth values. It follows that the blue model varies from one x-axis value to another one, whereas the orange model is the same over all x-axis values. We validate both models on a test set of 200 videos specific to each bandwidth scenario. In general, and regardless of the training set, the model accuracy is coherent with our intuition and increases w.r.t. the enforced bandwidth. For example, in case of enforced bandwidth of 3Mbps, both models show a low performance with a median accuracy of 62%. This is expected as for such low bandwidth, viewports show similar distributions for most important features (see Fig. 8). One can expect an even lower accuracy if the exact viewport resolution is to be predicted for such a low bandwidth value. Starting from 6 Mbps, both models show a median accuracy exceeding 80%, with the model based on mixed conditions showing better accuracy (in terms of both average and variance) than the model specific to the enforced bandwidth value, which is a good property of the orange model given its generality over different bandwidth scenarios. We recall that the best performance for both models is reached when no limitation is imposed on the network bandwidth with 92% median accuracy.

The previous results present a general evaluation of the model, yet, we need to evaluate the model per viewport class. Here, one can use metrics like *precision* and *recall* or simply the F1-score which is an average of both. To that aim, we benchmark a set of well known supervised machine learning algorithms, fit them with our dataset (for the no bandwidth limitation case) and compare them per class using the F1-score. We plot in Fig. 14 the k-fold (k =10) validation results for the set of machine learning algorithms we consider. According to this validation, the dataset is split into k folds, and at each of the k iterations, a new fold is used as a validation set while the k - 1 remaining folds form the training set. Average results are then calculated over the k iterations. In plain, Random Forest seems to be the most relevant algorithm, as it shows an average F1-score of 85%, while the other classification algorithms such as Linear Discriminant Analysis and Decision Tree come second and third with average F1-scores of 80% and 78% respectively. The lowest performance is recorded for the Multi-Layer Perceptron and Linear Regression classifiers with 72% and 68% F1-scores respectively. Moreover, we show in Table I the precision and recall values of a tree sample produced by our Random Forest model. The model classifies correctly 85% of the total video sessions issued from *HD* viewports and 93% of the sessions related to *SD* viewports. When our model labels a video session as *HD*, it is correct in 87% of the cases and in 89% of the cases for *SD*.

B. Viewport resolution classification

The previous analysis highlights the performance of our model in a binary scenario of *SD/HD* viewport classes. In this subsection, we illustrate the performance of our model in a multi-class scenario, where we aim at predicting the

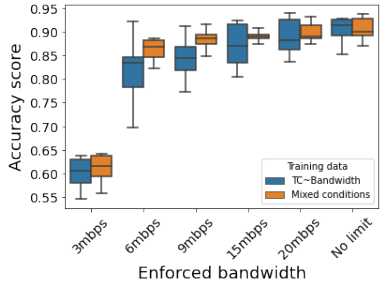


Fig. 13: Model accuracy vs enforced bandwidth

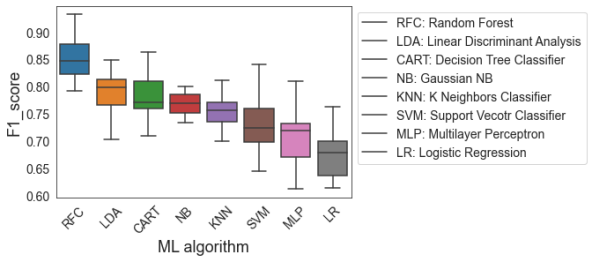


Fig. 14: ML algorithm comparison (no bandwidth limitation)

	Precision	Recall	F1
HD	0.87	0.85	0.86
SD	0.89	0.93	0.91

TABLE I: Random Forest case (precision/recall)

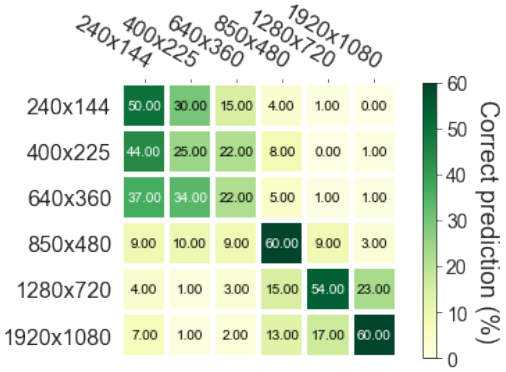


Fig. 15: Viewport resolution classification

exact viewport resolution as used in the experimental setup (ee Fig. 1). We show results for the Random Forest model trained with video sessions conducted in the no bandwidth limitation scenario. We leverage a heatmap to highlight the prediction accuracy of our model per viewport size.

We plot in Fig. 15 the confusion matrix of the predicted viewport sizes. The y-axis (rows) corresponds to the ground truth on viewport sizes, while the x-axis (columns) represents the predict ones. The value in case (i, j) represents the percentage of viewports of size i that are classified as of size j, the sum of elements in a row is equal to 100%. The color intensity of a case increases with its value. According to this heatmap, one can identify two regions (i) small viewports mainly from 240x144 up to 850x480, and (ii) high definition viewports of sizes 1280x720 and 1920x1080. The heatmap shows the difficulty to classify video sessions on small viewports, for instance, 44% and 37% of the video sessions on 400x240 and 640x360 viewports are labeled as 240x144. On the other hand, for large viewports, the majority of video sessions can be classified correctly with our set of features, as 54% and 60% of video sessions on 1280x720 and 1920x1080 viewports are correctly labeled. In the middle, the 850x480 viewport is the one with the largest uncertainty in the classification between the two viewport classes, with 28% of its video sessions labeled with inferior viewport sizes and 12% with superior viewport sizes. It is for this reason that we decided to consider viewports of this size as belonging to the *SD* class.

Following the same analysis for binary classification, we highlight in Table II the classification performance per class using the Precision/Recall and F1-score metrics. In plain, for the 400x225 and 640x360 viewports, the F1-score is the lowest with 25% and 27% respectively, hence highlighting the confusion of the model when it comes to video sessions on small screens. On the other hand, for large screens, the model is able to perform better with an F1-score of 55% for the 1280x720 viewport and 61% for the 1920x1080 one. This relatively low accuracy of the classification in the multi-class scenario is expected, as our analysis has pointed to two different subsets of viewports (*SD* and *HD*) presenting close properties internally for their inband network features and chunk size statistics. The result is a space of collision inside each of the subsets and therefore confusion regions where the model is of high uncertainty. Luckily the overlap is of less importance between the subsets leading to the good performance we have seen in the binary classification case.

C. Real time viewport classification

The statistics we used so far to train and test our model take into consideration the entire video session. This requires waiting until the end of the session to collect the features and predict the viewport, which can limit the usability of the method in practice by preventing from taking real time traffic engineering actions. For that, one needs to perform the classification as soon as the video starts playing out, thus allowing for mechanisms such as weighted fair queuing and load balancing to take place. So here we study the goodness of our model for viewport classification on the fly, which instead of using as input aggregated statistics on the entire video session, calculates features on the early part of the session.

We stream a total of 104 hours (4 days) and 70 hours (almost 3 days) of random YouTube videos using different *SD* and *HD* viewports respectively. We highlight in Fig. 16 the video duration distribution per viewport class. As expected, the two distributions look the same, with half of the videos requested from *SD* and *HD* viewports having a median duration of 120 seconds. We split this dataset into training and test sets. In

	Precision	Recall	F1
240x144	0.32	0.52	0.40
400x225	0.26	0.24	0.25
640x360	0.31	0.24	0.27
850x480	0.52	0.62	0.57
1280x720	0.59	0.52	0.55
1920x1080	0.70	0.54	0.61

TABLE II: Multi-class case: Precision/Recall & F1-score

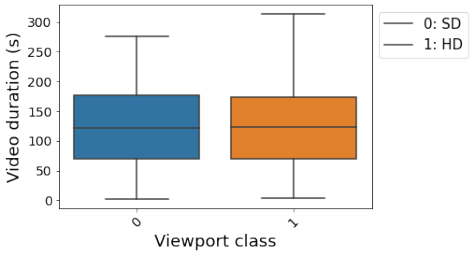


Fig. 16: Video duration distribution (seconds)

the training set, we compute the features by considering the entire video session. On the other hand, for the test set, we use a specific proportion of the video starting from its beginning and test over it. For instance, an input on the first 20% will consider a feature set $F_{inband+chunk}$ calculated over the first 20% of the video, and so on. To represent the proportions in seconds and give them a practical meaning, we consider the median video duration (120s) as reference duration, so proportions of 20% and 40% would correspond to the first 24 and 48 seconds of a video session, respectively.

We plot in Fig. 17 the F1-score w.r.t. the proportion used as input for the test. With no surprise, the larger the considered proportion of video session is, the higher the accuracy of the model becomes. This makes sense as the model gets more and more relevant input as compared to those used for the training. More importantly, the model still works with few seconds as input and provides good classification accuracy exceeding 80% on average. This confirms that the first few seconds of a video session do indeed carry an important signature of viewport class, with for example the first 24 seconds (assuming a median duration of 120 seconds) allowing a median F1-score of 80% (more than 78% in 75% of the cases). We recall that considering the full video data leads to 85% median F1-score. We shall thus confirm the feasibility of our approach for pseudo real-time viewport classification.

VIII. Conclusion and Future Work

In this paper, we present our methodology for building viewport classification models from YouTube encrypted video traces using controlled experimentation and machine learning. Our models infer the end-user viewport resolution from statistical features calculated over the encrypted video packets, fully or partially. Such information on the viewport can help the ISPs plan better traffic engineering actions for a more efficient network management and QoE optimization. Our

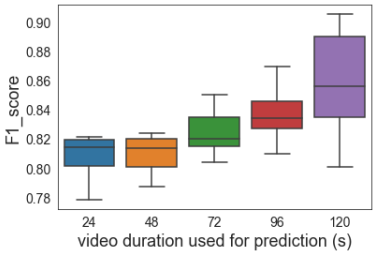


Fig. 17: Model accuracy vs video proportion considered

methodology starts by inferring chunk sizes, then relies on Gaussian Mixture Models (GMM) to separate video chunks from audio chunks. Statistics on video chunks are then used to train machine learning models for viewport classification. In a binary scenario of *SD* and *HD* viewports, our models show classification accuracy that improves with the available network bandwidth, and can go up to 92% in its median. The median F1-score can go up to 85%. Limiting the classification to the first few seconds of the video decreases its accuracy, but still leads to acceptable levels of F1-score. The inference of the exact viewport size shows a lower accuracy, as a subset of viewports presents similar statistical features, making the prediction more difficult to realize.

As future work, we plan to extend our study to cover other popular streaming platforms (e.g., Netflix) and viewport sizes (e.g., UHD). Meanwhile, our approach remains applicable to streaming platforms that provide video player API, data API and open source documentation to read the HTTP text messages and derive chunk related information. Moreover, we aim at reusing our results to perform optimal resource allocation at the edge of the network leveraging terminal characteristics and aiming at improving the Quality of Experience of end-users.

References

[1] Ericsson, "Ericsson Mobility Report, June 2018," https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf, 2018.

[2] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez, O. Hohlfeld, and G. Smaragdakis, "The lockdown effect: Implications of the covid-19 pandemic on internet traffic," in *Proceedings of the ACM Internet Measurement Conference*, 2020.

[3] E. Commission, "Commission and European regulators calls on streaming services, operators and users to prevent network congestion." https://ec.europa.eu/digital-single-market/en/news/commission-and-european-regulators-calls-streaming-services-/operators-and-users-prevent-network, 2020.

[4] Forbes, "Netflix Starts To Lift Its Coronavirus Streaming Restrictions." https://www.forbes.com/sites/johnarcher/2020/05/12/netflix-starts-to-liftits-coronavirus-streaming-restrictions/#7bcba5bf4738, 2020.

[5] sandvine, "The Global Internet Phenomena Report, October 2018," https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf, 2018.

[6] R. R. Pastrana-Vidal, J. C. Gicquel, C. Colomes, and H. Cherifi, "Sporadic frame dropping impact on quality perception," in *Proc. SPIE 5292, Human Vision and Electronic Imaging IX*, 2004.

[7] Y. Qi and M. Dai, "The effect of frame freezing and frame skipping on video quality," in *Proc. IEEE Int. Conf. Intell. Inform. Hiding Multimedia Signal Process.*, 2006.

[8] A. K. Moorthy, L. K. Choi, A. C. Bovik, and G. D. Veciana, "Video quality assessment on mobile devices: Subjective, behavioral and objective studies,," in *IEEE J. Sel. Topics Signal Process.*, 2012.

[9] G. Cermak, M. Pinson, and S. Wolf, "The relationship among video quality, screen resolution, and bit rate," in *IEEE Transactions on Broadcasting*, 2011.

[10] Google, "Chrome Web Request Extension," https://developer.chrome.com/extensions/webRequest, 2020.

[11] T. Stockhammer, "Dynamic adaptive streaming over http –: Standards and design principles," in *ACM conference on Multimedia systems (MMSys)*, 2011.

[12] C. Müller and C. Timmerer, "A vlc media player plugin enabling dynamic adaptive streaming over http," in *ACM international conference on Multimedia*, 2011.

[13] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *ACM Internet Measurement Conference*, 2016.

[14] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *IEEE International Symposium on Multimedia*, 2011.

[15] M. Khokhar, T. Ehlinger, and C. Barakat., "From network traffic measurements to qoe for internet video," in *IFIP Networking Conference*, YouTube catalogue link, 2019.

[16] O. Belmoukadam, T. Spetebroot, and C. Barakat, "Acqua: A user friendly platform for lightweight network monitoring and qoe forecasting," in *3rd International Workshop on Quality of Experience Management*, 2019.

[17] A. Mansy, M. Fayed, and M. Ammar, "Network-layer fairness for adaptive video streams," in *IFIP Networking Conference*, 2015.

[18] O. Belmoukadam, M. Khokhar, and C. Barakat., "On excess bandwidth usage of video streaming: when video resolution mismatches browser viewport," in *11th IEEE International Conference on Networks of the Future*, 2020.

[19] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "Yomoapp: A tool for analyzing qoe of youtube http adaptive streaming in mobile networks," in *European Conference on Networks and Communications (EuCNC)*, 2015.

[20] "Meteor: Free internet speed & app performance test," 2018, https://meteor.opensignal.com/.

[21] "Rtr-nettest," 2018, https://www.netztest.at/en/.

[22] Q. A. Chen, H. Luo, S. Rosen, Z. M. Mao, K. Iyer, J. Hui, K. Sontineni, and K. Lau, "Qoe doctor: Diagnosing mobile app qoe with automated ui control and cross-layer analysis," in *Internet Measurement Conference (IMC)*, 2014.

[23] D. Joumblatt, J. Chandrashekar, B. Kveton, N. Taft, and R. Teixeira, "Predicting user dissatisfaction with internet application performance at end-hosts," in *2013 Proceedings IEEE INFOCOM*, 2013.

[24] K. . Chen, C. . Tu, and W. . Xiao, "Oneclick: A framework for measuring network quality of experience," in *IEEE INFOCOM 2009*, 2009.

[25] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, and H. Yan, "Prometheus: Toward quality-of-experience estimation for mobile apps from passive network measurements," ser. HotMobile '14, 2014.

[26] W. Robitza, S. Göring, A. Raake, D. Lindegren, G. Heikkilä, J. Gustafsson, P. List, B. Feiten, U. Wüstenhagen, M.-N. Garcia, K. Yamagishi, and S. Broom, "HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P.1203 – Open Databases and Software," in *9th ACM Multimedia Systems Conference*, 2018.

[27] A. Raake, M.-N. Garcia, W. Robitza, P. List, S. Göring, and B. Feiten, "A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1," in *9th International Conference on Quality of Multimedia Experience (QoMEX)*, 2017.

[28] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive ip traffic," in *IEEE/ACM Transactions on Networking*, 2012.

[29] R. Thupae, B. Isong, N. Gasela, and A. Abu-Mahfouz, "Machine learning techniques for traffic identification and classification in sdwsn: A survey," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018.

[30] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," in *IEEE Transactions on Information Forensics and Security*, 2013.

[31] Opendpi, "," http://www.opendpi.org/, 2012.

[32] J. Khalife, A. Hajjar, and J. E. Díaz-Verdejo, "Performance of opendpi in identifying sampled network traffic," in *Journal of Networks*, 2013.

[33] LiJuan Zhang, DongMing Li, Jing Shi, and JunNan Wang, "P2p-based weighted behavioral characteristics of deep packet inspection algorithm," in *International Conference on Computer, Mechatronics, Control and Electronic Engineering*, 2010.

[34] F. Dehghani, N. Movahhedinia, M. R. Khayyambashi, and S. Kianian, "Real-time traffic classification based on statistical and payload content features," in *2nd International Workshop on Intelligent Systems and Applications*, 2010.

[35] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape, "Portload: Taking the best of two worlds in traffic classification," in *INFOCOM IEEE Conference on Computer Communications Workshops*, 2010.

[36] C. Wang, X. Zhou, F. You, and H. Chen, "Design of p2p traffic identification based on dpi and dfi," in *International Symposium on Computer Network and Multimedia Technology*, 2009.

[37] G. La Mantia, D. Rossi, A. Finamore, M. Mellia, and M. Meo, "Stochastic packet inspection for tcp traffic," in *IEEE International Conference on Communications*, 2010.

[38] A. Rao and P. Udupa, "A hardware accelerated system for deep packet inspection," in *ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2010.

[39] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, 2017.

[40] F. Li, J. Chung, and M. Claypool, "Silhouette: Identifying youtube video flows from encrypted traffic," in *28th ACM SIGMM Workshop*, 2018.

[41] O. Belmoukadam, M. Khokhar, and C. Barakat, "On accounting for screen resolution in adaptive video streaming: a qoe driven bandwidth sharing framework," in *CNSM*, 2019.

[42] Statcounter, "Desktop screen resolution stats worldwide," 2021, https://gs.statcounter.com/screen-resolution-stats/desktop/worldwide.

[43] YouTube, "Video Catalogue," https://drive.google.com/open?id=1tu0sBInt8xJ9Zn32IDlh6DW_ju2FhEou, 2020.

[44] M. Khokhar, T. Ehlinger, and C. Barakat., "From network traffic measurements to qoe for internet video," in *IFIP Networking Conference*, YouTube catalogue link, 2019.

[45] O. Nawaz, T. N. Minhas, and M. Fiedler, "Qoe based comparison of h.264/avc and webm/vp8 in an error-prone wireless network," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017.

[46] YouTube, "Itag documentation," https://www.genyt.xyz/formats-resolution-youtube-videos.html, 2020.

[47] Git, "Itag catalogue," https://gist.github.com/sidneys/7095afe4da4ae58694d128b1034e01e2, 2020.

[48] Python, "Random Forest classifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2020.

[49] M. blog, "Hyperparameter Tuning the Random Forest in Python," https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74, 2020.

AUTHOR BIOGRAPHY

Othmane Belmoukadam a 3rd year PhD student at Inria Sophia Antipolis. He received a computer science MSc from the university of Côte d'Azur, France, in 2018. His main research interests are QoE driven resource allocation, QoE modeling, Data science and machine learning applied to networking.

Chadi Barakat is Senior Research Scientist in the Diana project-team at Inria, Sophia Antipolis Research Centre. He served in the program committees of many international conferences as CoNEXT, Infocom, IMC, ICNP, and PAM. His main research interests are in Internet measurements and network data analytics, user quality of experience, and mobile wireless networking. Chadi Barakat is senior member of the IEEE and of the ACM.