I See What you See: Real Time Prediction of Video Quality from Encrypted Streaming Traffic

Sarah Wassermann*, Michael Seufert†, Pedro Casas*, Li Gang◇, Kuang Li◇
* AIT Austrian Institute of Technology, † University of Würzburg, ◇ Huawei Technologies

ABSTRACT

We address the problem of real-time QoE monitoring of HAS, from the ISP perspective, focusing in particular on video-resolution analysis. Given the wide adoption of end-to-end encryption, we resort to machine-learning models to predict different video resolution levels in a fine-grained scale, ranging from 144p to 1080p resolution, using as input only packet-level data. The proposed measurement system performs predictions in real time, during the course of an ongoing video-streaming session, with a time granularity as small as one second. We consider the particular case of YouTube video streaming. Empirical evaluations on a large and heterogeneous corpus of YouTube measurements demonstrate that the proposed system can predict video resolution with very high accuracy, and in real time. Different from state of the art, the prediction task is not bound to coarse-grained video quality classes and does not require chunk-detection approaches for feature extraction.

CCS CONCEPTS

• **Networks → Network measurement**; **Network monitoring**; *Network performance analysis*; • **Computing methodologies → Supervised learning**.

KEYWORDS

Network Monitoring; QoE; HTTP Adaptive Video Streaming; Encrypted Traffic.

ACM Reference Format:
Sarah Wassermann*, Michael Seufert†, Pedro Casas*, Li Gang◇, Kuang Li◇. 2019. I See What you See: Real Time Prediction of Video Quality from Encrypted Streaming Traffic. In *4th Internet-QoE Workshop: QoE-based Analysis and Management of Data Communication Networks (Internet-QoE'19), October 21, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3349611.3355549

ACM ISBN 978-1-4503-6927-5
https://doi.org/10.1145/3349611.3355549

1 INTRODUCTION

Video streaming is one of the key applications on the Internet. To satisfy end users and avoid customer churn, Internet Service Providers (ISPs) strive to deliver a high video-streaming Quality of Experience (QoE). In the past, video streaming mostly suffered from re-buffering events and initial playback delays [3, 15]. Such degradations have been partially mitigated by adapting the video bitrate to the network conditions, using the so-called HTTP Adaptive Streaming (HAS) protocols. For HAS, the video content has to be available in multiple bitrates, i.e., quality levels, and split into small segments each containing a few seconds of playtime. The adaptation logic on the client side requests the next part of the video in an appropriate bitrate, such that playback delay is minimized, stalling is avoided, and the quality level is maximized to best utilize the available bandwidth.

To change the video bitrate, also the visual quality level of the streamed video has to be altered, e.g., in terms of resolution, frame rate, or compression, which introduces an additional impact on QoE. As a consequence, ISPs are highly interested in solutions able to detect events when the played out quality level drops as soon as they happen, to take appropriate countermeasures. The trend towards end-to-end encryption (e.g., HTTPS), however, has significantly reduced the visibility of network operators on the traffic of their customers, making the monitoring process more challenging and cumbersome. It is no longer possible to rely on Deep Packet Inspection (DPI) based approaches to analyze the video data contained in each packet to reconstruct the streaming process and the video buffer [1] or to intercept and analyze segment requests.

In this paper, we present a machine-learning-based system to predict the video resolution in YouTube with very high accuracy, and in real time. In particular, and different from previous work, it focuses on the fine-grained prediction of the resolution. To do so, our approach analyzes ongoing streaming sessions using fine-grained time slots of

one-second length, computing multiple lightweight, statistical features from the video traffic in a stream-based fashion. Besides per-time-slot features, our approach additionally computes features for different temporal aggregations of past slots, including a short-term memory capturing the last T slots, as well as a long-term memory, aggregating all time slots since the start of the video session.

While there have been already multiple proposals presented in the past to deal with this prediction problem, the contributions brought by the proposed approach as compared to the state of the art are various and paramount, and in particular from an application perspective, to enhance the monitoring tasks of the ISP:

1 - Fine-grained, real-time operation: it predicts video resolution in real time, during the live streaming of a video session, using a temporal granularity as small as one second, enabling quick anomaly detection and troubleshooting approaches, as well as proactive traffic management.

2 - Stream-like computation, without chunk-detection requirements: different from all previously presented proposals, our methodology continuously extracts features from the encrypted stream of packets in a stream-like, recursive manner, using bounded - and lightweight - memory footprints; this enables its execution on top of limited memory hardware, such as set-top boxes or home routers, which are nowadays the most preferred devices for conducting end-customer monitoring by major vendors.

3 - Extensive machine-learning-model benchmarking: also different from previous work, we devote a significant part of the study to benchmark different machine-learning algorithms, as well as evaluating their performance using different sets of inputs, engineered by feature selection.

4 - Empirical validation over an heterogeneous YouTube dataset: lastly, we show that the proposed technique performs accurately under a very heterogeneous set of scenarios, by empirically testing it over a large dataset of 15,000 streaming sessions of **different YouTube videos**. The dataset covers different access technologies (WiFi and LTE), different transport protocols (QUIC and TCP), variable bandwidth configurations, different players and devices (standard HTML player in laptops and native YouTube app in smartphones), as well as considering measurements at 4 different ISPs in 4 different EU countries. This is an additional delimit from state of the art, where proposals are generally validated over limited and less representative scenarios.

2 RELATED WORK

The most important results on QoE for HTTP adaptive streaming (HAS) are summarized in [15]. More recent publications also confirm the findings that stalling, initial delay, and quality adaptation are the most dominant QoE-relevant metrics. Although adaptation incurs less severe QoE degradation than stalling, its impact should not be neglected. Each adaptation dimension (e.g., resolution, frame rate, quantization) has a specific impact on the perceived quality [15]. It has been shown that a quality switch implies a QoE degradation, and that the QoE changes according to the adaptation direction, even though switching down the video quality will have a stronger negative impact on video QoE [9]. The adaptation amplitude is the most dominant factor and a high amplitude leads to a low QoE, while low amplitudes might not be detectable [12]. Although a high frequency of quality adaptation will be annoying for end users [12], the actual quality changes have little impact on QoE. Only the resulting reduction of the time on high video quality causes relevant QoE degradation [6].

Due to the trend towards end-to-end encryption, DPI-based approaches are no longer effective. This has motivated a recent trend in QoE-based network monitoring using low-level network measurements rather than relying on application-layer metrics. Authors in [13] evaluate machine-learning-based architectures that estimate YouTube QoE from features derived from packet sizes, inter-arrival times, and throughput measurements. A similar approach is presented in [2], where authors rely on real cellular network measurements and machine-learning models to predict typical QoE indicators for streaming services (e.g., played resolutions, stalling events), based on features such as round-trip times, packet loss and chunk sizes. The authors of [16] estimate video-quality metrics (initial delay, stalling ratio, number of stallings, total stalling time) and user engagement for YouTube videos watched on smartphones, relying on machine learning and network-layer features. [8] focuses on the reconstruction of buffered playtime at the video player side, as previously done in [14], but for encrypted traffic. This is leveraged to estimate video-QoE metrics in [10].

Different from these papers, our method estimates video quality from encrypted video traffic in real time by using a stream-analysis approach. It considers three windows (current-, trend-, and session-based) with a minimal memory footprint, i.e., the windows store only a small feature set, computable in fixed memory bounds, allowing for effective real-time operation. The features are based on packet-level statistics of the network traffic and do not require chunk-detection mechanisms, allowing to accurately and continuously recognize QoE degradations within time slots of one second.

The two most similar approaches to ours are Requet [5] and the system presented in [11]. However, the proposed method improves both in multiple aspects: (*i*) while both approaches claim to be real-time, there is no temporal evaluation of the computational cost inquired in the feature extraction procedures, questioning their claims; (*ii*) while Requet also provides the same fine-grained classes for prediction of video resolution as our approach – [11] predictions are

way more coarse-grained, limiting their usability in practice; (*iii*) Requet requires chunk-detection mechanisms to extract chunk-based features, which is error prone and introduces additional complexities, which is not needed by ours; (*iv*) [11] operates at a 10 s temporal scale, limiting the applicability of the approach for critical troubleshooting applications.

3 METHODOLOGY & DATASET

The proposed system considers a video-streaming session as an ordered sequence of contiguous time slots of fixed length. Throughout this work, we use a slot length of 1 s, which constitutes a good trade-off between prediction delay and accuracy. At each time t, multiple statistical features are extracted from the packets contained at different temporal aggregations of current and past time slots, including: features extracted from the current time slot; short-memory or *trend* features, extracted from the last T time slots – in this paper we take $T = 3$; and long-memory or *progressive* features, extracted from all past time slots until time t, from the start of the video session. The total number of features computed from these slots adds to a total of 207 features. A prediction is computed at every new time slot, using the extracted features as input for machine-learning models.

Features include the number of total, uplink, and downlink packets and transferred bytes, as well as time-based features, including the time from the start of the slot until the first packet, and the time between the first and last packets of the time slot. To avoid the need to store previous traffic or detailed information about packets in the past, the entire feature set is computed in a stream, online fashion. This approach has minimal memory footprint, and can run in constrained hardware equipment. As we show next, this allows for real-time feature extraction and prediction.

3.1 Dataset Description

We streamed and recorded more than different 15,000 YouTube video sessions between June 2018 and February 2019, resulting in a total of more than 4,600,000 one-second time slots. For this task, we used a Java-based monitoring tool which relies on the Selenium browser automation library to automatically start a Chrome browser and browse to randomly selected YouTube videos. We configured Chrome such that all HTTP requests were logged and QUIC traffic was enabled. We injected a JavaScript-based monitoring script into the web page to record, every 250 ms, the current timestamp, as well as the current video playtime, buffered playtime, video resolution, and player state.

To obtain a generalizable model, we streamed the video sessions with highly diverse network characteristics. Videos were streamed either from a home or corporate WiFi network, or an LTE mobile network, using both QUIC and TCP. The maximum bandwidth was roughly 20 Mbps. Some streaming sessions faced bandwidth limitations, both in the uplink and downlink directions. Bandwidth limitations were either constant on a level of 300 kbps, 1 Mbps, 3 Mbps, or 5 Mbps, or fluctuated between these levels every 1 to 5 minutes. Videos were recorded over four different ISPs, and from different vantage points, located in four different EU countries, including Austria, Italy, France, and Germany. We captured network traffic for every streamed video session, and extracted features from the traffic of all flows composing the video-streaming session, i.e., all other non-YouTube flows were discarded. Finally, we also considered the recently published YouTube App open dataset [7], which includes measurements from the native, mobile Android YouTube app.

Recorded video sessions have durations of up to 11 minutes, with an average duration of about 5 minutes. Typical video-resolution levels in YouTube include 144p, 240p, 360p, 480p, 720p, and 1080p. The distribution of resolution levels in the considered dataset shows that the adaptation logic of YouTube decided to stream videos most of the time in 480p (55%), but also in very low resolutions (9% 144p, 6% 240p, 10% 360p). HD resolution was rare (18% 720p, 1% 1080p). Even if supported by YouTube, video resolutions above 1080p are not present in our dataset, therefore the study focuses on video resolution levels ranging from 144p to 1080p.

	Training time (min)	Accuracy (%)
DT	43	92
RF10	2	92
ADA	125	68
ERT10	1	90
BAGGING	37	95
BAYES	1	42
KNN	9	73
NN	507	58
SVM	194	54

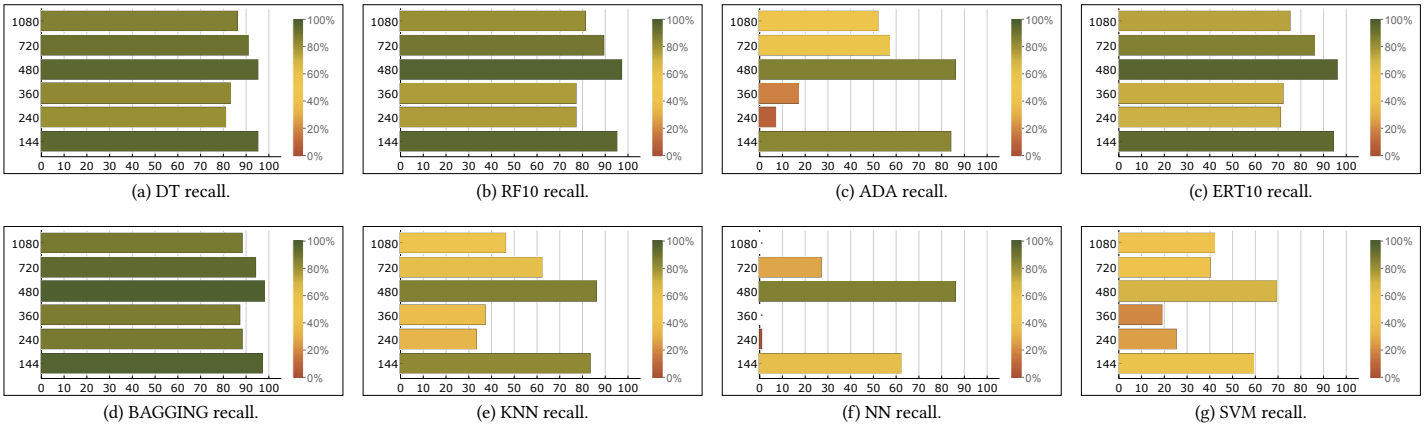
Table 1: Benchmarking of different ML models.

4 VIDEO RESOLUTION EVALUATION

We tackle the estimation of the video resolution as a classification task. The classes correspond to the typical YouTube video resolution levels, namely, 144p, 240p, 360p, 480p, 720p, and 1080p. Thus, our task is based on six classes, which is substantially more precise and fine-grained than other approaches such as [2, 11, 13], using at most 3 different levels.

4.1 Model Benchmarking

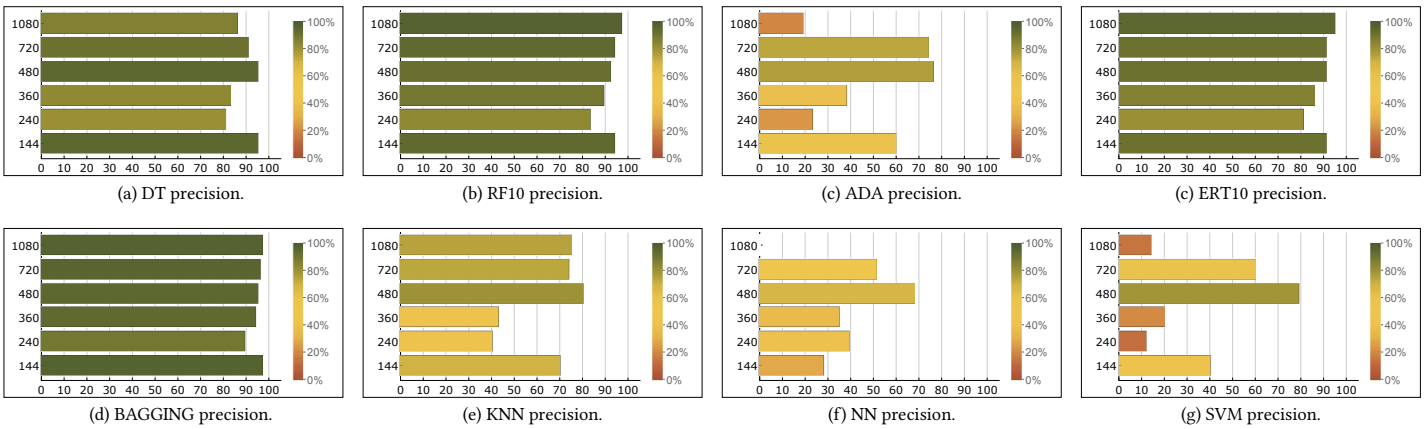
We benchmark nine machine-learning algorithms using 5-fold stratified cross-validation. With this kind of cross validation, we ensure that the five folds are computed by preserving the percentage of samples for each class. For each algorithm, we report the total running time, i.e. the time



(a) DT recall. (b) RF10 recall. (c) ADA recall. (c) ERT10 recall.

(d) BAGGING recall. (e) KNN recall. (f) NN recall. (g) SVM recall.

Figure 1: Accuracy per class (i.e. recall) obtained by the benchmarked ML models for the resolution prediction.



(a) DT precision. (b) RF10 precision. (c) ADA precision. (c) ERT10 precision.

(d) BAGGING precision. (e) KNN precision. (f) NN precision. (g) SVM precision.

Figure 2: Precision per class obtained by the benchmarked ML models for the resolution prediction.

needed to process the whole dataset, and the overall accuracy. Tests were done on a server machine, see Section 4.3 for specific hardware description. We consider the following models: (1) decision trees (DT), (2) random forests with 10 trees (RF10), (3) Adaboost using 50 trees (ADA), (4) an ensemble with 10 extremely randomized trees [4] (ERT10), (5) bagging with 10 trees (BAGGING), (6) Naïve Bayes (BAYES), (7) k-nearest neighbors with k= 5 (KNN), (8) feed forward neural networks with 3 hidden layers (NN), and (9) SVMs.

For DT, RF10, and ERT10, we penalize prediction errors for each sample i based on the occurrence frequency of its class, which improves the prediction accuracy for these classes. We further exploit the fact that RF10, ERT10, BAGGING, and KNN can be parallelized and let them run on the 48 virtual cores of the server machine, in a parallelized fashion. For NN, we rely on TensorFlow on GPU, while we use the well-known scikit-learn library for the remaining models.

Table 1 reports the model training processing times, and the accuracies obtained by the models. Besides Adaboost, all tree-based models provide high accuracy. KNN also outputs decent results with an accuracy of 73%. BAYES is by far the worst, most probably linked to a lack of input independence. NN and SVM also yield bad results, especially when considering their significantly higher training times. We can appreciate the benefit of parallelization: besides BAYES, the fastest algorithms are the ones which are parallelizable, which is a non-negligible advantage for these models. For instance, RF10 and ERT10 completed their tasks in at most 2 minutes, while ADA, NN, and SVM took several hours.

As the video-resolution classes are highly imbalanced, we take a closer look at the per-class accuracy (i.e. recall) and precision. Results are reported in Figures. 1 and 2, respectively. Due to its bad performance, we do not consider BAYES for this analysis. In Figure 1, we observe that the 480p-class is accurately detected by all of the eight models, with SVM being the worst with an accuracy below 70%. DT, RF10, ERT10, and BAGGING achieve a near perfect score for this video resolution. This comes as no real surprise, as more than 50%

Features	# Features	Accuracy (%)
F_C	69	70
F_T	69	73
F_S	69	96
F_{DOWN}	81	90
F_{UP}	81	90
F_{TOP20}	20	95

Table 2: Video-resolution-prediction performance using different feature subsets.

of the time slots have a resolution of 480p. However, it is interesting to note that most models accurately estimate the 144p class, even though it is a very underrepresented class, with only 9% of the samples. For all the models, these two classes are the ones that are the most accurately detected. For some models, in particular for ADA and NN, it is challenging to accurately classify the 240p and 360p resolutions; for NN, the accuracy for 360p is even close to 0%. From Figure 2, we can see that the precision of the benchmarked models is similar to the recall: it is at its highest for DT, RF10, ERT10, and BAGGING (always higher than 80%), while it is relatively low for ADA, NN, and SVM. This per-class analysis further indicates that DT, RF10, ERT10, and BAGGING provide excellent prediction results. Out of these four models, RF10 is the most appropriate one for the real-time prediction task, as besides being accurate it is also extremely fast. Therefore, from now on, **further evaluation results consider the usage of the RF10 model as predictor**.

4.2 Feature Importance Analysis

We study now the relevance of the different input features for video-resolution estimation, in terms of prediction accuracy. In particular, we subdivide the full input set of 20 features into six feature sets: (1) F_C – the set of features computed from the current time slot; (2) F_T – the set of features computed from the short-memory, trend window; (3) F_S – the set of features computed from the long-memory, progressive window; (4) F_{DOWN} – all features computed from the downlink traffic; (5) F_{UP} – all features computed from the uplink traffic; (6) F_{TOP20} – the 20 most important features, as selected by the RF10 model.

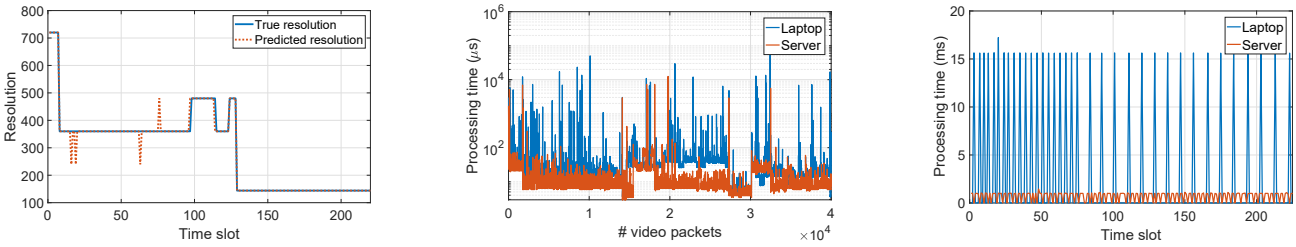
We rely on 5-fold stratified cross-validation to benchmark the accuracy of the system using each of these six feature sets. Table 2 reports the number of features in each feature set, and the accuracy realized through each of them. F_C and F_T feature sets yield the poorest results, with a down-performance of about 20 percentage points with respect to the usage of the full feature set. This indicates that using only snapshot and/or short-term memory features is not sufficient to accurately predict video resolution. Performance is much better when using global, session-based features – F_S, F_{DOWN}, or F_{UP}, with the F_S feature set achieving even higher accuracy – +4 percentage points, than when considering the full feature set. This is not surprising, as it is commonly accepted that using noisy or poorly correlated-to-the-target input features can actually degrade prediction performance for certain machine-learning models. The advantage here is that the F_S feature set consists of 81 out of the original 207 features, reducing the computational requirements needed for feature extraction and model training.

Last, we choose the 20 most important features – F_{TOP20}, according to an embedded feature-selection technique, which basically ranks the relevance of each feature according to its prediction accuracy, for the specific RF10 model. We proceed as follows: for each run of the 5-fold stratified cross-validation, we select the 20 most relevant features based on the training folds, and test the model trained on those 20 features on the test fold. We select the overall top 20 features based on their importance score averaged over the five folds, as well as the average accuracy of the algorithm over the folds with only the selected features. Session-related features are the most relevant among the top 20 features in F_{TOP20}, including features related to the session-downlink-throughput pattern (e.g., mean throughput, burst throughput, etc.), as well as related to the interarrival times between packets (e.g., mean and variation values). Prediction accuracy using the F_{TOP20} feature set is as high as 95%, indicating that a carefully engineered input-feature set can provide excellent results, saving significant resources in terms of feature-computation time. Indeed, F_{TOP20} consists of roughly 10% of the whole feature set, and provides a +3 percentage-point out-performance.

4.3 Computational Time Analysis

To sustain our claims of real-time operation capabilities, we devote the last part of the study to the analysis of the computational times involved in the complete end-to-end online process, including online feature computation and prediction times. We consider the RF10 model using the full feature set as input – 207 features, which represents an upper bound to the computational complexity of the system. Figure 3(a) depicts the fine-grained operation of the predictor over an example video-streaming session, showing the continuous evolution of both the real video resolution level and its prediction. Note that the prediction is almost perfect over time, with errors located mainly at the beginning of the video-streaming session – when the adaptation logic makes more dynamic changes.

Figure 3 additionally reports the computational times needed for the continuous update of the complete feature set in (b) and the video-resolution-prediction times in (c). For the sake of completeness, we perform the evaluations using two different hardware configurations, corresponding to different



(a) Video-resolution prediction. (b) Features update time at each new packet. (c) Prediction time.

Figure 3: Video-resolution prediction and computation analysis.

physical machines: the *server* configuration corresponds to a high-end computing server, equipped with two Intel Xeon Silver 4116 processors including 12 physical cores each – a total of 48 virtual cores thanks to Intel HyperThreading, 128 GB of RAM, and a NVIDIA GeForce RTX 2080 TI graphics card with 11 GB of VRAM; the *laptop* configurations corresponds to a standard end-user notebook computer, including an Intel Core i5-4200U CPU with 2 physical cores and a total of 4 virtual ones, 8 GB of RAM, and an integrated GPU Intel HD Graphics 4400.

Figure 3(b) depicts the time needed to update the full feature set at each packet arrival; as explained before, feature computation is done in a stream manner, with features updated with every new incoming packet. Feature updates are done extremely fast on both hardware configurations, taking only some microseconds. Most importantly, besides some small transient effects, the whole feature set is continuously updated in almost constant time. On *server*, the slowest feature-set update takes about 12 ms, while the average duration is 13 μs. More than 90% of the updates take less than 25 μs. On *laptop*, the average processing duration is 37 μs, with a maximum value of 129 ms.

Finally, Figure 3(c) depicts the time needed by the RF10 model to process the set of inputs and provide a prediction at every new time slot. To make computations harder, we disable the parallelization of the algorithm for the prediction phase. On *server*, almost every prediction is performed in around 1 ms, with an average of 0.7 ms and a maximum of around 1.4 ms. On *laptop*, the average prediction time is 2.5 ms, with a maximum of around 15 ms. These results demonstrate that overall, the proposed system is able to perform video-resolution predictions in real time, with an end-to-end delay way below the time slot length of 1 s.

5 CONCLUSION

We presented a machine-learning-based system for real-time prediction of video resolution in HAS, working fully on top of encrypted network traffic. The proposed system performs predictions with a fine-grained temporal resolution of one second, which is, up to date, the smallest temporal granularity used for video quality predictions in the context of encrypted traffic. In addition, video resolution is predicted over six different resolution levels, extending previous work in the area. We built a comprehensive dataset consisting of more than 15,000 randomly selected YouTube videos streamed under diverse network conditions, devices, ISPs, transport protocols, and geographical locations. We benchmarked multiple machine-learning models and found that tree-based techniques are by far the most appropriate models for the proposed task. Indeed, tree-based models provide highly accurate results and are execution-fast. We also showed that features tracking the characteristics of a video session since the start of the streaming are the most relevant ones in terms of prediction accuracy, and that by only using 20 input features, a simple random-forest model can achieve highly accurate results. Last, we also showed that the stream-based strategy used by our system to perform feature computations is highly efficient in terms of computational times, underlining its real-time properties.

REFERENCES

[1] P. Casas et al., "YOUQMON: A System for On-line Monitoring of YouTube QoE in Operational 3G Networks". ACM SIGMETRICS Perform. Eval. Rev. 41(2), 2013.
[2] G. Dimopoulos et al., "Measuring Video QoE from Encrypted Traffic". In ACM IMC Conference, 2016.
[3] S. Egger et al., "Waiting Times in Quality of Experience for Web Based Services". In QoMEX Conference, 2012.
[4] P. Geurts et al., "Extremely Randomized Trees". Machine Learning 63(1), 2006.
[5] C. Gutterman et al., "Requet: Real-time QoE Detection for Encrypted YouTube Traffic". In ACM MMSys Conference, 2019.
[6] T. Hoßfeld et al., "Assessing Effect Sizes of Influence Factors Towards a QoE Model for HTTP Adaptive Streaming". In QoMEX Conference, 2014.
[7] T. Karagkioules et al., "A Public Dataset for YouTube's Mobile Streaming Client". In TMA/MNM Workshop, 2018.
[8] V. Krishnamoorthi et al., "BUFFEST: Predicting Buffer Conditions and Realtime Requirements of HTTP(S) Adaptive Streaming Clients". In ACM MMSys Conference, 2017.
[9] B. Lewcio et al., "Video Quality in Next Generation Mobile Networks - Perception of Time-varying Transmission". In IEEE CQR Workshop, 2011.
[10] T. Mangla et al., "eMIMIC: Estimating HTTP-based Video QoE Metrics from Encrypted Network Traffic". In TMA Conference, 2018.
[11] M. H. Mazhar et al., "Real-time Video Quality of Experience Monitoring for HTTPS and QUIC". In IEEE INFOCOM Conference, 2018.
[12] P. Ni et al., "Flicker Effects in Adaptive Video Streaming to Handheld Devices". In ACM MM Conference, 2011.
[13] I. Orsolic et al., "YouTube QoE Estimation Based on the Analysis of Encrypted Network Traffic Using Machine Learning". In QoEMC Workshop, 2016.
[14] R. Schatz et al., "Passive YouTube QoE Monitoring for ISPs". In FINGNet Workshop, 2012.
[15] M. Seufert et al., "A Survey on Quality of Experience of HTTP Adaptive Streaming. IEEE Communications Surveys & Tutorials 17(1), 2015.
[16] S. Wassermann et al., "Machine Learning Models for YouTube QoE and User Engagement Prediction in Smartphones". ACM SIGMETRICS Perform. Eval. Rev. 46(3), 2019.