In-Network QoE and KPI Monitoring of Mobile YouTube Traffic: Insights for Encrypted iOS Flows

Irena Oršolić, Petra Rebernjak, Mirko Sužnjević and Lea Skorin-Kapov
University of Zagreb, Faculty of Electrical Engineering and Computing, 10000 Zagreb, Croatia
Email: {irena.orsolic, petra.rebernjak, mirko.suznjevic, lea.skorin-kapov}@fer.hr

***Abstract*—Solutions for in-network monitoring of QoE-related KPIs are a necessary prerequisite to detecting potential impairments, identifying their root cause, and consequently invoking QoE-aware management actions. We leverage a machine learning approach to train QoE and KPI classifiers for mobile YouTube video streaming sessions using features extracted from encrypted QUIC traffic. With previous studies having shown different service behavior across different access networks and different OSs, we go beyond related work and specifically address iOS measurements and models. We assess the performance of models trained on data from a lab WiFi environment and an iOS device through cross-validation, achieving promising results. Using the dataset collected in a lab WiFi network, and two additional datasets collected in operational mobile networks, we further report on the promising applicability of classifiers trained using the WiFi dataset when applied to traffic collected using mobile network probes. The implications of such findings show the potential to use the same classifiers for multiple usage scenarios, thus reducing efforts needed for data collection and training. Finally, we discuss the extent to which models previously trained for Android usage scenarios are applicable for the iOS platform.**

I. Introduction

With the continuous growth in mobile Internet traffic [1], network operators are faced with the challenge of efficiently managing constrained network resources so as to maintain a satisfied user base and prevent customer churn. This inherently calls for a Quality of Experience (QoE) driven network management approach, whereby QoE impairments need to be monitored and mitigated. From the wide variety of services and apps accessed from smartphones, the largest impact on the growth of network traffic is attributed to popular Over-the-Top (OTT) video streaming services (e.g., YouTube, Netflix, Facebook video). In previous years, video QoE monitoring solutions deployed by Internet Service Providers (ISPs) have relied on Deep Packet Inspection (DPI) to obtain insights into video quality levels by reading data from application-level headers. Due to security and privacy concerns, most streaming services have now introduced traffic encryption, thus deeming these DPI methods obsolete. Going beyond application-layer encryption, such as TLS, increasing deployment of the QUIC protocol [2] has introduced end-to-end encryption at the transport layer [3], [4]. Today, assuming limited to no information exchange between ISPs and OTT service providers, ISPs for the most part rely solely on the monitoring of statistical traffic characteristics for QoE estimation and root cause analysis of potential quality impairments. Popular video services implement the HTTP Adaptive Streaming (HAS) paradigm, primarily to decrease the occurrence of stalling (or rebuferring) as the primary cause of QoE degradations [5], making the estimation of application-level Key Performance Indicators (KPIs), or overall QoE, even more challenging.

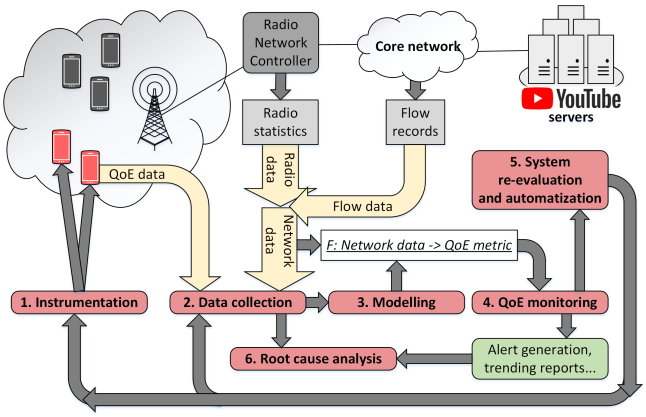


Fig. 1: QoE monitoring architecture (adapted from [6]).

Several recent studies have relied on machine learning (ML) techniques to find patterns in network-level data that correlate with application-level degradations [6], [7], [8], [9], [3], [4]. Figure 1 (adapted from [6]) depicts such a generic approach in a mobile network scenario. Following the instrumentation of measurement tools, data collection, and modeling (steps covered in aforementioned research), trained models can be deployed in the network. In-network measurements (e.g., radio stats, flow data) can be provided in real time to deployed models to calculate desired QoE metrics. A network operator can thus monitor QoE impairments, generate alerts regarding severe issues, aggregate QoE data to create trending reports, and finally use collected data for deriving the root cause of QoE degradations. At the same time, new application-level ground truth data can be collected to re-evaluate models over time and detect the need for a model update. Such updates may be necessary in response to changes such as those related to OTT service quality adaptation logic, or changes in underlying protocols (e.g., move from TLS to QUIC). In this paper, we follow the depicted approach, focusing on the first three points (instrumentation, data collection, modeling), and test numerous trained ML models that classify QoE and various KPIs of individual video streaming sessions.

A prerequisite to training ML-based KPI/QoE estimation models is the instrumentation of client-side tools to collect

978-3-903176-14-0

ground truth data regarding application-layer metrics. Numerous such apps, relying on different techniques for obtaining the data, were developed over the last years [10], [11], [8], [12], [13], [14], [15], [16], [4], [17]. While most of the listed approaches are focused on YouTube on Android, we point out the approaches also applicable for iOS [12], [15]. Besides utilization in the QoE/KPI model training process, such apps can also be used to characterize the service behavior. Previous studies have reported significant differences in YouTube behavior across different OSs [18], [19], possibly due to the use of different HAS mechanisms, such as DASH (Dynamic Adaptive Streaming over HTTP) and HLS (HTTP Live Streaming). In general, YouTube's player behavior in terms of adaptation mechanisms has been the topic of numerous studies [20], [8], [21], [22], with the focus mainly being on YouTube accessed via Android mobile devices or desktop PCs.

In recent years, numerous papers reported on efforts exploiting ML techniques to correlate network-level data with certain application-level degradations [6], [7], [8], [9]. Although the general objectives are similar, these approaches vary in terms of what is predicted by the models (e.g., quality class [6], [8], [9], app-specific KPIs [7], [3], [15]), and on what time-scale (e.g., per flow [6], per video-session [7], [8], [9], in "real-time" per set time window [23], [4], [3]). As stalling was identified to be the most degrading QoE influence factor [24], there are research efforts focused only on buffer state prediction, with the aim to detect low buffer conditions and take actions to prevent stalling [23], [25]. However, the majority of approaches listed above use information from TCP headers, and, in their current state, are not applicable for QUIC traffic (exceptions being [3], [4]). To the best of our knowledge, this is the first study which addresses ML-based QoE estimation from encrypted network traffic (QUIC) focusing on iOS as a measurement platform.

In the paper, we first provide a high-level view of the approach we have been using in our recent studies [26], [15], emphasizing the evolution of the approach driven by both our previous findings and by changes deployed in the YouTube service (Section II). Next, in Section III we present the methodology used in this study, including measurement procedure, collected datasets, and QoE/KPI classification objectives. Finally, Section IV provides an analysis of collected datasets and results in terms of performance of trained classifiers. The key contributions and findings of the paper may be summarized as follows:

- Given the gap in research addressing QoE/KPI monitoring of YouTube delivered to iOS devices, we propose a setup for collecting ground truth application-layer KPIs on iOS devices running the native YouTube App.
- Using a rigorous experimental setup, we collect a dataset of YouTube videos streamed over various emulated network conditions in a lab WiFi network. We train various ML models that classify KPIs (stalling occurrence, resolution played, initial delay, video bitrate) and overall QoE per video session, achieving accuracies ranging between 70 and 92%. Given that all videos in our datasets are delivered via QUIC, our feature selection approach focuses only on IP-level statistics, and is thus applicable for both TLS and QUIC traffic.
- We validate trained models using two additional datasets: one collected in an operational mobile network in Europe, and one collected in an operational mobile network in Asia. Our research goal was to evaluate to what extent models trained in a WiFi lab environment are applicable in a mobile network, with promising results achieved in terms of prediction accuracy.
- To investigate the need for training separate models for different platforms, we train models using a dataset collected on an Android device, and test these models on our iOS dataset. Results show that despite the fact that relatively high classification accuracy can be achieved in certain cases, for KPIs such as stalling occurrence and video bitrate, there is a likely need for training different models across different platforms.

II. Approach

The idea behind our approach is to use ML algorithms to train models able to predict overall QoE and various KPIs of video streaming sessions based solely on the analysis of encrypted network traffic. The approach includes the following steps: 1) collection of network-level data, as well as ground truth application-level performance data under a variety of conditions, 2) processing of collected data and preparing a dataset for ML, and 3) model training and evaluation.

A. Data collection

Application-level video playback information and network-level traffic are captured simultaneously. While network traffic capturing is easily performed using available tools, instrumentation of application-level performance measurement tools requires more effort. For YouTube, over the course of our previous studies we developed and compared different performance measurement tools [15]: YouQ_IF for Android and iOS, that rely on the YouTube IFrame API, YouQ_AA, developed using the YouTube Android API, and our newest tool called ViQMon, that relies on OCR techniques to extract YouTube performance data from the official YouTube app's *Stats for Nerds* window. In the context of iOS, we found that using YouQ_IF resulted in **significantly different player behavior** when compared to using the official YouTube app (i.e., no quality adaptation was observed), so we resorted to using ViQMon in our measurements.

To enable ML models to detect QoE/KPI degradations by learning from examples, there is a need to collect measurements under a variety of network conditions that induce such degradations. For that reason, in our measurement setup we emulate variable network conditions so we can observe a range of possible values of application KPIs. In earlier measurements we defined a variety of bandwidth envelopes [8], while in later measurement efforts we emulated realistic conditions by replaying bandwidth levels measured in an operational network with one second precision [15]. The latter approach is used in the study described in this paper.

B. Data processing

Data processing includes the calculation of statistical properties of the captured network traffic (i.e., traffic features such as throughput in 5 s intervals), calculation of KPIs and/or QoE from application-level logs, and synchronizing and combining the two to form a dataset for training ML classifiers. Data processing in our research has evolved in multiple aspects over the last few years. Besides changes in KPI calculation set off by changes in app-level performance measurement tools, there were also changes in QoE calculation, as we moved from a QoE model derived based on research studies available at the time [27], [28], [29], [30], [24] to the QoE model standardized in ITU-T Recommendation P.1203 [31]. On the network-level, changes in traffic feature calculation were needed when YouTube changed its transport protocol from TCP to QUIC. While with TLS/TCP, features such as number of retransmissions can be extracted, this data is encrypted in QUIC, and thus only throughput-based features can be extracted.

C. Model training and evaluation

A prepared dataset for each of the video streaming instances (videos) contains a list of network traffic features labelled with a ground truth target variable value (e.g., "short" or "long" initial delay). Using feature selection methods, a relevant subset of features needs to be found for each target variable and algorithm, to reduce computational cost and noise from irrelevant features. After feature selection, we train the models and assess their performance through 10-fold cross validation, but also using separate validation datasets.

III. Methodology

Providing more detailed insight into our test methodology, we first describe the measurement setup and data collection, then we summarize the information about datasets used in this paper, and describe how these datasets were used to train and test QoE/KPI classification models.

A. Measurement procedure

The laboratory testbed used for conducting experiments in the lab is depicted in Figure 2. A client device (iOS: iPhone 6s) runs the official YouTube app and a screen recording app. YouTube traffic between the client device and YouTube content servers is transmitted over an IEEE 802.11n wireless network and then routed through a PC running IMUNES[1], a general purpose IP network emulation/simulation tool enabling a test administrator to set up different bandwidth limitations and schedule bandwidth changes. Traffic is further sent through Albedo's Net.Shark device[2] where it is replicated and sent to a PC designated for network traffic capturing. The router runs on 2.4 Ghz frequency with more than 50 Mbit downlink.

To emulate realistic bandwidth limitations, we scripted the IMUNES emulator to use the 4G/LTE bandwidth logs[3] published in [32]. As bandwidth was generally high in the logs, we divided the values in the log with the factors of 10, 20 and 30, thus creating four bandwidth envelopes in total. To collect datasets D1 and D4, we played 100 YouTube videos over each of the four bandwidth envelopes. The exact number of videos in the final dataset is slightly smaller than 400 in each of the datasets, due to videos being removed or country restrictions changed over the course of the data collection period.

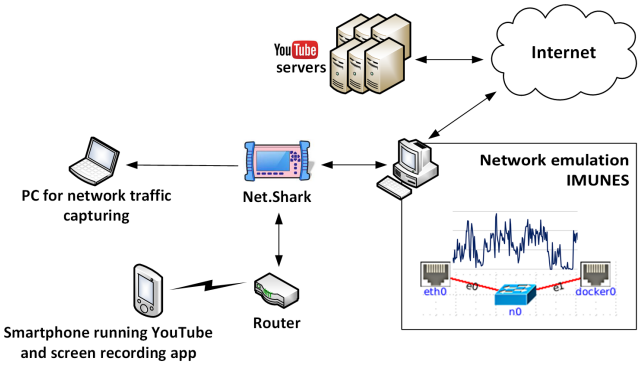


Fig. 2: Laboratory setup.

Our mobile network testbed consisted of an iPhone 6s running a screen recording app while playing YouTube videos, and a network probe running on the Gn interface for WCDMA, and S11 and S1U interfaces for LTE. As there was no way to limit the bandwidth in this scenario and the area covered by the probe has a very good 4G coverage, to induce quality degradations, we resorted to limiting the client device to only connect to 2G or 3G (dataset D2) and to limiting signal strength by using a mechanical barrier (dataset D3).

Screen recordings captured at client devices during measurements are later processed using the ViQMon tool, that uses OCR techniques to read data from the SFN overlay, available in the YouTube official app. Information offered in the SFN overlay includes parameters such as video ID and buffer state. From these parameters we can calculate various KPIs, and also MOS (Mean Opinion Score, corresponding to "overall QoE") according to ITU-T Recommendation P.1203 [31] using the implementation available online[4] [33]. We note that the model from ITU-T Recomm. P.1203 has been validated for H.264 video, while for VP9 (which we often observed in YouTube) we used the Non-Standard Codec Extension for ITU-T P.1203[5].

B. Dataset summary

A summary of characteristics of collected datasets is given in Table I. Three of the datasets were collected on an iOS device (D1-D3), two of which in a mobile network scenario (D2: European ISP, D3: Asian ISP). All three datasets are novel and have not been previously reported. In this paper we

[1] http://imunes.net/
[2] https://www.network-testers.com/albedo-net-shark.html
[3] http://users.ugent.be/~jvdrhoof/dataset-4g/
[4] https://github.com/itu-p1203/itu-p1203
[5] https://github.com/Telecommunication-Telemedia-Assessment/itu-p1203-codecextension

TABLE I: Characteristics of collected datasets.

Dataset	Platform	Collection time period	Access network	No. of videos
D1	iOS	Jun. 2018	WiFi	383
D2	iOS	Sept. 2017	Mobile (European ISP)	128
D3	iOS	Mar. 2018	Mobile (Asian ISP)	111
D4	Android	Jan. 2018	WiFi	394

also use one of our previously collected datasets involving an Android device (D4)[6].

Each of the datasets, for each of the video streaming instances contains calculated network traffic features and target labels i.e., ground truth classes. Network traffic features were calculated for the whole playback duration and include:

- averageThroughput{DL/UL} - average throughput,
- avgPacketSize{DL/UL} - average packet size,
- percOfUsedTransTimeDL - percentage of used transmission time (time window is considered to be "used" for transmission if all interarrival times are shorter than 0.1 s) [4],
- numOfPacketsLarger100BUL - number of uplink packets larger than 100 Bytes (the condition eliminates ACKs, thus counting segment requests) [4],
- avgSizeLarger100BUL, stdSizeLarger100BUL - average and standard deviation of uplink packet sizes, counting only the packets larger than 100 Bytes [4],
- {avg/hMean/median/min/max/stDev} SizeIn{5/3/2/1s} Intervals{DL/UL} - average, harmonic mean, median, min, max, standard deviation of volumes of downlink and uplink traffic in 5,3,2 and 1s-intervals.

We also list the 11 target variables used to classify each of the video instances, with corresponding class values:

1) MOS: "high" $\geq 4 >$ "medium", $\geq 3 >$ "low",
2) MOS: "high" $\geq 3.5 >$ "low",
3) Resolution: "high" $\geq 720p >$ "medium" $\geq 360p >$ "low",
4) Resolution: "hd" $\geq 720p >$ "sd",
5) Stalling Occurrence: "y" (yes), "n" (no),
6) Stalling Ratio (ratio of stall time and overall video playback time): "low" $\leq 0.05 <$ "medium" $\leq 0.2 <$ "high",
7) Stalling Ratio: "low" $\leq 0.1 <$ "high",
8) Initial delay: "short" $= 0\ s <$ "medium" $\leq 5\ s <$ "long",
9) Initial delay: "short" $\leq 1\ s <$ "long",
10) Video bitrate: "high" $\geq 2000\ kbps >$ "medium" $\geq 1000\ kbps >$ "low",
11) Video bitrate: "high" $\geq 1500\ kbps >$ "low".

We note that initial delays measured to be zero are due to the fact that SFN data is refreshed at a 1 sec granularity level, thus this value is in fact less than 1 sec.

C. QoE/KPI classification objectives

We used dataset D1 (iOS, WiFi) to train 7 types of classification models (out of 11 listed in previous section) using 3 ML algorithms. Some classifications were omitted due to dataset characteristics (low number of instances in certain classes, Section IV-A). The 3 used algorithms are OneR (simple rule-based algorithm), J48 (decision tree), and Random Forest (a number of decision trees "voting" for a class). Model performance was assessed through 10-fold cross validation.

Referring back to Figure 1, we note that data collection requires the collection of traces involving various QoE degradation scenarios. Given that instrumenting this process in a lab environment requires less effort when compared to field data collection in a mobile network, we tested to what extent models trained on data from a lab WiFi network are applicable to traffic collected from operational mobile networks. For this purpose we tested D1-trained models on D2 and D3.

Finally, we tested the applicability of D4-based models (Android, WiFi) on D1 (iOS, WiFi). Besides the obvious motivation for this being the convenience of using the same models across different platforms, we clarify another motivational point. ViQMon as a method for application-level performance measurement relies on extraction of information from video frames, which requires time and processing power. There is an alternative for Android which avoids video processing by using a Wrapper application [17] that copies SFN info into a file. Being able to use the same models for iOS and Android would thus reduce time and effort needed for model training and re-evaluation.

IV. Results

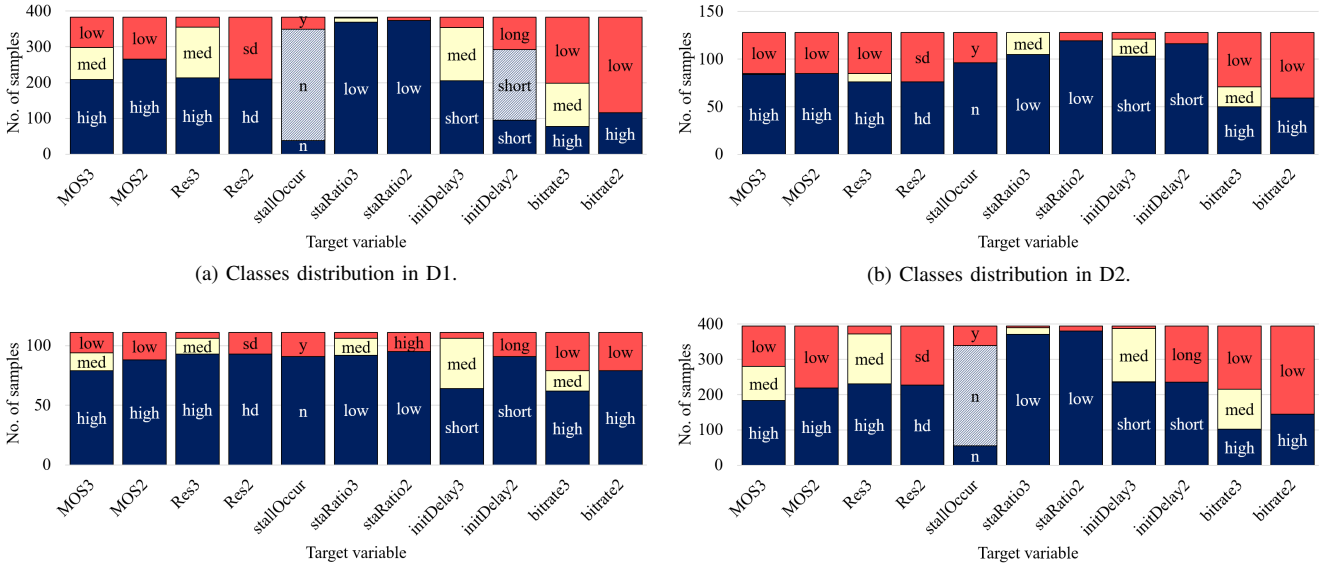
A. Dataset analysis

We briefly analyse collected datasets, as certain observations are relevant for assessing the performance of ML models. Charts in Figure 3 show the number of instances classified into each of the classes, for each of the target variables, and for each of the four datasets. Additionally, we provide CDFs of MOS values for all of the datasets, to depict the range of QoE degradations captured in our datasets (Figure 4).

In datasets D1 and D4 (WiFi), we observe a variety of MOS values, which resulted in a moderately balanced dataset with respect to MOS classifications. On the other hand, D2 is rather imbalanced. We see that MOS was either very low or very high, which results in the lack of instances in the MOS class "medium", but also in potentially too optimistic results when testing the models on this dataset. With D3 we captured a more balanced dataset in terms of MOS classification. In both WiFi datasets, "low" resolution (144p, 240p) was rarely observed. This is why we focused only on binary classification of resolution in model training. In dataset D2, we observe lack of middle resolution instances, while in dataset D3 resolution was generally high.

Stalling events were rare in all of the datasets. Due to the lack of samples with stalling events, we only train the models that predict if there was any stalling (yes/no). Even in that case, we had to balance out the dataset by undersampling the dominant class before model training. Initial delays were generally short across all of the datasets, and when labelling instances into initial delay classes, we set rather low values of

[6]D1 and D4 are available at https://muexlab.fer.hr/muexlab/research/datasets



(a) Classes distribution in D1.

(b) Classes distribution in D2.

(c) Classes distribution in D3.

(d) Classes distribution in D4.

Fig. 3: Distribution of video instances into classes. Given that for some target variables datasets were imbalanced, models were either not trained for these target variables, or instances were subset to balance out the dataset (pattern filled bars).

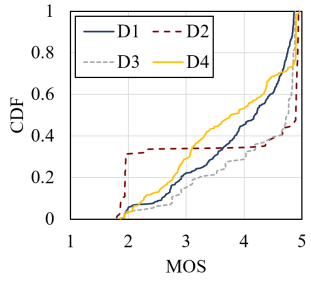


Fig. 4: MOS distributions across datasets.

initial delay to be the boundaries between classes. We therefore classify initial delay only into 2 classes to test the feasibility of initial delay classification, but these classes do not necessarily represent good and bad quality. With respect to bitrate, all of the datasets had a fair distribution of instances among classes.

B. Performance of classification models

We trained 42 different ML based classifiers: using 2 separate datasets (D1 and D4), 7 different prediction targets (MOS classified into 2 and 3 classes, longest resolution into 2 classes, stalling occurrence into 2 classes, initial delay into 2 classes, and video bitrate into 2 and 3 classes), and 3 different ML algorithms (OneR, J48 and RF). We test the model performance through 4 scenarios.

We first assess the performance of D1-trained models using 10-fold cross-validation. Table II summarizes the results in terms of achieved global accuracy, and precision and recall per class. Due to the page limitation, we list this information only for the best performing ML algorithm for each of the target variables[7]. From the Table, it can be observed that video bitrate is the most accurately predicted target (up to 91.91%), which seems logical, as it directly impacts the throughput. Most of the other targets can also be classified with more than 80% accuracy. With MOS classified into 3 classes, accuracy is just slightly lower (77.02%), which is expected due to a finer classification. The lowest is the prediction accuracy of initial delay, 70.43%.

When considering an ISP perspective, the aim may be to take action in cases when significant QoE degradations are estimated. In such cases, models with higher recall values in "low" and "medium" classes, and higher precision values in the class "high" are more appropriate.

In the second and the third test, we assessed the performance of D1-trained models on datasets from mobile networks (D2 and D3) (Table II). On D2 we achieved extremely good results, which can, to some extent, be attributed to D2 not being very balanced (the quality was either very good or very bad). We however note two problematic results. When classifying MOS into 3 classes, precision and recall on the class "medium" are equal to 0. This is because there was only one instance in MOS class "medium" in D2, and it was misclassified.

As dataset D3 is more balanced, the results of testing D1-trained models on D3 may be more realistic (Table II). Video bitrate is very accurately predicted, while with other targets we observe that the model accurately detects high-quality instances, while detection of low-quality instances should be improved. As collecting data for model training in a mobile environment requires significantly more effort, we are motivated to adjust the models trained on data from a

[7]Due to space limitations, details on selected attributes are omitted, but are available at https://muexlab.fer.hr/muexlab/research/datasets.

TABLE II: Performance of different ML based models considering different targets.

	10-fold cval on D1			D1-trained models on D2			D1-trained models on D3			D4-trained models on D1		
Target variable	**Acc. [%]**	**Prec.**	**Rec.**	**Acc. [%]**	**Prec.**	**Rec.**	**Acc. [%]**	**Prec.**	**Rec.**	**Acc. [%]**	**Prec.**	**Rec.**
MOS {high:h, medium:m, low:l}	OneR **77.02**	h: 0.92 m: 0.64 l: 0.71	h: 0.77 m: 0.75 l: 0.81	RF **98.44**	h: 0.99 m: 0.00 l: 0.98	h: 0.99 m: 0.00 l: 1.000	OneR **75.67**	h: 0.87 m: 0.00 l: 0.45	h: 0.89 m: 0.00 l: 0.82	RF **72.06**	h: 0.85 m: 0.57 l: 0.59	h: 0.80 m: 0.47 l: 0.78
MOS {high:h, low:l}	RF **84.59**	h: 0.88 l: 0.75	h: 0.89 l: 0.73	J48 **99.22**	h: 1.00 l: 0.98	h: 0.99 l: 1.00	RF **79.28**	h: 0.82 l: 0.50	h: 0.95 l: 0.17	RF **83.55**	h: 0.94 l: 0.68	h: 0.82 l: 0.88
Longest resolution {hd, sd}	J48 **83.28**	hd: 0.88 sd: 0.78	hd: 0.80 l: 0.87	J48 **91.41**	hd: 0.88 sd: 0.98	hd: 0.99 sd: 0.81	RF **86.49**	hd: 0.87 sd: 0.80	hd: 0.99 sd: 0.22	RF **78.85**	hd: 0.77 sd: 0.82	hd: 0.88 sd: 0.68
Stalling occurrence {yes, no}	J48 **80.55**	y: 0.83 n: 0.79	y: 0.73 n: 0.87	OneR **90.62**	y: 0.73 n: 1.00	y: 1.00 n: 0.87	RF **86.49**	y: 0.87 n: 0.80	y: 0.99 n: 0.22	RF **73.89**	y: 0.17 n: 0.94	y: 0.50 n: 0.76
Initial delay {short:s, long:l}	RF **70.43**	s: 0.70 l: 0.71	s: 0.74 l: 0.67	OneR **67.19**	s: 0.99 l: 0.21	s: 0.65 l: 0.92	J48 **75.67**	s: 0.85 l: 0.32	s: 0.86 l: 0.30	J48 **76.24**	s: 0.76 l: 0.00	s: 1.00 l: 0.00
Video bitrate {high:h, medium:m, low:l}	RF **87.99**	h: 0.86 m: 0.82 l: 0.92	h: 0.81 m: 0.83 l: 0.94	J48 **87.50**	h: 0.87 m: 0.67 l: 0.94	h: 0.98 m: 0.57 l: 0.89	OneR **86.49**	h: 0.84 m: 0.64 l: 1.00	h: 0.93 m: 0.41 l: 0.97	OneR **75.46**	h: 0.68 m: 0.62 l: 0.95	h: 0.91 m: 0.73 l: 0.71
Video bitrate {high:h, low:l}	RF **91.91**	h: 0.90 l: 0.93	h: 0.83 l: 0.96	J48 **93.75**	h: 0.93 l: 0.94	h: 0.93 l: 0.94	RF **91.89**	h: 0.99 l: 0.79	h: 0.90 l: 0.97	RF **88.77**	h: 0.78 l: 0.94	h: 0.88 l: 0.89

lab network to work on data from mobile network. To see why recall was low on low-quality instances in this test, we plan on collecting more data in a lab under low bandwidth conditions, adding it to D1, and redoing the test. Additionally, if the problem is not in the lack of low-quality instances used for training, another idea is to further analyse the distributions of network traffic features with respect to target variables in WiFi and mobile scenarios and see if the model parameters can be adjusted to address the difference in service behavior.

Finally, in the fourth test we assess the applicability of D4-trained models (Android) on D1 (iOS) (Table II). For most of the targets, results show only a slight decrease in prediction accuracy, when compared to the results from the first test. We detect significantly worse performance in the classification of stalling and initial delay, which may be due to the differences in Youtube's buffering behavior on the two platforms. The exact reasons and possible mitigations will be addressed in future work. However, the test shows that most of the models trained on data collected using an Android device could also be applied on iOS data, thus reducing the effort in terms of collecting separate datasets for these two platforms.

V. Conclusions

In this paper we presented an approach for estimating QoE and various KPIs for YouTube videos from encrypted network traffic, while focusing on iOS devices, and thus going beyond state-of-the-art approaches which focus primarily on desktop clients or Android-based mobile clients.

We analysed 4 collected datasets. Using the data collected in a WiFi lab network, we trained ML models that classify QoE and various KPIs of video streaming sessions on the iOS platform. Model performance was first assessed through cross-validation, and then tested on 2 additional datasets collected in mobile networks to evaluate the applicability of WiFi-trained models. We show promising results in all 3 of these tests, with prediction accuracies, depending on the classification target, ranging from roughly 70 to 90%. We also test the models trained on data from an Android platform on data from an iOS platform, both collected in a WiFi environment. The results show that some of the models are highly applicable across platforms, while on others we observe limited applicability.

With respect to the measurement methodology, the models trained and analyzed in this paper focus on monitoring KPIs and QoE on a per video session level, following ITU-T Recommendations when monitoring "overall MOS", and providing valuable insights for network planning purposes. Further rationale lies also in the fact that overall user satisfaction is impacted by the interplay of multiple KPIs (or QoE metrics) over the course of a viewing session. In future work, we plan to also extend this approach towards "real-time" monitoring, whereby QoE/KPI estimations would be performed on a finer grained level (e.g., in 10- or 20-second time windows) so as to facilitate the mitigation of QoE impairments using techniques such as adaptive resource allocation. While recent studies have already started looking into such approaches and provide valuable insights [3], [23], [4] challenges still remain in extending these findings to monitoring of multiple KPIs, monitoring across multiple platforms, and inferring root causes of impairments to drive effective QoE management decisions.

Acknowledgment

This research has been partially supported by the QoMoVid project funded by Ericsson Nikola Tesla, Croatia; the Croatian Science Foundation under the project UIP-2014-09-5605 (Q-MANIC); and by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS). The authors would like to thank Ivan Bartolec and Marko Jurković for their help in developing the tools used in this work.

REFERENCES

[1] "Cisco VNI: Forecast and Methodology, 20172021," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html, accessed: 2017-06-28.

[2] J. Iyengar and M. Thomson, "Quic: A udp-based multiplexed and secure transport," *draft-ietf-quic-transport-08*, 2017.

[3] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for https and quic," in *Proc. of IEEE INFOCOM 2018*, pp. 1–9.

[4] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying flows and buffer state for youtube's http adaptive streaming service in mobile networks," in *Proc. of ACM Multimedia Systems Conf. (MMSys 2018)*, June 2018.

[5] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *Multimedia (ISM), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 494–499.

[6] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, and H. Yan, "Prometheus: Toward Quality of Experience Estimation for Mobile Apps from Passive Network Measurements," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014, p. 18.

[7] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proceedings of the 2016 ACM on Internet Measurement Conference*. ACM, 2016, pp. 513–526.

[8] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "A machine learning approach to classifying YouTube QoE based on encrypted network traffic," *Multimedia tools and applications*, vol. 76, no. 21, pp. 22 267–22 301, 2017.

[9] P. Casas, A. D'Alconzo, F. Wamser, M. Seufert, B. Gardlo, A. Schwind, P. Tran-Gia, and R. Schatz, "Predicting QoE in Cellular Networks Using Machine Learning and In-smartphone Measurements," in *Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on*. IEEE, 2017, pp. 1–6.

[10] 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 *Proc. of EuCNC 2015*. IEEE, 2015, pp. 239–243.

[11] A. Schwind, M. Seufert, Ö. Alay, P. Casas, P. Tran-Gia, and F. Wamser, "Concept and implementation of video qoe measurements in a mobile broadband testbed," in *Network Traffic Measurement and Analysis Conference (TMA), 2017*. IEEE, 2017, pp. 1–6.

[12] H. Nam, H. Schulzrinne, H. Nam, K.-H. Kim, H. Schulzrinne, M. Varela, H. Nam, H. Schulzrinne, T. Mäki, H. Nam *et al.*, "Youslow: What influences user abandonment behavior for internet video?" *Tech. report*, 2017.

[13] "Ericsson NPT application." [Online]. Available: https://play.google.com/store/apps/details?id=com.ericsson.mbbmeasurement

[14] "V3D." [Online]. Available: https://www.v3d.fr

[15] I. Orsolic, M. Suznjevic, and L. Skorin-Kapov, "YouTube QoE Estimation from Encrypted Traffic: Comparison of Test Methodologies and Machine Learning Based Models," in *Proc. of 10th International Conference on Quality of Multimedia Experience (QoMEX 2018)*. IEEE, 2018, pp. 1–6.

[16] G. Szabo, S. Rácz, S. Malomsoky, and A. Bolle, "Potential Gains of Reactive Video QoE Enhancement by App Agnostic QoE Deduction," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–7.

[17] M. Seufert, B. Zeidler, F. Wamser, T. Karagkioules, D. Tsilimantos, F. Loh, P. Tran-Gia, and S. Valentin, "A Wrapper for Automatic Measurements with YouTubes Native Android App," in *To appear in Network Traffic Measurement and Analysis Conference*, 2018, pp. 1–6.

[18] R. Alshareef and L. Sun, "Performance analysis of youtube video streaming for different mobile devices," *Advances in Communications, Electronics, Networks, Robotics and Security Volume 13*, vol. 13, p. 17, 2016.

[19] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A comparative study of android and ios for accessing internet streaming services," in *International Conference on Passive and Active Network Measurement*. Springer, 2013, pp. 104–114.

[20] J. Añorga, S. Arrizabalaga, B. Sedano, J. Goya, M. Alonso-Arce, and J. Mendizabal, "Analysis of YouTubes Traffic Adaptation to Dynamic Environments," *Multimedia Tools and Applications*, pp. 1–24, 2017.

[21] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hossfeld, "Modeling the YouTube Stack: From Packets to Quality of Experience," *Computer Networks*, 2016.

[22] A. Mondal, S. Sengupta, B. R. Reddy, M. Koundinya, C. Govindarajan, P. De, N. Ganguly, and S. Chakraborty, "Candid with youtube: Adaptive streaming behavior and implications on data consumption," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2017, pp. 19–24.

[23] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "BUFFEST: Predicting Buffer Conditions and Real-time Requirements of HTTP(S) Adaptive Streaming Clients," in *Proceedings of the 8th ACM on Multimedia Systems Conference Pages (MMSys 2017)*, 2017.

[24] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial Delay vs. Interruptions: Between the Devil and the Deep Blue Sea," in *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. IEEE, 2012, pp. 1–6.

[25] D. Tsilimantos, T. Karagkioules, A. Nogales-Gómez, and S. Valentin, "Traffic Profiling for Mobile Video Streaming," *arXiv preprint arXiv:1705.08733*, 2017.

[26] I. Orsolic, L. Skorin-Kapov, and M. Suznjevic, "Towards a Framework for Classifying YouTube QoE Based on Monitoring of Encrypted Traffic," in *International Young Researcher Summit on Quality of Experience in Emerging Multimedia Services (QEEMS 2017)*, 2017.

[27] P. Casas, M. Seufert, and R. Schatz, "YOUQMON: A System for Online Monitoring of YouTube QoE in Operational 3G Networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 2, pp. 44–46, 2013.

[28] D. Ghadiyaram, A. C. Bovik, H. Yeganeh, R. Kordasiewicz, and M. Gallant, "Study of the Effects of Stalling Events on the Quality of Experience of Mobile Streaming Videos," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*. IEEE, 2014, pp. 989–993.

[29] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[30] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience," in *Data Traffic Monitoring and Analysis*. Springer, 2013, pp. 264–301.

[31] Telecommunication standarization sector of ITU, "Parametric Bitstream-based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services over Reliable Transport," ITU, Tech. Rep. P.1203, 2017.

[32] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[33] A. Raake, M.-N. Garcia, W. Robitza, P. List, S. Gring, and B. Feiten, "A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. Erfurt: IEEE, May 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7965631/