IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 1, 2020, accepted April 9, 2020, date of publication April 20, 2020, date of current version May 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2988735

A Framework for In-Network QoE Monitoring of Encrypted Video Streaming

IRENA ORSOLIC, (Graduate Student Member, IEEE), AND LEA SKORIN-KAPOV, (Senior Member, IEEE)
University of Zagreb, Faculty of Electrical Engineering and Computing, 10000 Zagreb, Croatia

Corresponding author: Irena Orsolic (irena.orsolic@fer.hr)

This work was supported in part by the Croatian Science Foundation under Project IP-2019-04-9793 (Q-MERSIVE), and in part by the QoMoVid Project funded by Ericsson Nikola Tesla, Croatia.

ABSTRACT With the amount of global network traffic steadily increasing, mainly due to video streaming services, network operators are faced with the challenge of efficiently managing their resources while meeting customer demands and expectations. A prerequisite for such Quality-of-Experience–driven (QoE) network traffic management is the monitoring and inference of application-level performance in terms of video Key Performance Indicators (KPIs) that directly influence end-user QoE. Given the persistent adoption of end-to-end encryption, operators lack direct insights into video quality metrics such as start-up delays, resolutions, or stalling events, which are needed to adequately estimate QoE and drive resource management decisions. Numerous solutions have been proposed to tackle this challenge on individual use-cases, most of them relying on machine learning (ML) for inferring KPIs from observable traffic patterns and statistics. In this paper, we summarize the key findings in state-of-the-art research on the topic. Going beyond previous work, we devise the concept of a generic framework for ML–based QoE/KPI monitoring of HTTP adaptive streaming (HAS) services, including model training, deployment, and re-evaluation. Components of the framework are designed in a generic way, independent of a particular streaming service and platform. The methodology for applying different framework components is discussed across various use-cases. In particular, we demonstrate framework applicability in a concrete use-case involving the YouTube service delivered to smartphones via the mobile YouTube app, as this presents one of the most prominent examples of accessing YouTube. We tackle both QoE/KPI estimation on a per-video-session level (utilizing the validated ITU-T P.1203 QoE model), as well as ''real-time'' KPI estimation over short time intervals. Obtained results provide important insights and challenges related to the deployment of a generic in-network QoE monitoring framework for encrypted video streams.

INDEX TERMS Quality of Experience (QoE), video streaming, in-network QoE estimation, machine learning, encrypted traffic.

I. INTRODUCTION

With the rapid growth of global mobile Internet traffic, it has become critical to manage network traffic flows in such a way as to provide a satisfactory experience to end users, while efficiently using limited resources. As a prerequisite, both network and service providers rely on the understanding and monitoring of key factors that impact the end users' perceived service quality and experience. This has resulted in extensive Quality of Experience (QoE) related research conducted over the past decade, with the networking community increasingly aiming to introduce QoE-awareness into network management cycles [1].

With the massive amounts of traffic passing through global mobile networks, the greatest share belongs to video content, mainly originating from popular video streaming services and social networks, such as YouTube, Netflix, Amazon Prime, Facebook, and Twitch [2]. According to Cisco's VNI white paper, video accounted for 59% of global mobile traffic in 2017, and the share is expected to grow to 79% by 2022 [3]. Newer numbers, published in the Ericsson Mobility Report, state that 63% of global mobile traffic was video in 2019. The same report is slightly more conservative in its forecasts, predicting the share to grow to 76% by 2025 [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Tariq Umer.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Nevertheless, despite the emergence of new and network-intensive apps such as VR/AR and cloud gaming, especially in the context of 5G, video originating from video streaming platforms (either as Video on Demand or live video) is expected to dominate network traffic in the foreseeable future.

Nowadays, most video streaming services implement the HTTP Adaptive Streaming (HAS) paradigm, commonly in compliance with standards MPEG-DASH (Dynamic Adaptive Streaming over HTTP) [5] and HLS (HTTP Live Streaming) [6]. The main idea behind HAS is to enable dynamic adaptation of video quality (e.g., in terms of bitrate and resolution) according to variable network conditions, so as to primarily avoid playback stalling and ensure a smooth playout experience. This is achieved by storing short chunks of video content on the server, available in different quality levels, and running an adaptation algorithm on the client side responsible for estimating network conditions and requesting each video chunk accordingly. The aforementioned standards define methods and formats, while each compliant service, on top of that, defines its own adaptation algorithm – chunk size, the amount of data that is kept in the buffer, etc. Various employed adaptation strategies have been analysed over the past years [7]–[10], but we note that these findings are susceptible to change and may become outdated, as service providers change their deployed strategies with newer service versions available on the market.

Up until recently, Web encryption efforts were mostly focused on highly sensitive information. Today, with the rise of privacy awareness, we are witnessing encryption in all major video streaming services as well. For example, Google reports that, as of late 2019, all YouTube content is delivered via HTTPS [11]. While YouTube uses mostly UDP/QUIC [12], which was designed to be secure and incorporates TLS by default, services such as Netflix, Twitch, and Facebook Watch use TCP/TLS. However, it is worth mentioning that there are standardization efforts within IETF for a new version of HTTP, HTTP/3, that is aimed to resolve the issues of HTTP/2 by employing QUIC, making the Web inherently secure [13].

Due to the adoption of traffic encryption and the dynamics introduced by HAS, it is becoming increasingly challenging for network operators to monitor service performance at the *application level*, which is crucial when aiming to estimate end users' QoE. The challenge is further complicated by the wide variety of services accessed from different platforms, apps, via different access networks, using different protocols, etc. (Figure 1). As HAS dominates global Internet traffic, there is a lot of interest from the industry to tackle these challenges. In available literature, a number of solutions have been proposed and validated on isolated use-cases, that infer application performance in terms of Key Performance Indicators (KPIs) or overall QoE, solely based on the analysis of encrypted network traffic (Section II). We give a generalized view of these approaches in Figure 2 (adapted from [14]). All solutions require the instrumentation of some sort of

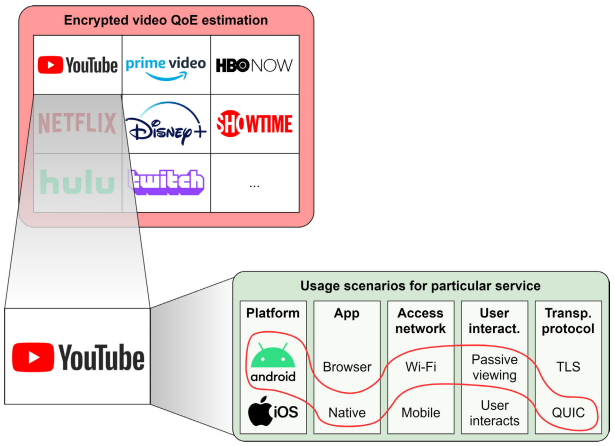


FIGURE 1. An example of the variety of use-cases that may differ in terms of service implementation, and thus in network traffic patterns for video delivery platforms. Consequently, the estimation of QoE/KPIs for these use-cases may need to be addressed separately.

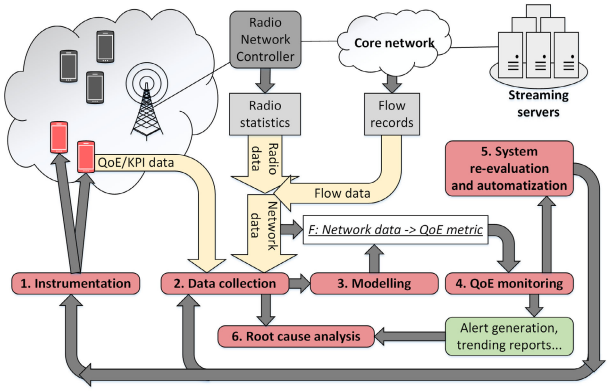


FIGURE 2. Generic service QoE/KPI monitoring approach in the context of traffic encryption (adapted from [14]).

measurement tools used to collect relevant application- and network-layer data. Based on collected data, models that map traffic patterns to QoE/KPIs are built analytically or using machine learning (ML) techniques. In-network measurements (e.g., radio stats, flow data) can be provided in real time to deployed models to calculate desired QoE/KPI 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 causes 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 Over The Top (OTT) service quality adaptation logic, or changes in underlying protocols (e.g., move from TCP/TLS to UDP/QUIC).

In this paper, we consolidate ideas presented in related work that proposed solutions for ML–based in-network QoE/KPI estimation for HAS, and we build upon them in multiple aspects. The contributions of this paper are summarized as follows:

- We give an overview of state-of-the-art monitoring approaches, key challenges, and opportunities for future research.
- We propose a generic, flexible, and extensible framework for in-network QoE/KPI monitoring of encrypted video streams, consisting of components related to *model training*, *deployment*, and *re-evaluation*. Framework applicability is discussed and demonstrated in a case study involving multiple YouTube video on demand datasets and trained models.
- Building on previous work, we present a novel methodology for real-time KPI estimation model training, and compare models of different complexities.
- We further present a methodology for session-level QoE/KPI estimation model training relying on simple packet-size–based features applicable for any HAS use-case, regardless of used protocols and streaming algorithm.
- Finally, we investigate and outline a number of related challenges to be addressed by the research community: training of platform-agnostic models, aggregation of real-time predictions as additional session-level features, detection of model under-performance, and session delimitation (as a prerequisite for session-level QoE/KPI estimation).

The remainder of the paper is organized as follows. A detailed overview of related work is presented in Section II, with selected studies compared in terms of their objectives, datasets, and key findings. Section III presents a generic conceptual QoE/KPI monitoring framework, and discusses key components related to model training, deployment, and re-evaluation. In Section IV we present our collected YouTube datasets (comprised of both application and network layer data), and highlight specific research questions which we aimed to answer with each of the datasets. Our real-time KPI estimation model training methodology and results in terms of model performance are given in Section V. Section VI focuses on session-level QoE/KPI estimation and related questions. A discussion of open research challenges and concluding remarks are given in Section VII.

II. BACKGROUND AND RELATED WORK

Prior to the widespread use of traffic encryption, in-network QoE monitoring solutions relied on deep packet inspection (DPI) for extracting information about video quality (e.g., streamed resolution, codecs, bitrate) [28], [29]. With the adoption of encryption, such solutions were for the most part no longer viable, which opened up new research questions related to in-network QoE estimation based on the analysis of encrypted video traffic.

On a high level, two different approaches have been proposed to tackle this problem: *session-modeling–based* (SM) and *machine-learning–based* (ML). SM–based solutions require knowledge about the streaming protocol and rely on session reconstruction for the inference of QoE–influencing KPIs. Mangla *et al.* present such a solution called eMIMIC in [30], highlighting the performance it achieves in comparison to ML–based solutions. The solution is based on reconstructing the chunk–based delivery sequence of a video session from packet traces, and then the use of this sequence to model a video session and estimate average bitrate, re-buffering ratio, bitrate switches, and start-up-time. The presented system would need to be adapted to work with QUIC traffic, potentially reducing chunk-detection performance which the system is based upon. Although in [31] Krishnamoorthi *et al.* do not estimate QoE/KPIs as such, but rather try to predict buffer conditions by emulating the client buffer, parts of the approach can be applied for this problem as well. However, given the problem dimensionality – resulting from a large variety of devices, platforms, streaming services, apps from which the content is accessed, different network types, and different protocols – finding analytical solutions for a wide range of potential use-cases becomes extremely complex. Moreover, service providers may change the streaming protocol, be it in terms of adaptation strategy, used network protocols or something else, potentially leading to the need for network operators to adapt their QoE/KPI estimaton models. For such reasons, numerous recent studies have turned to utilizing ML techniques for QoE/KPI estimation, arguing that such approaches are potentially more flexible and sustainable in the long run [8], [14], [15], [17], [19], [21], [25], [27].

In both SM and ML approaches, application-level ground-truth data needs to be collected for the learning phase. This can be relatively easy for some services and platforms (such as YouTube viewed on a desktop device or an Android phone), but more difficult for others. Most related work is focused on YouTube, partially due to application-layer data being easily obtainable. Earlier video streaming performance measurement apps, such as first versions of the YoMoApp [32] and YouQ [8], relied on the YouTube IFrame API for embedding YouTube videos and extracting performance data. For Android, there is an alternative in the form of the YouTube Android API, which however does not report on all relevant application events [24]. As studies have shown that these measurement apps do not necessarily behave in the same way as the official native YouTube app [24], new approaches have been proposed that extract data from YouTube's *Stats for Nerds* window, available both in the browser and in the app [24], [33]. A similar performance report window is available for Twitch and Netflix, but only when viewed in the browser. While in SM–based approaches, collecting a smaller dataset for determining the streaming protocol is usually sufficient, in ML–based approaches, collecting a large and varied (in terms of application-layer performance) dataset is crucial. This is why most recent papers report on their data collection automation efforts, employing frameworks such as Selenium [21].

ML–based QoE/KPI estimation solutions proposed in literature are fairly similar in their core idea (Figure 3). Once the dataset is collected, network traffic features are extracted from the network traffic trace, and corresponding desired

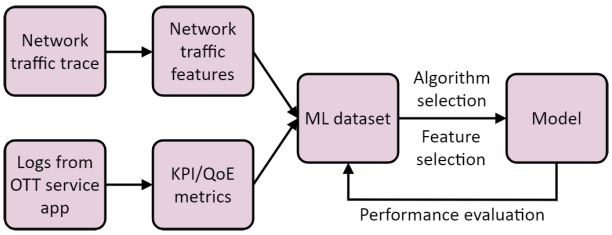


FIGURE 3. Generic approach for ML–based QoE/KPI estimation model training.

KPI/QoE metrics are calculated based on application-level performance data. The combined data is fed to ML algorithms for the purpose of training models that are then able to estimate KPIs/QoE based only on network traffic features. Published papers for the most part differ in terms of: data collection methods, specified features and targets, objectives, and use-cases used for validation. We provide an overview of recent studies on ML–based in-network QoE/KPI estimation for HAS in reverse chronological order in Table 1. The table summarizes the main objectives of these studies (such as session-level vs. real-time QoE/KPI estimation, addressed use-cases, target variables), information on used datasets (e.g., collected in a lab WiFi network vs. mobile network, number of videos in the dataset), and key findings.

While all of the listed papers presented either a novel methodology or addressed new use-cases at the time they were published, the aim of this paper is to bring together these different methods under a **joint framework**, covering all key steps in designing and deploying a solution for in-network QoE monitoring. We propose a conceptual generic and extensible framework for model training, model deployment, and re-evaluation. Through the description of the framework, we address questions related to actual deployment and utilization of models. The framework is defined in a way that allows different levels of automation, ranging from fully automated, in case data collection is also done in an automated manner, to partially automated, if some kind of expert intervention is necessary, such as starting a new measurement campaign to re-evaluate existing models. We validate individual components of the framework on use-cases concerning YouTube video on demand, and address timely challenges related to session-level QoE/KPI and real-time KPI classification, going beyond related work in terms of addressing additional questions relevant for practical applications, thus making connections between otherwise fragmented research.

III. FRAMEWORK OVERVIEW

In this section, we present the concept of a generic and flexible framework for in-network ML–based HAS QoE/KPI monitoring. The idea behind the framework is to integrate and automate the process of QoE/KPI estimation model training, in-network model utilization, detection of model under-performance in light of service behavior changes, and consequent model adaptation.

The framework consists of three main functional components: (1) *Model training*, (2) *Model deployment (in-network QoE/KPI estimation)*, and (3) *Model re-evaluation and adaptation* (Figure 4). Additionally, the *Configuration and orchestration* component provides an interface for an expert to specify parameters to be used for each use-case (e.g., for on-demand YouTube streaming, run real-time video resolution prediction based on a decision tree model trained on the top 10 features selected using the SFS algorithm from a specified feature set). We note that this section describes possible configuration options of the conceptual framework, while parameters that work best in practice for a specific service need to be determined experimentally. However, the following sections describe the implementation of the framework, validated in the case of YouTube, and serve as guidelines that can be applied to other services as well.

A. MODEL TRAINING

The *Model training* component takes network- (flow and, if available, radio) and application-level data as input. First, it extracts network-level features and ground-truth labels, according to the specification for the service in question. The exact features and labels to be extracted are specified by an expert through the *Configuration and orchestration* component, separately for real-time and session-level QoE/KPI estimation. The component itself can be used for the training of both types of models, with what is utilized in practice depending on the configuration. Extracted features and labels are fed to the feature selection algorithm, to eliminate irrelevant features and reduce complexity of the models. The algorithm and algorithm parameters that are to be used in this step are specified through the *Configuration and orchestration* component. Finally, the models are trained using the algorithm and parameters selected in the configuration.

Collected data, that is used as input to the model training component, may vary across different services, but also across different use-cases concerning a single service (e.g., different apps used for accessing the content). The approach we describe in later text is applicable for any use-case, regardless of network-level differences, as it relies solely on packet size information. However, additional information used as predictors, such as TCP-data (where applicable), context-data, or radio-data available to network operators may improve model performance. Ground-truth application-level data extraction is more challenging in the sense that data is not always obtainable. For example, YouTube offers video performance metrics in their *Stats for Nerds* window, both on desktop and mobile devices (in the browser and in the official app), but a similar overlay offered by Twitch is only accessible in the browser (both desktop and mobile) and not in the app.

The exact features and labels to be extracted from collected data depend on the purpose and desired goals. If the goal is to manage the network traffic in real-time, ground-truth data used for training should reflect the conditions that need to be detected (e.g., detecting that multiple users in a network segment experienced a stalling moments ago requires a model to be trained on short time-windows of network traffic labelled

TABLE 1. Overview of selected recent studies addressing machine-learning–based in-network QoE/KPI estimation for adaptive video streaming.

Author (Year)	Objectives	Datasets	Key findings
Bronzino *et al.* (2019) [15]	• Real-time startup delay and resolution estimation • Propose generic models applicable across multiple services (Netflix, YouTube over QUIC and TCP, Amazon and Twitch)	• 5 datasets containing 13765 sessions (Nov. 2017 – May 2019) • 6 laptops in home networks, 5 laptops in the lab network with emulated conditions • Ground-truth data collected through a Chrome extension	• Precise initial delay regression models • Fine-grained resolution classification models on 10-second windows • **Generic models applicable for five service scenarios, if the training data included data from all services**
Schwarzmann *et al.* (2019) [16]	• Session-level MOS estimation (regression) based on **5G monitoring data** • Simulation of a mobile video streaming use-case	• Generated using an *OMNeT++* simulator • Varying simulation parameters, simulation scenarios (e.g., no. of users, their locations), client parameters, and video properties	• First step to understanding the correlation of 5G monitoring data to user QoE
Wassermann *et al.* (2019) [17], [18]	• **Real-time resolution classification**; bitrate considered in [18] • Focus on YouTube in browser • IP-level traffic features in time-windows	• Dataset with more than 15000 videos (Jun. 2018 – Feb. 2019) • Home/corporate WiFi, lab WiFi with Bw emulation, and LTE mobile network • **Data collection automation – Selenium**	• Tree–based models provide highly accurate results and are execution-fast • Highly efficient **stream–based strategy for feature computation**
Gutterman *et al.* (2019) [19]	• Real-time prediction of buffer warning, streaming phase, and video resolution • YouTube in browser • **Inclusion of chunk–based features** derived by the chunk detection algorithm	• Dataset containing over 500 sessions (2018) • 3 different WiFi networks • Static and movement scenarios • Ground-truth data collection through IFrame API	• Estimated **chunk–based features improved prediction accuracy**, when compared to IP-level features only
Bartolec *et al.* (2019) [20]	• Session-level KPI classification (resolution, initial delay, video bitrate) • YouTube for Android • IP-level traffic features • **Consideration of realistic end-user playback–related interactions**	• Dataset without user interactions, 299 videos (2019) • Dataset with user interactions (pause, seek, abandon), 307 videos (2019) • Ground-truth data from *Stats for Nerds* • Lab env. with bandwidth emulation	• Models trained on the dataset that does not include user interactions perform worse on data with user interactions • **Need to include more realistic data, with user interactions**, into the model training phase
Seufert *et al.* (2019) [21], [22]	• Real-time **detection of stalling** • YouTube in browser • IP-level traffic features in time-windows	• 4714 YouTube sessions (2018) • Home/corporate WiFi, lab WiFI with Bw emulation, and LTE mobile network • **Data collection automation – Selenium**	• Promising results for random-forest–based stalling detection in a stream-based **real-time** fashion • Evaluation of relevance of different feature sets in [22]
Orsolic *et al.* (2018) [23]	• Session-level MOS (ITU-T P.1203) and KPI classification (resolution, stalling, initial delay, video bitrate) • IP-level traffic features • **Focus on YouTube via iOS platform**	• 4 datasets from Sept. 2017 – Mar. 2018 • Two datasets collected in the lab (emulated conditions), one on Android (394 videos), one on iOS (383 videos) • Two datasets collected in mobile networks on iOS (128 + 111 videos) • Ground-truth data from *Stats for Nerds*	• Promising performance of models for iOS, also on data from mobile network • **Android–trained models applicable to iOS data, with slight decrease in performance**
Orsolic *et al.* (2018) [24]	• Session-level MOS (ITU-T P.1203) and KPI classification (resolution, bitrate) • YouTube on Android • IP-level traffic features • **Comparison of ground-truth data collection methodologies**	• 3 datasets from 2017 – Jan. 2018 • One collected using three app-level data collection methods (YouTube IFrame API, Android API, *Stats for Nerds* **extraction**; 300 videos) • Two collected using the *SfN* extraction: in a lab network with Bw limitations (394 videos), in a mobile network (105 videos)	• Methodology for *Stats for Nerds* extraction based on screen recording and OCR • **Approaches relying on embedding YouTube videos for test purposes exhibit different characteristics as compared to the official YouTube app**
Mazhar *et al.* (2018) [25]	• **Real-time KPI classification** (initial delay, stalling, resolution) • YouTube in browser • IP-level features (and TCP-level, if applicable)	• Dataset containing 5488 sessions over QUIC and 5375 over TCP (2017) • Data collection automation – Selenium • Lab env. with Bw, delay and packet loss emulation	• Approach for highly accurate decision-tree–based binary classification of KPIs in real-time
Orsolic *et al.* (2016, 2017) [26] [8]	• Session-level QoE classification (custom classes; prior to the publication of ITU-T P.1203) • YouTube on Android • TCP-level traffic features	• Dataset containing 1060 videos (Apr.-Jun. 2016) • Ground-truth data from IFrame–based app called YouQ • Lab env. with Bw emulation	• Detailed methodology – data collection, network conditions, ML analysis • Proved the **feasibility of classifying YouTube video streams into QoE classes**
Dimopoulos *et al.* (2016) [27]	• Session-level classification of stalling, average video quality and quality variations • Focus on YouTube • TCP-level traffic features	• Dataset containing 390000 unique sessions collected at a Web proxy (2016) • Only a small percentage of adaptive streaming • Significant amount of unencrypted streams	• Framework for detecting video streaming KPIs applicable for encrypted traffic • Changes in size and interarrival times of video segments are among the most important indicators of quality impairments

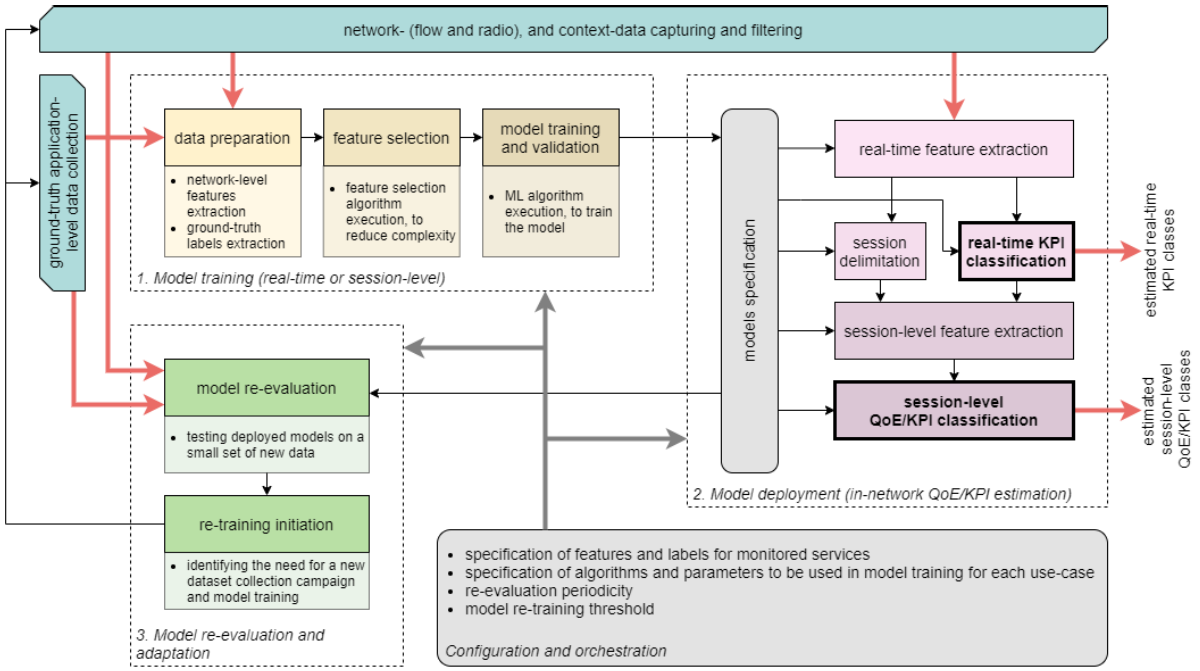


FIGURE 4. Conceptual framework for in-network ML–based QoE/KPI monitoring of HTTP adaptive streaming.

with stalling information). If the goal is to plan and dimension the network to meet users' expectations on a larger scale, estimating QoE on a session-level might be more appropriate. A challenge also lies in identifying network-level features that correlate with these application-level events. Related work on QoE/KPI estimation, discussed in the previous section, provides information on possible network-level metrics that have proven to be promising on selected use-cases.

Popular ML frameworks offer a lot of valuable ready-to-use tools for feature selection and model training. In our work, we relied on the wrapper approach, using sequential feature selection to select relevant features. This method enables the selection of features that work well together, as opposed to methods that evaluate each feature separately (discussed further in Section V). For model training, we found that models trained using tree-based algorithms, such as Decision Tree or Random Forest, in most cases performed best for this purpose [8], [15], [17], [19], [21], [25], [34]. We also argue that using more lightweight models (such as decision trees instead of random forests) may be better in actual in-network applications, as they may require significantly less resources, while potentially only slightly sacrificing the performance.

B. MODEL DEPLOYMENT AND QoE/KPI ESTIMATION COMPONENT

The purpose of the *QoE/KPI estimation component* is to be deployed in the network, and output QoE/KPI estimates on session-level and/or in real-time, based on the network- and context-data input. We note that by 'real-time' we are referring to estimations made across a chosen time interval. While models can be trained to make estimates on a per-second time frame, an operator may decide that making estimations for 5-, 10-, or 20-second intervals may be sufficient. As opposed to QoE/KPI estimations made on a per-session level, such approaches may be utilized for real-time resource allocation and optimization mechanisms.

We assume that the data serving as input for this component has previously been filtered, i.e., we do not explicitly portray traffic classification and the separation of flows corresponding to different users. The component only receives relevant filtered information corresponding to both uplink and downlink data (e.g., packet sizes, timestamps).

The component can continuously track real-time features, output real-time KPI outputs, and detect session starts or ends, to be used for session-level QoE/KPI classification. In our prototype implementation, various network traffic statistics are tracked in 1 s windows for this purpose, but different precisions and additional features may be used as well. Real-time KPIs estimated based on the deployed model can be used as final outputs in this step to take desired actions, such as traffic rerouting or resource reallocation.

Once a session start is detected by the *Session delimitation* module, the *Session-level feature extraction* module starts aggregating session-level traffic features. As soon as the session end is detected, calculated features can be forwarded to the *Session-level QoE/KPI classification* module, which outputs classes of interest, as defined by the model specification. We note that some statistics of the network traffic cannot be calculated in this way (such as median, which requires full state). If such features are necessary, the traffic information needs to be buffered for the whole session. Additionally,

TABLE 2. Summary of measurement campaigns.

Dataset label	Platform	Collection time	Num. of videos	Conditions
And19	Android (Samsung Galaxy S8)	Feb-Mar 2019	299	100 different YouTube videos played in 3 different network conditions: 4G, 3G, and 1.5 Mbps
iOS19	iOS (Apple iPhone 8)	Feb-Mar 2019	299	100 different YouTube videos (same set as in And19) played in 3 different network conditions: 4G, 3G, and 1.5 Mbps
And18	Android (Samsung Galaxy S6)	Jan 2018	394	100 different YouTube videos (different set, compared to 2019 measurements) played in 4 different network conditions: 4G, and 4G divided by factors of 10, 20 and 30

if for a certain use-case both real-time and session-level QoE/KPI estimation is employed, session-level features can be enriched with real-time KPI estimates, for potentially better performing session-level estimation. However, in that case, models need to be trained with those estimates in mind. This idea is briefly explored in Section VI-E, but is omitted in Figure 4 for the sake of simplicity and cleanliness.

C. MODEL RE-EVALUATION AND ADAPTATION COMPONENT

The most intuitive way to check if the *QoE/KPI estimation component* is up-to-date is to check if its performance is comparable to the performance achieved during the model training phase. This requires a data collection campaign, to obtain a dataset labelled with ground-truth. The *Model re-evaluation and adaptation component* initiates a smaller-scale periodical data collection campaign, tests the models and reports on results. The periodicity of re-evaluation is defined in the configuration, and will depend on the observations regarding a certain service. The component can also be configured to automatically start a larger measurement campaign for the purpose of training new models, when such action is required, i.e., if the model is performing worse than the defined threshold.

IV. DATASETS AND ADDRESSED RESEARCH TOPICS

To validate individual components of the framework, we used YouTube data collected across three measurement campaigns.[1] While a number of large scale studies have conducted measurements using fixed network access and desktop browsers [15], [17], [19], [21], [25], less focus has been on measurement campaigns conducted on mobile devices [20], [23], [24], and in particular using the native YouTube app. All three of our campaigns were conducted in a lab environment with emulated network conditions, in order to induce QoE degradations and provide a variety of examples to ML algorithms to learn from and incorporate that knowledge into trained models.

The laboratory environment is depicted in Figure 5. A smartphone playing YouTube videos using the official app is connected to the Internet through a router whose traffic is routed through the IMUNES[2] node, where bandwidth

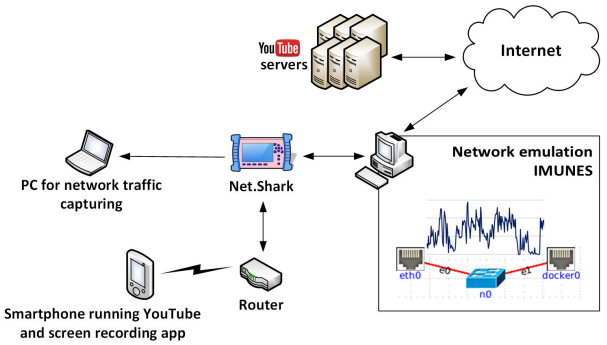


FIGURE 5. Laboratory setup for data collection.

limitations are imposed. All the traffic passing through the router is replicated with an Albedo Net.Shark[3] portable TAP device and captured using tcpdump.[4] In this environment, IMUNES (a freely available general purpose IP network emulation/simulation tool) was scripted to imitate 4G[5] [35] and 3G[6] [36] bandwidth, as provided in publicly available measurement logs, and limit the bandwidth to fixed values. In our previous work, we also performed measurements in operational mobile networks [23], [24], but collecting a varied dataset is of utter importance to train robust models, and is much easier to obtain in the lab environment. For obtaining the ground-truth application-level data, we recorded the screen while YouTube's *Stats for Nerds* window was enabled and displayed, and extracted the performance data using Optical Character Recognition (OCR). We note that there is a simpler approach able to directly extract text from *Stats for Nerds* [33], but is currently not applicable for the iOS platform. Table 2 summarizes the details on three aforementioned measurement campaigns. The total number of videos across all three datasets is roughly 1000. While automation frameworks such as Selenium have enabled large scale measurements presented in related work [15], [17], [19], [21], [25], these can only be applied if viewing content in the browser. We are aware of mobile automation frameworks such as Appium[7] and plan to employ these in the future.

[1] Select datasets are available at https://muexlab.fer.hr/muexlab/research/datasets.

[2] http://imunes.net/

[3] http://www.albedotelecom.com/pages/fieldtools/src/netshark.php

[4] https://www.tcpdump.org/

[5] http://users.ugent.be/~jvdrhoof/dataset-4g/

[6] http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/

[7] http://appium.io/

In the following sections of the paper, we validate separate parts of the framework. Here we provide a summary of addressed research topics and used datasets, and map them to the sections of the paper:

1) *Real-time KPI classification model training and validation.* We present a methodology devised for real-time KPI classification, and validate it using the dataset And19 for the classification of video resolution and bitrate. (Section V)
2) *Session-level QoE/KPI classification model training and validation.* We present a methodology devised for session-level QoE/KPI classification, and validate it separately using datasets And19 and iOS19 for the classification of MOS (Mean Opinion Score), video resolution, and bitrate. (Section VI-B)
3) *Session delimitation.* A prerequisite for session-level QoE/KPI classification is detecting session start and end times. We present our preliminary work addressing this problem in Section VI-A, using dataset And19 in the analysis.
4) *Model generalization.* Given the variety of possible service usage scenarios, training models able to address QoE/KPI estimation for multiple use-cases is of great value. We present our results in that direction in Section VI-C, using datasets And19 and iOS19.
5) *Model re-evaluation.* To emphasize the need for periodic model re-evaluation and adaptation, we test the models trained on dataset And18 on dataset And19. The results are presented in Section VI-D.
6) *Hybrid approach.* We test to what extent we can utilize knowledge about real-time KPI predictions to potentially improve session-level QoE prediction. The used dataset in this case is And19.

TABLE 3. Condensed list of network traffic features used for real-time KPI classification. Each feature is calculated for both downlink and uplink traffic, and on windows of 1, 2, 3, 5, 10, and 20 seconds, constituting the full set of 228 features.

Feature name	Description
pckt_count	Packet count.
pckt_count_gt100	Packet count where all the packets smaller than 100B are ignored (mostly acknowledgements).
throughput	Average throughput.
active_time	Percentage of the window that is used for transmission; interval is considered as used if interarrival time between two consecutive packets in one direction is not longer than 100ms.
mean_pckt_size_gt100	Mean packet size in the window; packets smaller than 100B are ignored.
median_pckt_size_gt100	Median packet size in the window; packets smaller than 100B are ignored.
max_pckt_size_gt100	Maximum packet size in the window; packets smaller than 100B are ignored.
min_pckt_size_gt100	Minimum packet size in the window; packets smaller than 100B are ignored.
stdev_pckt_size_gt100	Standard deviation of the packet size in the window; packets smaller than 100B are ignored.
mean_pckt_size	Mean packet size in the window.
median_pckt_size	Median packet size in the window.
max_pckt_size	Maximum packet size in the window.
min_pckt_size	Minimum packet size in the window.
stdev_pckt_size	Standard deviation of the packet size in the window.
mean_iat	Mean interarrival time in the window.
median_iat	Median interarrival time in the window.
max_iat	Maximum interarrival time in the window.
min_iat	Minimum interarrival time in the window.
stdev_iat	Standard deviation of the interarrival time in the window.

V. REAL-TIME KPI MONITORING

A. METHODOLOGY

Based on ideas presented in related work on buffer estimation and stalling prediction [22], [25], [37], we defined a network traffic feature set containing 228 features. These features include various network-level statistics calculated using only the IP addresses and packet sizes from the traffic trace, making the methodology applicable for various services, platforms, protocols, etc. The core idea is to train models that estimate KPIs for each second of the video streaming session, based on network traffic features calculated on the traffic exchanged during that second, but also wider time-windows, including traffic exchanged in intervals preceding the observed 1 s interval. We cast the problem of KPI estimation as a classification problem, described further on.

The description of used statistics is given in Table 3. Each of the statistics is calculated for both downlink and uplink traffic, and on window sizes of 1, 2, 3, 5, 10, and 20 seconds. To clarify the notion of the window, consider the following example: A 5 s window feature is calculated on a 5 s interval of network traffic, where the most recent of the 5 seconds is the one we are trying to classify into a target KPI class. To indicate the exact statistic, direction, and window size, we use the following convention for naming the features: `<direction>_<statistic>_<window_size>`. For example, `dl_max_iat_w20` refers to a feature calculated as maximum downlink packet interarrival time, calculated based on traffic captured in the last 20 seconds.

The dataset used for KPI estimation model training (And19), includes both features corresponding to each interval, as well as ground-truth labels. As stalling was rarely observed, we focused on resolution and video bitrate classification to demonstrate our approach. Each 1 s interval instance (row in the dataset) consists of 228 feature values and 2 labels: resolution classified into 2 classes (“sd”/“hd”), and bitrate classified into 2 classes (“low”/“high”). Class “hd” corresponds to intervals in which the played video resolution was 720p or higher, while in the case of bitrate, class “high” corresponds to instances with played video bitrate higher or equal to 1000 kbps. The dataset contains 49825 instances, each sample corresponding to 1 s of video playback. Figure 6 shows the distribution of instances across classes.

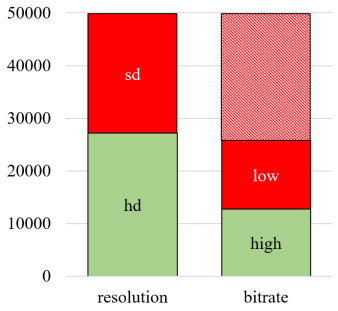


FIGURE 6. Distribution of instances across real-time KPI classes in dataset And19. Instances omitted to balance out the number of samples are shaded.

To train resolution and bitrate classification models, we have taken the following steps: (1) balancing out the number of samples per class by subsampling the dominant class (only in the bitrate classification case), (2) splitting the dataset into training (67%) and validation (33%) set, (3) selecting the 10 most relevant features for each classification by using 2 common approaches: Feature-Importance–based selection (FI)[8] and Sequential Feature Selection (SFS),[9] and (4) training the models by using 2 algorithms: Decision Tree (DT) and Random Forest (RF). We train and test 8 models in total, addressing 2 aforementioned classifications using 2 feature selection methods and 2 ML algorithms.

The results in terms of model performance are often dependent on the complexity of the feature selection process and on the complexity of the algorithm used to train the model. We compare model performances while relying on two commonly used feature selection methods that differ greatly in the amount of resources needed – selection based on feature importance and sequential feature selection. We also compare results achieved with two commonly used algorithms, Decision Tree, and Random Forest consisting of 1000 trees. We note that all median features were omitted, as their calculation is memory-intensive and thus not preferred for potential in-network utilization.

B. RESULTS

Table 4 summarizes the results in terms of resolution classification models' performance on the validation set. The table shows precision, recall, and accuracy values for all four cases depending on whether FI or SFS was used for feature selection, and on whether DT or RF was used for model training. SFS is significantly more computationally-intensive than FI (for comparison purposes only, taking a few hours to complete, as compared to a second for FI run on the same PC). However, as feature selection is done only in the model training phase, and models with SFS–selected features perform significantly better, it makes sense to favor SFS. On the other hand, the performance of RF–trained models is comparable to that of DT–trained models. Given that we are dealing with models that need to be utilized and deployed in the network and run in real-time, simpler models are preferred, thus yielding the conclusion that DT may be a better option.

[8]https://scikit-learn.org/

[9]http://rasbt.github.io/mlxtend/

TABLE 4. Performance of real-time resolution classification models trained and tested using the And19 dataset.

		Algorithm		
		DT	RF	
Feat. selection	FI	hd: 0.80 sd:0.74	hd: 0.82 sd:0.74	Prec.
		hd: 0.78 sd:0.76	hd: 0.77 sd:0.79	Rec.
		0.77	**0.78**	Acc.
	SFS	hd: 0.87 sd:0.83	hd: 0.89 sd:0.81	Prec.
		hd: 0.86 sd:0.84	hd: 0.83 sd:0.87	Rec.
		0.85	**0.85**	Acc.

Figure 7 shows the importance of features in each trained model. In subfigures 7a and 7b, the feature list is the same, given that in both cases the same FI-based selection was used. However, the importance of these features in trained models differs. The same is true in case of SFS-based selection (subfigures 7c and 7d). It can be observed that the most important features in all four cases mostly are the same, but less important features are different depending on whether FI or SFS was used. This indicates that models based on SFS benefited from the exhaustive search of the feature space through employing less important but still very relevant features.

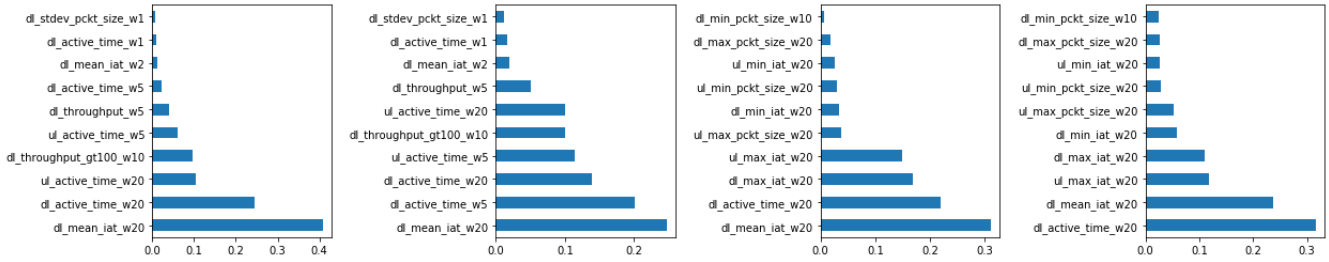
In case of bitrate classification (Table 5), there are no drastic differences in performance of the four models, regardless of the feature selection method and training algorithm. The most important features are mostly the same for all four models, while other features differ (Figure 8). However, it may be concluded that these less important features, in case of bitrate classification, do not contribute much to the model performance anyway, as the relationship between video bitrate and traffic volume is more straightforward than in the case of resolution.

TABLE 5. Performance of real-time video bitrate classification models trained and tested using the And19 dataset.

		Algorithm		
		DT	RF	
Feat. selection	FI	high: 0.84 low:0.89	high: 0.85 low:0.89	Prec.
		high: 0.89 low:0.84	high: 0.89 low:0.85	Rec.
		0.86	**0.87**	Acc.
	SFS	high: 0.87 low:0.89	high: 0.88 low:0.89	Prec.
		high: 0.88 low:0.87	high: 0.89 low:0.88	Rec.
		0.88	**0.89**	Acc.

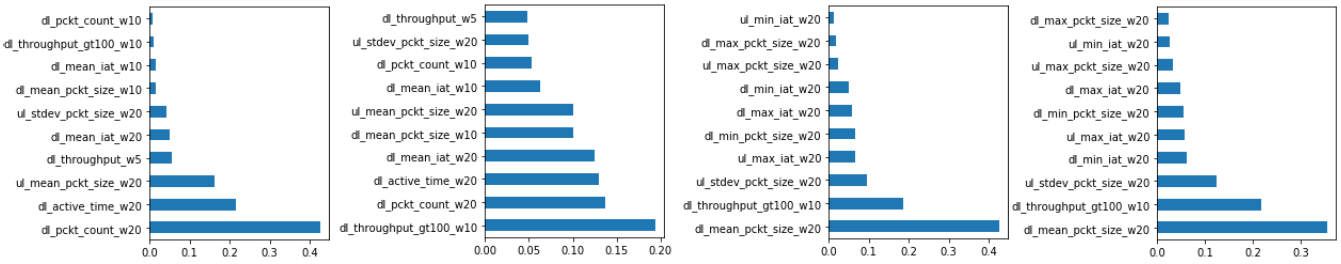
VI. SESSION-LEVEL QoE/KPI MONITORING

To train models that estimate QoE/KPIs on a per-video session-level, it is necessary to extract network traffic features (predictors) from portions of the traffic corresponding to video streaming sessions and label the lists of features with



(a) FI-selected, DT model. (b) FI-selected, RF model. (c) SFS-selected, DT model. (d) SFS-selected, RF model.

FIGURE 7. Feature importances in real-time resolution classification models.



(a) FI-selected, DT model. (b) FI-selected, RF model. (c) SFS-selected, DT model. (d) SFS-selected, RF model.

FIGURE 8. Feature importances in real-time video bitrate classification models.

TABLE 6. Condensed list of network traffic features used for session-level QoE/KPI classification. The full set consists of 62 features.

Feature name	Description
[dl, ul]_ avg_throughput	Average downlink and uplink throughputs in the whole session.
dl_avg_throughput_ [first5s, first10s, first_half, second_half, first20perc, last20perc, mid20perc]	Average downlink throughput in first 5 and 10 seconds of the session, in first and last 50% of the session, first, last and middle 20% of the session.
[dl, ul]_ mean_pckt_size	Average downlink and uplink packet size in the whole session.
dl_active_time	Percentage of the session duration that is used for transmission; interval is considered as used if interarrival time between two consecutive packets in one direction is not longer than 100ms.
ul_pckt_count_gt100	Session packet count, where all the packets smaller than 100B are ignored (mostly acknowledgements).
ul_mean_pckt_size_gt100	Mean packet size in the session; packets smaller than 100B are ignored.
ul_stdev_pckt_size_gt100	Standard deviation of the packet size in the session; packets smaller than 100B are ignored.
[dl, ul]_ [mean, hmean, median, min, max, stdev]_ data_in_ [5s, 3s, 2s, 1s]_intervals	Mean, harmonic mean, median, minimum, maximum and the standard deviation of data amounts bucketed in 5–, 3–, 2–, and 1–s intervals, for downlink and uplink traffic.

ground-truth (targets). In this section, we consider features listed in Table 6 and label the instances with MOS (Mean Opinion Score) classes (“high,” “medium,” “low”), longest played resolution classes (“hd,” “sd”), and average played video bitrate classes (“high,” “low”).

For each video in the dataset, MOS is calculated using the ITU-T Recomm. P.1203 and the implementation published online[10][11] [38], [39], and then classified as follows: into class “high” if MOS if higher or equal to 4.0, class “low” if it is lower than 3.0, and “medium” otherwise. Class “hd” in longest played resolution classification indicates 720p or higher resolutions, and average bitrates higher than 1000 kbps were classified as “high” bitrate.

A. SESSION DELIMITATION

A prerequisite for session-level QoE/KPI estimation is session delimitation, i.e., detection of the *start* and the *end* of the video playback in the traffic. This problem has been investigated in [15], where the authors delimit sessions based on (1) a spike of non-video traffic in downlink at the start of the session, and (2) no video traffic at the end of the session. The authors attribute the aforementioned spike to the download of the player code or to the download of the catalog webpage. However, this solution is only viable in the case when a user is using Web–based streaming applications, and refreshing the Webpage to access the catalog before watching another video. There are also limitations with regards to the session *end* detection, as the amount of content in the buffer will greatly depend on the available bandwidth, and thus the

[10]https://github.com/itu-p1203/itu-p1203

[11]https://github.com/Telecommunication-Telemedia-Assessment/itu-p1203-codecextension

amount of time at the end of the playback in which no content is downloaded can vary.

As an initial step towards finding a more generic and robust solution, we tried applying ML on 1 s intervals for this purpose, and doing so on a dataset collected in a variety of network conditions. We labelled the real-time features (as listed in Table 3) of the And19 dataset with classes "start," "end," and "other." As all data was collected with possible time-shift of up to 1 s (due to the precision of data collection methods), we labelled the first three intervals as "start," the last three as "end," and otherwise as "other." We note that for session-level QoE/KPI estimation, a delimitation error of up to a few seconds is likely not to be considered critical. We split the dataset in half, balanced out the number of instances per class in the first half, selected traffic features using SFS, and trained the models using Random Forest. The models were tested on the whole other half, without balancing.

We first list the selected features as follows:

- ul_min_pckt_size_gt100_w2,
- ul_max_pckt_size_gt100_w3,
- ul_min_pckt_size_gt100_w3,
- dl_mean_pckt_size_w5,
- ul_min_pckt_size_gt100_w5,
- ul_max_pckt_size_w5,
- dl_active_time_w10,
- ul_min_pckt_size_gt100_w10,
- ul_max_pckt_size_w10,
- ul_throughput_w20.

Using these features, we achieved an accuracy of 83 %. Although this would mean that in a 2-minute-long video, over 20 1 s intervals would be misclassified, further inspection of the predictions shows that it is still a viable approach. The results showed some typical classification errors that can be addressed by simple postprocessing of the predictions:

- More than 3 "start" or "end" intervals labelled as "start"/"end" – the algorithm detects the initial burst and the depleting phase – this does not affect the ability of detecting the *start* or *end*.
- Instances between classes "start" and "end" are confused – in our measurements, videos were played one after the other, which may be the reason for this confusion – the predictions still offer enough knowledge to split between the videos.

We also inspected the benefits of adding feature windows that succeed the interval window, as session-level QoE/KPI estimation does not necessarily need to happen right after the session is finished. We added the same features calculated on windows surrounding the instance interval (the instance interval is in the center of the window), and the same features calculated on windows succeeding the instance interval (the instance interval is the first interval in the window). The accuracy increased to up to 92%, with misclassifications following the same patterns as described earlier. However, we note that the performance increase may originate from the model detecting spikes after the end of the video, thus making it only applicable in the case when videos are played one after the other. While these are only initial results, they show the potential of using ML for session delimitation, and present challenges that are yet to be addressed.

B. MODEL TRAINING AND PERFORMANCE

We trained 12 models to validate the framework's session-level QoE/KPI classification: using two datasets (one collected on an Android device, and one on iOS), two ML algorithms for training purposes (Decision Tree and Random Forest), and focusing on MOS, longest played resolution, and video bitrate classification. For each of the models, we first handle imbalances with regards to the number of instances in each class (Figures 9 and 10). In the case of bitrate, we randomly subsample the dominant class, while in the case of MOS, we subsample the majority class and upsample the minority class using SMOTE (Synthetic Minority Over-sampling Technique).[12] Following the class balancing, the dataset was split into training and validation sets (67% : 33%), features were subset using SFS, and top 10 features were used to train DT and RF models. The results in terms of model performance are presented in Table 7 for Android, and in Table 8 for iOS.

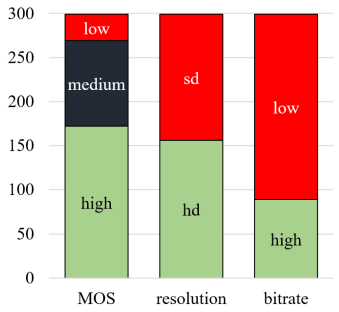


FIGURE 9. Distribution of instances across session-level QoE/KPI estimation classes in dataset And19.

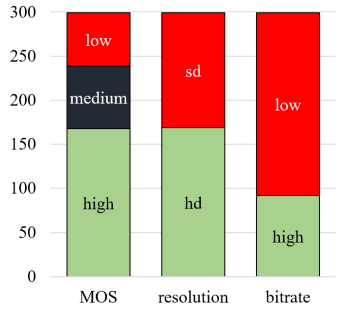
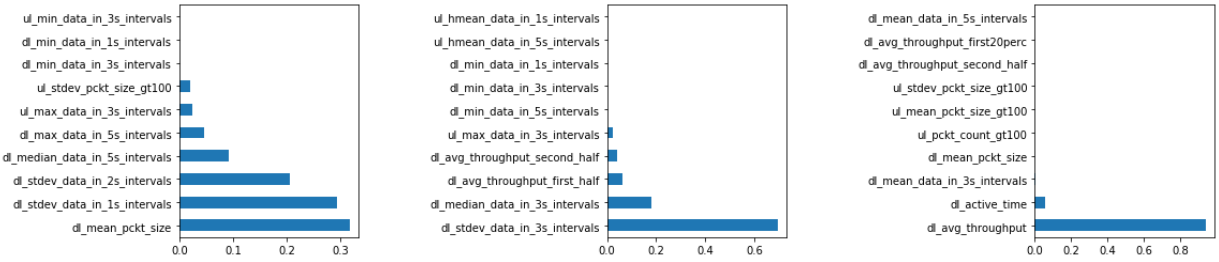


FIGURE 10. Distribution of instances across session-level QoE/KPI estimation classes in dataset iOS19.

The results show that, both for Android and iOS, average video bitrate is predicted with highest performance. Looking into features that were used in bitrate classification

[12]https://github.com/scikit-learn-contrib/imbalanced-learn



(a) MOS, 3 classes, RF model. (b) Resolution, 2 classes, DT model. (c) Bitrate, 2 classes, DT model.

FIGURE 11. Feature importances in session-level QoE/KPI classification models trained for Android platform.

TABLE 7. Performance of session-level YouTube QoE/KPI classification models trained and tested using the And19 dataset.

	Algorithm		
	DT	RF	
MOS: high/medium/low	h:0.76 m:0.66 l:0.71	h:0.89 m:0.69 l:0.69	Prec.
	h:0.57 m:0.66 l:0.87	h:0.61 m:0.66 l:0.94	Rec.
	0.70	**0.73**	Acc.
Longest played resolution: hd/sd	hd:0.84 sd:0.80	hd:0.92 sd:0.72	Prec.
	hd:0.81 sd:0.83	hd:0.67 sd:0.94	Rec.
	0.82	**0.80**	Acc.
Average video bitrate: high/low	h:0.92 l:0.97	h:1.00 l:0.95	Prec.
	h:0.96 l:0.94	h:0.92 l:1.00	Rec.
	0.95	**0.97**	Acc.

TABLE 8. Performance of session-level YouTube QoE/KPI classification models trained and tested using the iOS19 dataset.

	Algorithm		
	DT	RF	
MOS: high/medium/low	h:0.67 m:0.70 l:0.56	h:0.62 m:0.64 l:0.47	Prec.
	h:0.83 m:0.55 l:0.56	h:0.75 m:0.55 l:0.44	Rec.
	0.65	**0.59**	Acc.
Longest played resolution: hd/sd	hd:0.77 sd:0.72	hd:0.85 sd:0.77	Prec.
	hd:0.81 sd:0.67	hd:0.82 sd:0.81	Rec.
	0.75	**0.82**	Acc.
Average video bitrate: high/low	h:0.81 l:0.92	h:0.85 l:0.94	Prec.
	h:0.88 l:0.86	h:0.92 l:0.89	Rec.
	0.87	**0.90**	Acc.

models, it is clear that these models are also very simple, as video bitrate highly correlates with downlink throughput. We demonstrate this with Figure 11c, depicting the feature importance in DT–based bitrate classification for Android. Resolution classification models also heavily rely on downlink throughput features, but result in lower performance, when compared to bitrate classification models. This may be emphasized due to the fact that used datasets include around 20% of static content (such as music with album covers). In these cases, high resolutions do not necessarily result in high bitrates and consequently downlink throughput. Features used in DT–based resolution classification for Android are shown in Figure 11b.

Finally, MOS classification proves to perform the worst, ranging from 59 to 65%. This is expected, as MOS is already a complex construct influenced by a variety of KPIs that reflect in different ways on the network-level. Features used in RF–based MOS classification for Android are shown in Figure 11a. Aiming to improve the performance of MOS classification models, we propose the hybrid approach in Section VI-E, which uses real-time KPI predictions as additional features. However, we stress that the MOS classification models presented in this section may be sufficient in certain applications, given that most misclassifications occur between classes ''low'' and ''medium'' or ''high'' and ''medium.''

Due to a lack of guidelines in terms of network conditions in which the data should be collected, what kind of content the dataset should include, etc., data collection methods differ across related work. Moreover, related work mostly focuses on YouTube viewed in the browser, rather the native YouTube app. We thus avoid direct comparisons with related work in terms model performance. We are aware that introducing estimated chunk–based features, such as in [15], [19] would potentially improve the performance of the models presented in this section. However, chunk detection requires significantly more processing, and may be impossible in cases when multiple chunks are downloaded in parallel. This is why, to keep our models simple and robust, we only focus on network-layer features.

C. POTENTIAL FOR MODEL GENERALIZATION

In light of the variety of use-cases, even if only focusing on YouTube, it is beneficial to explore the possibility of training models able to address multiple cases. We train 6 models – for classifying MOS, resolution and bitrate using DT and RF – on the merged dataset that includes both Android and iOS data (And19 and iOS19). To the best of our knowledge, QoE estimation for video streaming on iOS has only to a certain extent been addressed in previous work [23], but not in the context of generic models that address multiple platforms. We follow the same procedure as in Section VI-B, including splitting, subsetting the merged dataset to balance out the

number of samples in line with the least populated class and SFS–based feature selection. Trained models' performance is summarized in Table 9.

TABLE 9. Performance of session-level YouTube QoE/KPI classification models for Android and iOS (trained and tested using the merged And19 and iOS19 dataset).

	Algorithm		
	DT	RF	
MOS: high/medium/low	h:0.57 m:0.55 l:0.77	h:0.81 m:0.60 l:0.68	Prec.
	h:0.61 m:0.68 l:0.55	h:0.61 m:0.68 l:0.74	Rec.
	0.61	**0.68**	Acc.
Longest played resolution: hd/sd	hd:0.91 sd:0.69	hd:0.85 sd:0.74	Prec.
	hd:0.64 sd:0.92	hd:0.74 sd:0.85	Rec.
	0.77	**0.79**	Acc.
Average video bitrate: high/low	h:0.88 l:0.89	h:0.95 l:0.76	Prec.
	h:0.88 l:0.89	h:0.67 l:0.97	Rec.
	0.89	**0.83**	Acc.

The results are evidently comparable with the results achieved with separate models. This means that a single model can potentially address multiple use-cases and there is no need to first recognize the platform once the model is deployed. This does not necessarily mean that the model itself can be trained on a single-platform dataset and applied to another platform, as has been investigated in [23], due to service implementation differences on the two platforms [40]. Similar conclusions, but focused on different services and not platforms, have been found in [15]. The authors show that developing well–performing generic models is feasible if the training set included data from all services. Applying models trained on one service to another brought a significant drop in model performance. In [23] we found that the drop for different platforms is not that significant, but still existent.

D. MODEL RE-EVALUATION

Streaming services can occasionally apply updates of the adaptation logic and thus change the typical network traffic patterns that QoE/KPI estimation models are dependent on. While the changes can be observed on an application-level as changes in the amount of data that is preloaded (buffered), changed segment size, etc., in this section we test to what extent these changes reflect on the performance of QoE/KPI classification models. We train the models on the dataset And18, collected on an Android device in early 2018 and test it on the dataset collected on Android in early 2019 (And19). Figure 12 depicts the number of instances in each of the classes in dataset And18. The same is displayed for dataset And19 in Figure 9. Prior to model training, we subsample the And18 dataset to balance out the number of instances in accordance with the least populated class. We note that the dataset And19 was not subsampled prior to model testing. Models' performance is reported in Table 10.

The results show a slight decrease in performance of resolution– and bitrate–classification models, but significant

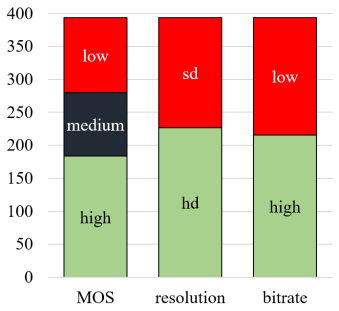


FIGURE 12. Distribution of instances across session-level QoE/KPI estimation classes in dataset And18.

TABLE 10. Performance of session-level YouTube QoE/KPI classification models trained on Android dataset from 2018 (And18) on a newer Android dataset from 2019 (And19).

	Algorithm		
	DT	RF	
MOS: high/medium/low	h:0.99 m:0.57 l:0.27	h:0.95 m:0.53 l:0.20	Prec.
	h:0.48 m:0.65 l:0.93	h:0.48 m:0.35 l:1.00	Rec.
	0.58	**0.49**	Acc.
Longest played resolution: hd/sd	hd:0.85 sd:0.68	hd:0.60 sd:0.95	Prec.
	hd:0.62 sd:0.88	hd:0.60 sd:0.95	Rec.
	0.74	**0.77**	Acc.
Average video bitrate: high/low	h:0.64 l:0.99	h:0.70 l:0.96	Prec.
	h:0.98 l:0.77	h:0.92 l:0.83	Rec.
	0.83	**0.86**	Acc.

decrease in MOS–classification performance. We explain this as follows. The KPIs we aim to predict, *longest played resolution* and *average video bitrate*, are averaged across the whole session and do not carry temporal information. Thus, a lot of information regarding the adaptation strategy is lost. On the other hand, the ITU-T Recomm. P.1203 model, based upon which MOS is calculated, takes into account all the QoE-influencing data on a per-second level, which is where the adaptation strategy changes come into play.

E. CONSIDERATIONS FOR IMPROVING QoE/KPI MONITORING PERFORMANCE- HYBRID APPROACH

With the aim of improving the performance of MOS classification models, we enrich them with predictions made by real-time resolution and bitrate classification models. Once the per-second predictions for the session are available, they are aggregated into two additional features (on top of the ones defined in Table 6) – percHD and percHBr – percentage of intervals with high definition and percentage of intervals with high bitrate. As in previous sections, we balance out the dataset, split it into train and validation set (67% : 33%), select relevant features, and train DT and RF models. The results are given in Table 11.

The performance of MOS classification models was significantly improved, while the complexity of the model is reduced. We illustrate this statement with Figure 13 that

TABLE 11. Performance of session-level YouTube QoE classification models enriched with real-time predictions as additional features. Models were trained and tested using the And19 dataset.

	Algorithm		
	DT	RF	
MOS: high/medium/low	h:0.93 m:0.89 l:0.73	h:1.00 m:0.85 l:0.72	Prec.
	h:0.93 m:0.66 l:0.97	h:0.89 m:0.74 l:0.90	Rec.
	0.84	**0.84**	Acc.

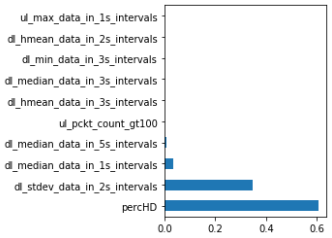


FIGURE 13. Feature importances in the hybrid MOS classification model.

shows feature importances in the DT–based MOS classification model. The percentage of intervals with HD resolution was confirmed as the most important feature, followed by one of the commonly used throughput–based features. PercHBr was not considered relevant by the SFS algorithm. The reason for that may be that bitrate highly correlates with certain throughput–based features, and introducing percHBr carries no additional value to the model. On the other hand, as throughput–based features can be misleading for videos with static content, percHD carries additional value in identifying such cases.

VII. CONCLUSIONS AND FUTURE RESEARCH CHALLENGES

The applicability of machine learning techniques has been widely studied in the domain of in-network QoE/KPI estimation for HAS. Studies have, however, for the most part been focused on isolated problems, mostly trying to solve methodological issues, such as data collection or feature extraction. In this paper we provide a broader view of the challenges in this domain by presenting a conceptual framework for QoE/KPI estimation that enables automatic model training, deployment and execution, and re-evaluation. Individual components of the framework have been demonstrated on a use-case concerning YouTube. The framework is generic and can be translated to other services as well, by following the guidelines given through the YouTube example. Following this example, we present our methodology for real-time KPI estimation and session-level QoE/KPI estimation, focusing on YouTube video on demand. We emphasize the fact that we collected data through the official YouTube apps, thus observing realistic adaptation behaviors. Related work very scarcely addresses this scenario (especially for iOS), due to data collection being more complex.

As efficient data collection is crucial, we are currently putting our efforts into automating the data collection process, which will enable running larger measurement campaigns, and thus potentially will increase the robustness of the models. This would also enable including playback–related user interactions, making our datasets more realistic. What is currently missing are guidelines on what data distributions are needed in collected datasets for models to be robust, and furthermore how much data is needed. Specifically, what are the different types of content that should be included, what kind of network conditions should be included, etc.

There are also numerous challenges related to the deployment of devised models, especially in the context of 5G networks and 5G-specific network functions that support network data analytics [16]. One of the problems related to the actual deployment is also the computational complexity. While we scratch the surface of this question by addressing the trade-off between model complexity and accuracy, it remains unclear how much resources these models would require if deployed in the network, and what portion of the traffic in the network should and could be analysed. An interesting avenue for future research is also exploration of the potential for utilizing P4 and deploying QoE estimations directly in the data plane [41]–[43].

Undoubtedly, a big challenge for in-network QoE/KPI estimation lies in the dimensionality of the problem and in the changes that can be introduced within a single service over time. From the data analytics perspective, this may be observed as the "concept drift" phenomenon, depending on the domain also known as "covariate shift" or "dataset shift." An interesting research question is whether these shifts could be addressed in deployed models without collecting new ground-truth data, or at least collecting significantly less ground-truth data, using methods from the domain adaptation field. With regards to the services with different streaming adaptation logic, related work has shown that it is possible to train well-performing generic models that infer QoE/KPIs for multiple services. However, this has proven to be the case only if the model was trained on data from all services. The question remains whether different techniques, for example those from the domain of transfer learning and semi-supervised learning, could make training data collection less exhaustive.

In future work, we plan to address additional use-cases, but are especially interested in live streaming services, such as YouTube Live and Twitch, as these types of services are gaining popularity and are more delay–sensitive and thus susceptible to QoE degradations.

ACKNOWLEDGMENT

The authors would like to thank Ivan Bartolec for his help on developing the used tools.

REFERENCES

[1] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen, "A survey of emerging concepts and challenges for QoE management of multimedia services," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 14, no. 2, pp. 1–29, May 2018.

[2] Sandvine Corp., "The mobile Internet phenomena report," Sandvine, San Jose, CA, USA, Tech. Rep., 2020. [Online]. Available: https://www.sandvine.com/download-report-mobile-internet-phenomena-report-2020-sandvine

[3] Cisco Systems, Inc., "Cisco visual networking index: Global mobile data traffic forecast update 2017-2022," Cisco, San Jose, CA, USA, White Paper C11-741490-01, 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.pdf

[4] Ericsson Corp., "Ericsson mobility report," Ericsson, Stockholm, Sweden, Tech. Rep. EAB-19:007381, 2019. [Online]. Available: https://www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf

[5] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, 2011.

[6] R. Pantos and W. May, "HTTP live streaming," IETF, Fremont, CA, USA, Tech. Rep. RFC 8216, 2017. [Online]. Available: https://datatracker.ietf.org/doc/rfc8216/

[7] J. Añorga, S. Arrizabalaga, B. Sedano, J. Goya, M. Alonso-Arce, and J. Mendizabal, "Analysis of YouTube's traffic adaptation to dynamic environments," *Multimedia Tools Appl.*, vol. 77, no. 7, pp. 7977–8000, Apr. 2018.

[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 Appl.*, vol. 76, no. 21, pp. 22267–22301, Nov. 2017.

[9] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hossfeld, "Modeling the YouTube stack: From packets to quality of experience," *Comput. Netw.*, vol. 109, pp. 211–224, Nov. 2016.

[10] A. Mondal, S. Sengupta, B. R. Reddy, M. J. V. Koundinya, C. Govindarajan, P. De, N. Ganguly, and S. Chakraborty, "Candid with YouTube: Adaptive streaming behavior and implications on data consumption," in *Proc. 27th Workshop Netw. Operating Syst. Support Digit. Audio Video (NOSSDAV)*, 2017, pp. 19–24.

[11] Google LLC, "HTTPS encryption on the Web," Google, Mountain View, CA, USA, Tech. Rep., 2020. [Online]. Available: https://transparencyreport.google.com/https/overview

[12] J. Iyengar and M. Thomson, "QUIC: A UDP-based multiplexed and secure transport," IETF, Fremont, CA, USA, Tech. Rep. draft-ietf-quic-transport-27, 2020. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-transport/

[13] M. Bishop, "Hypertext transfer protocol version 3 (HTTP/3)," IETF, Fremont, CA, USA, Tech. Rep. draft-ietf-quic-http-27, 2020. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-http/

[14] 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 *Proc. 15th Workshop Mobile Comput. Syst. Appl. (HotMobile)*, 2014, p. 18.

[15] F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, pp. 1–25, Dec. 2019.

[16] S. Schwarzmann, C. Cassales Marquezan, M. Bosk, H. Liu, R. Trivisonno, and T. Zinner, "Estimating video streaming QoE in the 5G architecture using machine learning," in *Proc. 4th Internet-QoE Workshop QoE-Based Anal. Manage. Data Commun. Netw. (Internet-QoE)*, 2019, pp. 7–12.

[17] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, "I see what you see: Real time prediction of video quality from encrypted streaming traffic," in *Proc. 4th Internet-QoE Workshop QoE-Based Anal. Manage. Data Commun. Netw. (Internet-QoE)*, 2019, pp. 1–6.

[18] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, "Let me decrypt your beauty: Real-time prediction of video resolution and bitrate for encrypted video streaming," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2019, pp. 199–200.

[19] C. Gutterman, K. Guo, S. Arora, X. Wang, L. Wu, E. Katz-Bassett, and G. Zussman, "Requet: Real-time QoE detection for encrypted YouTube traffic," in *Proc. 10th ACM Multimedia Syst. Conf.*, Jun. 2019, pp. 48–59.

[20] I. Bartolec, I. Orsolic, and L. Skorin-Kapov, "In-network YouTube performance estimation in light of end user playback-related interactions," in *Proc. 11th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, Jun. 2019, pp. 1–3.

[21] M. Seufert, P. Casas, N. Wehner, L. Gang, and K. Li, "Stream-based machine learning for real-time QoE analysis of encrypted video streaming traffic," in *Proc. 22nd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Feb. 2019, pp. 76–81.

[22] M. Seufert, P. Casas, N. Wehner, L. Gang, and K. Li, "Features that matter: Feature selection for on-line stalling prediction in encrypted video streaming," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 688–695.

[23] I. Orsolic, P. Rebernjak, M. Suznjevic, and L. Skorin-Kapov, "In-network QoE and KPI monitoring of mobile YouTube traffic: Insights for encrypted iOS flows," in *Proc. 14th Int. Conf. Netw. Service Manage. (CNSM)*, Nov. 2018, pp. 233–239.

[24] 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. 10th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, May 2018, pp. 1–6.

[25] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for HTTPS and QUIC," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 1331–1339.

[26] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "YouTube QoE estimation based on the analysis of encrypted network traffic using machine learning," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.

[27] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video QoE from encrypted traffic," in *Proc. ACM Internet Meas. Conf. (IMC)*, 2016, pp. 513–526.

[28] P. Casas, R. Schatz, and T. Hossfeld, "Monitoring YouTube QoE: Is your mobile network delivering the right experience to your customers?" in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 1609–1614.

[29] R. Schatz, T. Hossfeld, and P. Casas, "Passive YouTube QoE monitoring for ISPs," in *Proc. 6th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.*, Jul. 2012, pp. 358–364.

[30] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Using session modeling to estimate HTTP-based video QoE metrics from encrypted network traffic," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 1086–1099, Sep. 2019.

[31] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "BUFFEST: Predicting buffer conditions and real-time requirements of HTTP(S) adaptive streaming clients," in *Proc. 8th ACM Multimedia Syst. Conf.*, Jun. 2017, pp. 76–87.

[32] 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. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2015, pp. 239–243.

[33] M. Seufert, B. Zeidler, F. Wamser, T. Karagkioules, D. Tsilimantos, F. Loh, P. Tran-Gia, and S. Valentin, "A wrapper for automatic measurements with YouTube's native Android app," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2018, pp. 1–8.

[34] P. Casas, M. Seufert, N. Wehner, A. Schwind, and F. Wamser, "Enhancing machine learning based QoE prediction by ensemble models," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1642–1647.

[35] 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 Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.

[36] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 8, no. 3, pp. 1–19, Jul. 2012.

[37] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying flows and buffer state for Youtube's HTTP adaptive streaming service in mobile networks," in *Proc. 9th ACM Multimedia Syst. Conf. (MMSys)*, 2018, pp. 138–149.

[38] A. Raake, M.-N. Garcia, W. Robitza, P. List, S. Goring, and B. Feiten, "A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P.1203.1," in *Proc. 9th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, May 2017, pp. 1–6.

[39] W. Robitza, M.-N. Garcia, K. Yamagishi, S. Broom, S. Göring, A. Raake, D. Lindegren, G. Heikkilä, J. Gustafsson, P. List, B. Feiten, and U. Wüstenhagen, "HTTP adaptive streaming QoE estimation with ITU-T rec. P. 1203: Open databases and software," in *Proc. 9th ACM Multimedia Syst. Conf. (MMSys)*, Amsterdam, The Netherlands, 2018, pp. 466–471.

[40] I. Orsolic, L. Skorin-Kapov, and T. Hosfeld, "To share or not to share? How exploitation of context data can improve in-network QoE monitoring of encrypted YouTube streams," in *Proc. 11th Int. Conf. Qual. Multimedia Exper. (QoMEX)*, Jun. 2019, pp. 1–3.

[41] D. Bhamare, A. Kassler, J. Vestin, M. A. Khoshkholghi, and J. Taheri, "IntOpt: In-band network telemetry optimization for NFV service chain monitoring," in *Proc. ICC-IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.
[42] J. Vestin, A. Kassler, D. Bhamare, K.-J. Grinnemo, J.-O. Andersson, and G. Pongracz, "Programmable event detection for in-band network telemetry," 2019, *arXiv:1909.12101*. [Online]. Available: http://arxiv.org/abs/1909.12101
[43] J. Langlet, A. Kassler, and D. Bhamare, "Towards neural network inference on programmable switches," Karlstad Univ., Karlstad, Sweden, Tech. Rep., 2019. [Online]. Available: https://kau.app.box.com/s/4k41n6s30d0fh1rawnhmedt0akkdgtnn

IRENA ORSOLIC (Graduate Student Member, IEEE) received the M.Sc. degree in information and communication technologies, in 2016. After receiving the M.Sc. degree, she enrolled in the Ph.D. Program in computer science with the Faculty of Electrical Engineering and Computing. She is currently a Research Assistant with the Faculty of Electrical Engineering and Computing, University of Zagreb, and a member of the Multimedia Quality of Experience Research Lab (MUEXlab). The focus of her research is on Quality of Experience (QoE) estimation of encrypted video streaming by using machine learning methods. In particular, she is focusing on researching potential solutions for QoE monitoring based on the analysis of encrypted network traffic using machine learning methods, which could possibly enable the development of mechanisms for improving QoE and efficient use of network resources.

LEA SKORIN-KAPOV (Senior Member, IEEE) was previously employed as a Senior Research Engineer and the Project Manager with the Research and Development Center, Ericsson Nikola Tesla, Croatia. She is currently an Associate Professor with the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, and the Head of the Multimedia Quality of Experience Research Lab (MUEXlab). Her research interests include Quality of Experience (QoE) modeling of multimedia applications, QoE monitoring of encrypted video traffic, cross-layer negotiation and management of QoS/QoE, and resource allocation and optimization mechanisms. She has published over 100 scientific articles and serves on the editorial boards of the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and *Multimedia Systems* Journal (Springer). She has served as a Guest Editor for the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING and *ACM Transactions on Multimedia Computing, Communications, and Applications*.