On machine learning based video QoE estimation across different networks

Irena Orsolic, Michael Seufert

Angaben zur Veröffentlichung / Publication details:

Orsolic, Irena, and Michael Seufert. 2021. "On machine learning based video QoE estimation across different networks." In *16th International Conference on Telecommunications (ConTEL), 30 June - 2 July 2021, Zagreb, Croatia*, edited by Martina Antonić and Jurica Babić, 62–69. Piscataway, NJ: IEEE. https://doi.org/10.23919/contel52528.2021.9495971.

On Machine Learning Based Video QoE Estimation Across Different Networks

Irena Orsolic
University of Zagreb
Faculty of Electrical Engineering and Computing
Zagreb, Croatia
irena.orsolic@fer.hr

Michael Seufert
University of Würzburg
Chair of Communication Networks
Würzburg, Germany
michael.seufert@uni-wuerzburg.de

***Abstract*—With Over-The-Top traffic being extensively encrypted end-to-end, network operators typically lack insight into the performance of these services, as perceived by the end users. Yet, such an insight is essential for employing QoE-aware network management and potential alleviation of problems that may originate in the network. There is a clear interest from network operators to find ways to estimate service performance in terms of Key Performance Indicators (KPIs) and Quality of Experience (QoE). Over the last years, machine-learning–based (ML) models have proven to be capable of inferring QoE/KPIs from patterns in encrypted network traffic. The particular focus has mostly been on adaptive video streaming services, considering their share of the global network traffic. Those ML–based models have typically been trained and tested on a single dataset collected under specific conditions only. Going beyond related work on the topic of QoE/KPI estimation, we collected two large datasets related to YouTube streaming using the same setup at two different locations in Europe and analyzed the extent to which the differences in network characteristics and location specifics influence the performance of such models. This is of interest, as applicability of the models across diverse networks would significantly reduce the needed extensiveness of data collection typically required for ML–based approaches. In this paper, we compare models trained and tested on a single dataset/location (network-specific), models trained on the merged dataset (general), and models trained on one dataset and tested on the other dataset (cross-tested). The results show that the performance of general models is comparable to that of network-specific models, but cross-tests exhibit a considerable reduction in performance. With the aim to understand and improve cross-network applicability of the models in the future, the paper also provides an investigation of underlying reasons for the performance degradation.**

I. Introduction

As the global number of mobile subscribers is expected to reach 5.7 billion in 2023, which is 71% of global population [1], Quality of Experience (QoE) monitoring is an important ingredient for the successful management of mobile networks. It empowers network operators to track the delivery of networked services and to rapidly detect issues which can negatively affect the users' experience. Thus, it allows to ensure a high service quality and a high user satisfaction, which avoids user churn in the competitive market. This has resulted in extensive QoE–related research conducted over the past decade, with the networking community increasingly aiming to introduce QoE-awareness into network management cycles [2]. In the last decade, video streaming has been one of the users' most popular Internet services in mobile networks. According to [3], video accounted for 63% of mobile traffic in 2019, and its share is expected to grow to 76% by 2025.

As end-to-end encrypted traffic over HTTPS becomes the predominant type of video streaming traffic, 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. A lot of effort has been put into tackling the problem of inferring application-level Key Performance Indicators (KPIs) and QoE from the statistical features extracted from encrypted video streaming traffic using machine learning (ML) approaches, e.g., [4], [5], [6], [7]. The presented data-driven models already managed to learn application behavior and the resulting QoE with a high accuracy in tested lab scenarios. Nonetheless, such solutions are currently far from deployment in operational networks.

The biggest challenge is that ML approaches require a huge amount of training data, which necessitates extensive network measurements. However, ground-truth information, i.e., application-level KPIs available at the client that can be used as labels, are not (easily) available for all video streaming services. Moreover, as such measurements are time-consuming and costly, they can only be conducted for specific scenarios, i.e., specific video streaming services, devices, underlying transport protocols, access technologies, network types, and network conditions. Here, training too specific models might limit the generality, as ML models assume the same distribution of training and test data. Although the performance on the training scenario could be better than for a general model, applying the specific model to scenarios for which it was not trained, i.e., cross-testing, could result in performance degradation. To fight cross-testing performance degradation, additional measurements need to be conducted in the new scenario and ML models need to be retrained. This might improve performance on the currently considered scenario, but eventually could only reset the problem when the model is to be applied to more scenarios. Therefore, a desirable model would be general, such that it is cross-applicable to many scenarios, although there might be a trade-off between generality and performance.

In this work, we investigate the trade-off between cross-applicability of ML–based models and the video QoE estimation performance for different networks. We conducted a set of 24000 video streaming measurements in a high speed optical fiber campus network at the University of Würzburg, Germany and in a cable broadband network offered by an ISP in Zagreb, Croatia. In both locations, the same videos were streamed under the same controlled network conditions, recording application-level KPIs of the streamed videos, namely, initial delay, stalling, average resolution, average bitrate, and mean opinion score (MOS). We train general and network-specific ML–based models, present baseline results of the models for six KPI classification tasks, and provide an outlook to approaches for future performance improvements. Thereby, our work is the first to analyze the performance and discuss the cross-applicability of the models for QoE estimation across different networks.

This paper is structured as follows. Section II outlines background and related work on ML–based QoE measurements. Section III describes the measurement setup, the dataset preparation, as well as the analysis procedure. We present the baseline results for network-specific models and general models, and investigate their cross-network applicability in Section IV. Finally, Section V concludes this work and provides an outlook to future work.

II. Background and Related Work

Network operators and communications researchers found that the subjectively perceived experience with networked applications is a major business factor, and thus, introduced the concept of Quality of Experience (QoE) [8], [9]. In contrast to the previously prevailing Quality of Service (QoS), it puts the users and their perceived experience to the center of the evaluation process [10]. Numerous QoE studies have been conducted to investigate the impact of technical parameters of systems and networks on the experience of end users. The results of these studies can be considered by network operators for network management to avoid QoE degradation and improve the experience of end users with networked applications [11], [12].

When it comes to QoE for HTTP adaptive video streaming (HAS), which is one of the most prominent Internet services today, the summary of key results is given in [13]. It was found that initial delay, stalling, and quality adaptation are the most dominant QoE factors. Stalling (or re-buffering), i.e., playback interruption due to buffer depletion, is considered the worst QoE degradation. Moreover, the played-out video quality and the time spent on each quality layer also impact the QoE, whereas the impact of initial delay is rather small. Recently, QoE models for HAS were standardized in ITU-T Recomm. P.1203 and P.1204 [14], [15], and are able to estimate the MOS from stream inspection considering four different modes of input information.

With HAS services predominantly using HTTPS in recent years and traffic being encrypted end-to-end, approaches for in-network QoE measurement relying on Deep Packet Inspection (DPI) can no longer be used. Thus, ML methods are employed to model the video traffic based on statistical features with the goal to derive QoE information. Such an approach was initially described in [4], where the authors collected passive in-network measurements and leveraged ML to obtain mappings between QoS and QoE for mobile video applications. Going beyond just classifying QoE, [5] classified stalling, average video quality, and quality variations from TCP-level traffic features using random forests. These approaches, however, still required access to packet payloads, at least in the model training phase. Later, approaches fully applicable in the context of traffic encryption were developed. In [16], [17], ML was applied to predict QoE of several types of mobile apps (including video streaming) from network parameters. For QoE/KPI estimation problem, tree-based algorithms were found to perform well, such as in [18] for video bitrate estimation and in [6], [19] for the estimation of QoE and various KPIs.

While the listed papers addressed the QoE/KPI estimation problem in a per-video manner, in the context of QoE-aware network management, monitoring these metrics in shorter intervals may be more valuable. In [20], YouTube QoE/KPIs were classified in time slots of 10 s from features derived from packet sizes, interarrival times, and throughput measurements. Going for an even more fine-grained estimation, [7], [21] estimated video resolution, average bitrate, and stalling from encrypted video traffic in real-time within slots of 1 s by using a stream-like analysis approach and IP-level packet features.

Besides relying on features that can easily be calculated from the captured traffic traces, there are also approaches that rely on the detection of typical HAS behavior. The approach presented in [22] detects video chunks (segments) to additionally include chunk-based features, when the chunks are identifiable. Features beyond traffic statistics have also been investigated in [23], showing that minor context information provided by streaming service providers willing to share it could significantly improve the performance of in-network QoE estimation models.

Apart from the sole definition of network traffic features, QoE/KPI prediction targets, used measurement setups, and ML procedures, there is a series of challenges yet to be investigated. The performance of QoE/KPI estimation models is greatly affected by playback–related user interactions, which has been demonstrated in [24], stressing the need to include such interactions in the model training phase. The actual utilization of QoE/KPI estimation models in the network has been briefly addressed in [25], [26], [27], but the exact mapping of these models to network architectures and the amount of resources required for their operation is still unclear.

Related work has barely scratched the surface of the problem of inherent dimensionality originating from the variety of possible video streaming usage scenarios. The cross-testing efforts described in [28], [19] were focused on the applicability of YouTube QoE/KPI classification models trained in a lab setting on data collected in an operational mobile network. Similarly, models trained on data collected on Android platform were cross-tested with data collected on iOS [28]. The paper reports limited cross-applicability capabilities, with models demonstrating a decrease in performance. On the other hand, the performance of general models, trained on the dataset containing samples from both platforms is comparable to that of models trained for a specific platform [29]. Similar conclusions, but focused on different services and not platforms, have been found in [30]. The authors show that developing well–performing general models is feasible if the training set included data from all services. Applying the model trained on Amazon, YouTube, and Twitch data to Netflix data resulted in a significant drop in model performance. Regarding the generalization efforts, interesting approaches can be found in [31], [32], where the authors investigate challenges related to model sharing and a transfer-learning approach which allows local models to learn a generic base model for MOS, and then consider additional features for location-specific QoE models. However, both approaches rely on application-level KPIs and do not consider estimating QoE from encrypted network traffic. Our work further extends the investigation of cross-testing, focusing on QoE/KPI estimation models applicable in the context of encrypted traffic, and by considering model performance across two different operational networks.

III. Methodology

A. Measurement setup

The measurements were conducted using a Java-based framework similar to [33], [7]. The measurement framework is able to automatically start a Chrome browser using the Selenium browser automation tool[1]. The browser was configured to log all HTTP requests to a file (`-log-net-log`) and QUIC traffic was enabled (`--enable-quic`). Optionally, the browser could also load and install a Chrome extension during startup. For a single measurement run, the browser creates a new and isolated browsing session independent of browsing history or previously stored session or user data (e.g., cookies), and accesses the YouTube main page. After the page has fully loaded and occasional pop-ups have been handled, the framework spawns a separate thread, which captures the network traffic using tshark[2]. Next, the browser accesses a single YouTube video page and injects a JavaScript-based monitoring script [34], [35] into the webpage, which periodically polls the current timestamp, the current video playtime, buffered playtime, video resolution, and player state every 250 ms. The video is then streamed for 180 s or until the video end, and the application-layer information about the streaming session is logged to a file. Afterwards, the framework closes the browser and terminates the network traffic capture, before a new measurement run can be started.

A list of 2000 YouTube IDs was selected according to the popularity of the video content, such that the full range of video popularity, ranging from below 100 views to over billions of views, was represented in the list. The measurements were conducted in a high speed optical fiber campus network at the University of Würzburg, Germany and in a cable broadband network of an ISP in Zagreb, Croatia. In both locations, the framework was installed on a laptop and a Raspberry Pi 4 was used as a bridge to connect the laptop to the network. The Raspberry Pi acted as a network emulator and was able to limit the bandwidth using Linux traffic control (`tc`). Three different network conditions were emulated in both locations, namely, no limitation, a fixed limitation of 1 Mbps, as well as a stochastic limitation following an exponential distribution with a mean of 1 Mbps. The whole list of 2000 videos was measured both without and with an ad blocking Chrome extension, in both locations, and in all three network conditions, which results in a dataset of 24000 YouTube video sessions. The measurement runs were conducted over five months from July to November 2020.

B. Dataset preparation

The recorded pcap network traces were parsed with a Java parser based on Kaitai Struct[3] and transformed into text-based log files. These log files include all basic information for each packet in a simple comma-separated values (CSV) format, namely, timestamp, source IP, source port, destination IP, destination port, and size. Moreover, DNS lookup responses were extracted to obtain a mapping between IP addresses and domain names, which could later be used to identify YouTube video flows based on their characteristic *googlevideo.com* domain name.

The feature extraction considered all traffic of YouTube video flows between the start of the browsing to the video web page until the closing of the browser. It collected basic statistics from the traffic, such as packet count (both for all packets and only for packets greater than 100 bytes, thereby eliminating acknowledgements), byte count, and average throughput. Moreover, summary statistics (mean, variance, standard deviation, coefficient of variation, skewness, kurtosis, minimum, maximum) were computed for the distribution of packet sizes (again, both for all packets and only for packets greater than 100 bytes) and interarrival times, and for the distribution of traffic volume in time slots of length 100 ms, 1 s, as well as 10 s. All these features were computed separately in both directions (dir.), i.e., for both uplink (ul) and downlink (dl) traffic. In addition, the session duration and four downlink-only features were added, namely, the average throughput in the first 1 s, 5 s, and 10 s, and the active ratio, i.e., the share of time spent downloading. This results in 109 features in total, which are summarized in Table I.

The recorded application-level information allowed the computation of application-level KPIs, which serve as labels. These QoE metrics include the initial delay, the number and duration of stalling events, the played-out video resolution, and the number of quality changes. Moreover, the requested video formats (itags) were extracted from the network log of the browser, and the YouTube API was queried to obtain the average bitrate of the downloaded video content. Eventually,

[1] https://www.selenium.dev/

[2] https://www.wireshark.org/docs/man-pages/tshark.html

[3] https://kaitai.io/

TABLE I: Overview of recorded data from stream of encrypted packets and the derived features.

recorded data	derived features
packet	count (dir.: ul/dl; size: all/>100 B)
packet size	volume (dir.: ul/dl), distribution (dir.: ul/dl; size: all/>100 B)
packet IAT	distribution (dir.: ul/dl)
time slot volume	distribution (dir.: ul/dl; slot length: 100 ms/1 s/10 s)
throughput	avg. session throughput (dir.: ul/dl), avg. start phase throughput (dir.: dl, phase length: 1 s/5 s/10 s)
duration	session

TABLE II: Overview of classification targets per QoE metric and respective split conditions.

QoE metric	#	split conditions
MOS	2	high (>3.5), low (≤3.5)
avg. resolution	3	high (≥700 p), mid (≥400 p ∧ <700 p), low (<400 p)
stalling	2	false (no stalling), true (contains stalling)
initial delay	2	short (<2 s), long (≥2 s)
initial delay	3	short (<2 s), mid (≥2 s ∧ <10 s), long (≥10 s)
avg. bitrate	2	high (≥500 kbps), low (<500 kbps)

all information was used to compute a Mean Opinon Score (MOS) using the standardized ITU-T P.1203 QoE model [14]. As we resort to classification tasks, the KPIs were discretized into classes, and those classes were used as labels during the training of the machine learning models. Thus, the final classification tasks are binary and ternary classification of initial delay, binary classifications of average bitrate, MOS, and whether the video contained stalling, as well as ternary classification of average resolution. The overview of the number and the definition of classes for each KPI is given in Table II. The dataset used further on consists of samples corresponding to each video being played, where each sample is represented with calculated network traffic features and ground-truth QoE/KPI classes.

C. Analysis procedure

To analyze how network characteristics and location-related streaming service specifics influence in-network QoE/KPI classification models, we evaluate three types of models:

1) *Network-specific models* - trained and tested on separate splits of a dataset collected in the same network (separate models for Würzburg and Zagreb data). The performance of these models serves as a baseline for comparison with other models.
2) *General models* - trained and tested on separate splits of the merged dataset (one model for both Würzburg and Zagreb data). General models may prove to be more robust, and they simplify the model training, deployment, and maintenance, as both locations can be handled by the same models.
3) *Cross-network applicable models* - trained on a dataset collected in one location and tested on data from the other location (referred to as *cross-test*). The aim here is to check whether trained models can be reused at new locations, thus significantly reducing the data collection efforts. Besides simply repeating the training procedure as it would be performed for network-specific models, in the training of cross-network applicable models various methods could be utilized to reduce the effect of network differences.

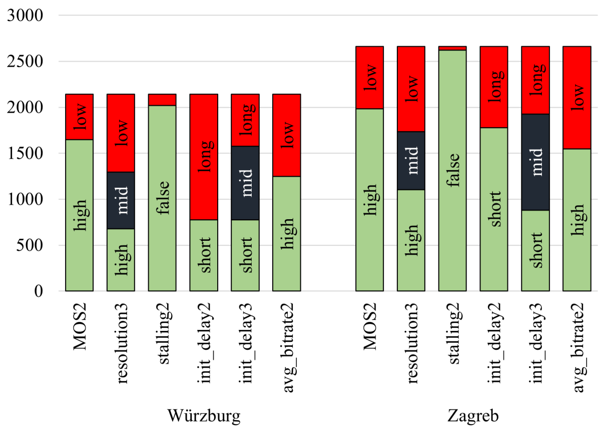


Fig. 1: Distribution of samples across the datasets.

In this analysis, we removed all the samples which included the use of the ad blocking extension and the samples which contained an advertisement. By doing so, we eliminate the potential impact of different ad strategies enforced by YouTube at different locations, and focus exclusively on heterogeneity originating from the network. Following the elimination of these samples, as well as samples containing NA values, the considered datasets contained 2142 samples collected in Würzburg (corresponding to 2142 videos being played) and 2661 samples collected in Zagreb. The samples contain calculated network traffic features and are labeled with QoE/KPI classes, as specified in Section III-B. The number of samples in each class is shown in Figure 1 for both datasets. For most classification targets, each class contains at least roughly 500 samples. The exception is stalling classification, where stalling (re-buffering) events rarely occurred at any of the locations.

In the case of network-specific models, for each network/location and for each target class, we first subsample the classes with respect to the least populated class (i.e., if the least populated class contains n samples, we randomly select n samples from each of the remaining classes). The resulting dataset is then randomly split into train and test sets (67% : 33%). Using the Sequential Feature Selection (SFS)[4] method on the train set, we select the 10 most relevant features to be used in the model training step. Once relevant features are selected, a model is trained on the train set using the Random Forest (RF) algorithm with 10 trees and maximum depth of 5. The choice of methods, algorithms, and parameters was made based on related work [18], [21], [29], but future work may include a systematic study of model performance with respect to different methods and parameters being used. The models are tested on the test set. We emphasize once again that the models are trained separately for the two locations.

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

The same procedure is applied for general models for each target class, but on a merged dataset (i.e., containing samples collected in both Würzburg and Zagreb). We split the merged dataset into train and test sets, ran SFS on the train set, and trained RF models using 10 SFS–selected features.

For cross-network applicability tests, we use a dataset collected in one location as a train set and the dataset collected in the other location as a test set. In the train set, we balance out the number of samples across classes as described above, while we keep the test set as is. Using SFS on the train set, 10 relevant features are selected and the RF model is trained. The model is tested on a separate dataset, collected in the other location. This is done for both combinations, with each dataset acting once as train and once as test set, respectively.

IV. Results

In this section, we present the results in terms of the performance of trained models on the corresponding test set, as defined in Section III-C. The classification performance report, displaying model precision, recall, and accuracy, is given in Table III. The rows of the table present results for different KPIs, while the columns indicate the model type (network-specific models for both locations, cross-tested models for both combinations of train and test locations, and a general model). The performance of the models is also visualized with an F-measure heatmap in Figure 2. Further analysis of the results is provided below, separately for each model type.

A. Network-specific models

As shown in Table III, the models for initial delay classification perform significantly better than any of the other models, with up to 99% accuracy. In particular, the identification of short initial delays performs well. The analysis of the features used by the models shows that the shape of initial bursts in downlink traffic, associated with the initial buffering of video content, is highly indicative of the initial delay duration. For binary classification, downlink throughput in the first second is the most important feature, while for ternary classification, the feature selection algorithm chose throughput in a longer period of 10 seconds. We also note that maxima of slotted data amounts highly correlate with initial delays, as these maxima typically occur at the beginning of the session, and in that sense also describe initial bursts. Features describing these maxima were seen to be employed by Zagreb network-specific initial delay classification models.

MOS and resolution classification models greatly relied on uplink packet length features. Most uplink packets in HAS are acknowledgements. However, larger packets contain requests for video segments and possibly feedback on streaming performance. Looking into the features related to uplink packet lengths, significant differences can be seen among the two datasets, i.e., networks. Figure 3a shows the distribution of the largest observed packets towards YouTube video servers across all sessions for both datasets. It can be seen that the largest uplink packet in Würzburg data is considerably larger for most sessions, compared to those in Zagreb data. On the other hand, the average uplink packet sizes (taking only packets longer than 100 B into account) appear to be generally larger in Zagreb data. These features were also found to be valuable in the Würzburg network-specific stalling classification model, while for Zagreb stalling classification model, we refrain from making any conclusions due to severe class imbalance and consequent poor model performance.

Both network-specific models for the estimation of average video bitrate rely a lot on mean downlink packet interarrival times. While video bitrates are known to correlate highly with the amounts of downlink traffic, interarrival times can carry the same amount of information for the prediction. The higher the video encoding bitrate, the larger the video segments, and consequently, more data needs to be downloaded, resulting in more packets and shorter interarrivals. The distribution of this feature for both datasets is given in Figure 3c, exhibiting similar shape in both networks.

B. General models

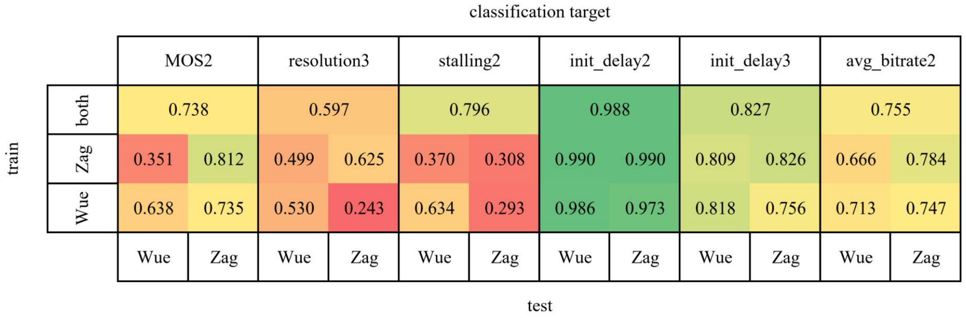
The performance of general models is consistently comparable to the performance of network-specific models for all targets. The general initial delay classification models rely on the throughput at the beginning of the session, while average bitrate classification models employ data-amount and interarrival-time features, as in the network-specific case. Contrarily, the selection of features for MOS and resolution classification is different. In the general models, downlink traffic features were considered to be more important, as opposed to network-specific MOS and resolution classification models which used mostly uplink traffic features. We attribute this to the differences in distributions of uplink features in the two datasets (cf. Figures 3a and 3b), which makes these features less relevant in the context of the merged dataset. We again refrain from making any conclusions about the stalling classification model, which shows misleadingly high accuracy due to class imbalance.

C. Cross-network applicability

While general models inherently learn about the differences among the datasets, and thus, exhibit performance similar to that of network-specific models, cross-tested models perform significantly worse for some KPIs. In case of initial delay classification, which largely depends on initial throughput features, the performance is not reduced. The same goes for average bitrate classification, which mostly relies on downloaded data amounts and interarrival times. These features display similar properties for the two datasets (cf. Figure 3c), and the high cross-network performance for initial delay and bitrate classification may be attributed to the used features. On the other hand, in MOS and resolution classification, the models are based on uplink features that display certain dissimilarities in their distributions for the two datasets (cf. Figures 3a and 3b). This reflects on the cross-network performance of models, making it clear that models trained on data collected in one network may not be applicable in another network without any intermediate intervention.

TABLE III: QoE/KPI classification model performance report for network-specific, cross-tested, and general models.

	Model type (datasets used for training and testing)					
	network-specific		cross-tested		general	
	Wue	Zag	Wue → Zag	Zag → Wue	merged dataset	
MOS: high/low	h:0.62 l:0.69	h:0.85 l:0.81	h:0.91 l:0.56	h:0.93 l:0.26	h:0.76 l:0.73	Precision
	h:0.68 l:0.63	h:0.82 l:0.85	h:0.79 l:0.77	h:0.18 l:0.95	h:0.73 l:0.76	Recall
	0.656	**0.832**	**0.786**	**0.356**	**0.745**	Accuracy
resolution: high/mid/low	h:0.80 m:0.48 l:0.53	h:0.78 m:0.52 l:0.59	h:0.44 m:0.49 l:0.35	h:0.54 m:0.43 l:0.56	h:0.82 m:0.50 l:0.57	Precision
	h:0.41 m:0.34 l:0.87	h:0.66 m:0.55 l:0.66	h:0.01 m:0.12 l:0.95	h:0.56 m:0.26 l:0.70	h:0.63 m:0.34 l:0.81	Recall
	0.564	**0.624**	**0.363**	**0.530**	**0.627**	Accuracy
stalling: True/False	t:0.54 f:0.78	t:0.44 f:0.00	t:0.82 f:0.77	t:0.01 f:0.98	t:0.07 f:0.96	Precision
	t:0.79 f:0.52	t:1.00 f:0.00	t:0.76 f:0.83	t:0.57 f:0.39	t:0.71 f:0.45	Recall
	0.634	**0.444**	**0.393**	**0.460**	**0.786**	Accuracy
initial delay: short/long	s:1.00 l:0.97	s:0.98 l:1.00	s:0.94 l:0.99	s:1.00 l:0.99	s:0.99 l:0.99	Precision
	s:0.97 l:1.00	s:1.00 l:0.98	s:0.99 l:0.97	s:0.98 l:1.00	s:0.99 l:0.98	Recall
	0.986	**0.990**	**0.976**	**0.991**	**0.988**	Accuracy
initial delay: short/mid/long	s:0.99 m:0.74 l:0.78	s:0.97 m:0.74 l:0.77	s:0.94 m:0.77 l:0.57	s:1.00 m:0.75 l:0.69	s:0.97 m:0.76 l:0.79	Precision
	s:0.97 m:0.88 l:0.59	s:1.00 m:0.70 l:0.78	s:1.00 m:0.55 l:0.77	s:0.98 m:0.79 l:0.66	s:0.99 m:0.85 l:0.70	Recall
	0.836	**0.841**	**0.759**	**0.822**	**0.837**	Accuracy
average bitrate: high/low	h:0.81 l:0.65	h:0.86 l:0.73	h:0.85 l:0.66	h:0.74 l:0.59	h:0.81 l:0.70	Precision
	h:0.62 l:0.83	h:0.69 l:0.88	h:0.69 l:0.83	h:0.67 l:0.66	h:0.81 l:0.70	Recall
	0.714	**0.785**	**0.747**	**0.670**	**0.755**	Accuracy



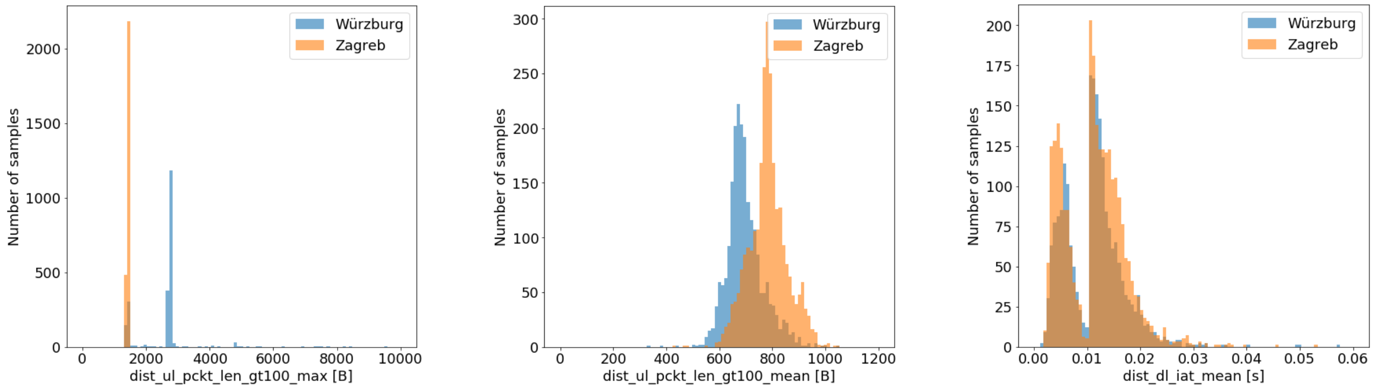
	classification target											
train	MOS2		resolution3		stalling2		init_delay2		init_delay3		avg_bitrate2	
both	0.738		0.597		0.796		0.988		0.827		0.755	
Zag	0.351	0.812	0.499	0.625	0.370	0.308	0.990	0.990	0.809	0.826	0.666	0.784
Wue	0.638	0.735	0.530	0.243	0.634	0.293	0.986	0.973	0.818	0.756	0.713	0.747
test	Wue	Zag	Wue	Zag	Wue	Zag	Wue	Zag	Wue	Zag	Wue	Zag

Fig. 2: F-measure heatmap summarizing the performance of network-specific, general, and cross-tested models.

As a first step towards potentially improving the cross-network performance of QoE/KPI classification models, we try two different methods: 1) identify features whose distributions vary across datasets and eliminate those during the model training phase, and 2) enrich the train set with small portions of data from the test location (and remove that particular portion from the test set).

We test the applicability of methods commonly used in dataset drift detection and handling when suspecting covariate shift (shift in independent variables, i.e., features) [36], [37]. The main idea of shift identification is to merge the features from both datasets and use the origin (dataset label) as a target variable. If the features possess the ability to separate the merged dataset and classify samples based on their dataset origin, then the features' distributions are shifted. We evaluate one feature at a time in the prediction of the origin and record the value of ROC-AUC for each feature. The greater the value, the feature is better at distinguishing the origin of a sample, and is thus drifting.

The top drifting features with scores above 0.8 according to the ROC-AUC metric indicate differences in packet length maximums (both uplink and downlink), but also differences in amounts of uplink data and interarrivals of uplink packets. We retrained the models, as described in Section III-C, with drifting features eliminated prior to the feature selection step. We focus on MOS and resolution classification, as these cross-tests were affected by drifting features. However, as depicted in Figure 4, our initial tests with dropping of drifting features indicate that the shifts in the two datasets may be more complex and that the two datasets may have completely distinct relationships between the features and the target variables, thus requiring the application of more complex drift handling

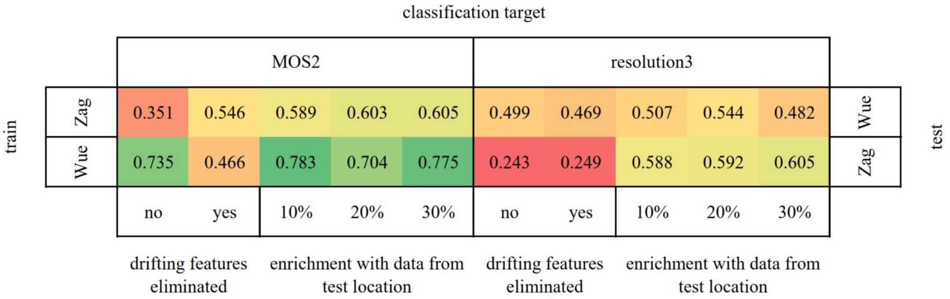


(a) Maximum uplink packet length.

(b) Average uplink packet length for packets larger than 100 B.

(c) Average downlink interarrival time.

Fig. 3: Distribution of values for selected features in both datasets.


classification target

train	MOS2					resolution3					test
	drifting features eliminated		enrichment with data from test location			drifting features eliminated		enrichment with data from test location			
	no	yes	10%	20%	30%	no	yes	10%	20%	30%	
Zag	0.351	0.546	0.589	0.603	0.605	0.499	0.469	0.507	0.544	0.482	Wue
Wue	0.735	0.466	0.783	0.704	0.775	0.243	0.249	0.588	0.592	0.605	Zag

Fig. 4: F-measure heatmap for cross-tested models with and without the elimination of drifting features, cross-tested models for which the training set was enriched with data from the test location.

methods in future work.

In the other method, we moved certain amounts of samples from the test set to the train set, thus adding data from the test location to data collected at the train location. Concretely, these amounts equaled to 10%, 20%, and 30% of the amount of samples in the train set. The results are presented in Figure 4, showing improved performance.

V. Conclusion and Outlook

The widespread adoption of end-to-end encryption in HAS has resulted in a surge of research efforts aimed at estimating application-level streaming performance from IP-level traffic patterns. These efforts are generally focused on individual use cases proving the potential of ML methods for estimating QoE from encrypted traffic. However, diverse combinations of used networks, end-devices, applications, protocols, etc., result in distinctive patterns in video streaming traffic, potentially requiring separate and specific models for each use case.

With the intention of reducing the complexity of QoE/KPI estimation and offering more robust solutions, we train and test general models (applicable for multiple networks), and test the cross-network applicability of network-specific models (trained on data collected in one network and tested on another). The results show that the performance of general models is comparable to the performance of models trained and tested on data collected in a single network. In that sense, it is possible to reduce the model training efforts and use a single model in multiple environments.

Cross-applicable models would more significantly reduce the needed efforts, focusing on easing the exhaustive data collection, which is one of the key challenges of ML approaches. The results with respect to cross-tested models prove that ML–based models may work well only for the specific use cases they were trained for, while using trained models for other use cases might lead to performance degradation and is not recommended in practice. The results emphasize the need for exploring more complex methods that would potentially improve the cross-network performance of models. As a possible way forward to addressing this challenge, in our future work we plan to investigate methods from the domain of deep learning and transfer learning [38], [39].

Acknowledgment

This work has been supported in part by the Croatian Science Foundation under the project IP-2019-04-9793 (Q-MERSIVE).

References

[1] Cisco, “Cisco Annual Internet Report (2018-2023) White Paper,” Cisco, Tech. Rep., 2020.

[2] 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 Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 2s, pp. 1–29, 2018.

[3] Ericsson, “Ericsson Mobility Report,” Ericsson, Tech. Rep., 2019. [Online]. Available: https://www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf

[4] 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, pp. 1–6.

[5] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, “Measuring Video QoE from Encrypted traffic,” in *Proceedings of the 2016 Internet Measurement Conference*, 2016, pp. 513–526.

[6] 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.

[7] 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 *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2019, pp. 76–81.

[8] P. Le Callet, S. Möller, and A. Perkis (eds), “Qualinet White Paper on Definitions of Quality of Experience,” European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), Lausanne, Switzerland, Tech. Rep., 2013, version 1.2.

[9] T. Hoßfeld, R. Schatz, M. Varela, and C. Timmerer, “Challenges of QoE Management for Cloud Applications,” *IEEE Communications Magazine*, vol. 50, no. 4, pp. 28–36, 2012.

[10] R. Schatz, M. Fiedler, and L. Skorin-Kapov, “QoE-based Network and Application Management,” in *Quality of Experience*. Springer, 2014, pp. 411–426.

[11] S. Baraković and L. Skorin-Kapov, “Survey and Challenges of QoE Management Issues in Wireless Networks,” *Journal of Computer Networks and Communications*, vol. 2013, no. 165146, pp. 1–28, 2013.

[12] Q. M. Qadir, A. A. Kist, and Z. Zhang, “Mechanisms for QoE Optimisation of Video Traffic: A Review Paper,” *Australasian Journal of Information, Communication Technology and Applications*, vol. 1, no. 1, pp. 1–18, 2015.

[13] 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.

[14] International Telecommunication Union, “ITU-T Recommendation P.1203: Parametric Bitstream-based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services over Reliable Transport,” 2017. [Online]. Available: https://www.itu.int/rec/T-REC-P.1203/en

[15] ——, “ITU-T Recommendation P.1204: Video Quality Assessment of Streaming Services Over Reliable Transport for Resolutions up to 4K,” 2020. [Online]. Available: https://www.itu.int/rec/T-REC-P.1204-202001-P/en

[16] P. Casas, M. Seufert, F. Wamser, B. Gardlo, A. Sackl, and R. Schatz, “Next to You: Monitoring Quality of Experience in Cellular Networks from the End-devices,” *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 181–196, 2016.

[17] 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 *Proceedings of the 9th International Conference on Quality of Multimedia Experience (QoMEX)*, Erfurt, Germany, 2017.

[18] W. Pan, G. Cheng, H. Wu, and Y. Tang, “Towards QoE Assessment of Encrypted YouTube Adaptive Video Streaming in Mobile Networks,” in *Proceedings of the 24th IEEE/ACM International Symposium on Quality of Service (IWQoS)*, Vilanova i la Geltrú, Spain, 2016.

[19] 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 *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018, pp. 233–239.

[20] M. H. Mazhar and Z. Shafiq, “Real-time Video Quality of Experience Monitoring for HTTPS and QUIC,” in *INFOCOM 2018 - Conference on Computer Communications*. IEEE, 2018, pp. 1331–1339.

[21] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, “ViCrypt to the Rescue: Real-Time, Machine-Learning-Driven Video-QoE Monitoring for Encrypted Streaming Traffic,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2007–2023, 2020.

[22] 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 *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019, pp. 48–59.

[23] I. Orsolic, L. Skorin-Kapov, and T. Hoßfeld, “To Share or Not to Share? How Exploitation of Context Data Can Improve In-Network QoE Monitoring of Encrypted YouTube Streams,” in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2019, pp. 1–3.

[24] I. Bartolec, I. Orsolic, and L. Skorin-Kapov, “Inclusion of End User Playback-Related Interactions in YouTube Video Data Collection and ML-Based Performance Model Training,” in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2020, pp. 1–6.

[25] 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 *Proceedings of the 4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks*, 2019, pp. 7–12.

[26] D. Moura, M. Sousa, P. Vieira, A. Rodrigues, and M. P. Queluz, “A No-Reference Video Streaming QoE Estimator based on Physical Layer 4G Radio Measurements,” in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6.

[27] J.-M. Martinez-Caro and M.-D. Cano, “On the Identification and Prediction of Stalling Events to Improve QoE in Video Streaming,” *Electronics*, vol. 10, no. 6, p. 753, 2021.

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

[29] I. Orsolic and L. Skorin-Kapov, “A Framework for In-Network QoE Monitoring of Encrypted Video Streaming,” *IEEE Access*, vol. 8, pp. 74 691–74 706, 2020.

[30] 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,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–25, 2019.

[31] S. Ickin, K. Vandikas, F. Moradi, J. Taghia, and W. Hu, “Ensemble-based Synthetic Data Synthesis for Federated QoE Modeling,” in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 72–76.

[32] S. Ickin, M. Fiedler, and K. Vandikas, “Customized Video QoE Estimation with Algorithm-Agnostic Transfer Learning,” *arXiv preprint arXiv:2003.08730*, 2020.

[33] M. Seufert, R. Schatz, N. Wehner, and P. Casas, “QUICker or not? - an Empirical Analysis of QUIC vs TCP for Video Streaming QoE Provisioning,” in *3rd International Workshop on Quality of Experience Management (QoE-Management)*, 2019.

[34] 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 *2015 European Conference on Networks and Communications (EuCNC)*. IEEE, 2015, pp. 239–243.

[35] M. Seufert, “Quality of Experience and Access Network Traffic Management of HTTP Adaptive Video Streaming,” Doctoral Thesis, University of Würzburg, 2017. [Online]. Available: https://opus.bibliothek.uni-wuerzburg.de/files/15413/Seufert_Michael_Thomas_HTTP.pdf

[36] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with Drift Detection,” in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[37] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[38] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[39] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A Survey of Transfer Learning,” *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.