Improving the Transfer of Machine Learning-Based Video QoE Estimation Across Diverse Networks

Michael Seufert, *Member, IEEE,* and Irena Orsolic

***Abstract*—With video streaming traffic generally being encrypted end-to-end, there is a lot of interest from network operators to find novel ways to evaluate streaming performance at the application layer. Machine learning (ML) has been extensively used to develop solutions that infer application-level Key Performance Indicators (KPI) and/or Quality of Experience (QoE) from the patterns in encrypted traffic. Having such insights provides the means for more user-centric traffic management and enables the mitigation of QoE degradations, thus potentially preventing customer churn. The ML–based QoE/KPI estimation solutions proposed in literature are typically trained on a limited set of network scenarios and it is often unclear how the obtained models perform if applied in a previously unseen setting (e.g., if the model is applied at the premises of a different network operator). In this paper, we address this gap by cross-evaluating the performance of QoE/KPI estimation models trained on 4 separate datasets generated from streaming 48000 video streaming sessions. The paper evaluates a set of methods for improving the performance of models when applied in a different network. Analyzed methods require no or considerably less application-level ground-truth data collected in the new setting, thus significantly reducing the extensiveness of required data collection.**

***Index Terms*—video streaming, traffic encryption, machine learning.**

I. Introduction

Video streaming services are held accountable for the largest portion of globally generated network traffic. According to Ericsson Mobility Report from Nov. 2021 [1], video traffic is estimated to account for 69 percent of all mobile data traffic and its share is expected to increase to 79 percent by 2027. During the same time, total global mobile data traffic is expected to grow 4.4 times. Additional strain on networks has been put amid the COVID-19 pandemic. Dramatic surges in network traffic have been observed corresponding to web conferencing, video-on-demand and gaming services [2], [3], with billions of people opting for or being forced into using online meetings, education, entertainment, etc.

In the context of such immense amounts of generated network traffic, it is becoming more and more important to manage both the services and the networks in a way to efficiently use available resources while keeping the customers satisfied. This requires the understanding of how measureable network- and application-level parameters influence end-users' Quality of Experience (QoE) and thereupon invoking QoE-aware service/network management mechanisms [4], [5]. A special research focus in this direction has been put on video streaming sevices, due to their impact on the global network traffic. Aiming to meet the users' expectations, streaming services may employ various quality adaptation strategies (e.g., adaptive streaming in compliance with MPEG-DASH [6]), utilize increasingly efficient compression techniques [7], etc. Management mechanisms from the network perspective may include, for example, QoE-aware resource (re)allocation and (re)routing [8]. While the impact of application-level parameters onto QoE is described in literature and embodied in existing QoE models [9], [10], the relationship between network-level metrics and QoE is far more complex to unveil, particularly given the widespread use of encryption.

Motivated by the high interest from the industry, plenty of research efforts have been put into describing the relationship between network-level metrics and video streaming performance [11]–[21]. In these studies, streaming performance is commonly expressed in terms of QoE and/or application-level key performance indicators (KPI) such as startup delay, video encoding bitrate, and video resolution. Most of aforementioned studies are exploiting methods from the domain of machine learning (ML) in order to map the network traffic patterns to quality degradations perceived by users. While proposed approaches have proven to perform well on individual use-cases focused on a particular streaming service and when applied on a dataset collected in one particular experiment setup, it is often unclear how well they generalize across various use-cases. In this paper we go beyond related work by investigating the applicability of such ML models across different network setups and over longer periods of time.

The **goals of this study** are 1) to investigate how well QoE/KPI estimation models perform when applied on data collected in a different network setting and in a different time period, and 2) to propose potential solutions for adapting the models in order to perform well in new setups and over extended periods of time. The paper evaluates various data science methods for detecting and eliminating dissimilarities in data from different sources, with the aim to reduce the extensiveness of data collection while developing and maintaining accurate QoE/KPI estimation solutions. We believe that this research is of high interest to companies developing user experience analytics solutions based on ML, as the presented analysis and methods may help them reduce the time to customize the solution for a specific network as well as reduce maintenance costs.

The contributions of the paper can be summarized as follows:

1) **Large video on demand streaming datasets.** During this study, four datasets were collected using measurement setups

M. Seufert is with the University of Augsburg, Chair of Networked Embedded Systems and Communication Systems, Augsburg, Germany. During this work he was with the University of Würzburg, Chair of Communication Networks, Würzburg, Germany, e-mail: michael.seufert@uni-a.de

I. Orsolic was with the University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia, e-mail: irena.orsolic@outlook.com

described in Section III-A. The datasets were collected during the period from July 2020 to August 2021 and differ in the location where the measurement campaign was run (Würzburg, Zagreb) as well as in the type of the access network (Ethernet, Wi-Fi). All datasets include data corresponding to the same set of 2000 distinct videos being streamed to a laptop, with and without using an ad-blocking plugin, under 3 different bandwidth constraints. This results in 48000 streamed video sessions. The description of the dataset is given in Section IV and we will publish the processed dataset (ready for ML analysis) online[1].

2) **Dataset analysis.** We analyzed and compared the datasets in terms of both application-level data (Section IV-A) and network traffic features. In Section IV-B we analyze network traffic features on a per-video level, while in Section IV-C network traffic features are analyzed on a per-second level. We note that in this paper we consider session-level (per-video) classification only, but both types of datasets are made publicly available, and thus described.

3) **Session-level QoE/KPI estimation models.** We trained numerous models for estimating QoE, startup delay, video resolution, video bitrate, and rebuffering occurrence. For this, we focus on shallow learning only, which proved to work well for QoE/KPI estimation in related works. Moreover, since QoE datasets are typically small due to the expensive dataset creation, the added value and utility of training deep learning models might be very small, such that we do not consider deep learning in this work. Using a systematic approach, we compare *i)* the performance of network-specific models (trained and tested on a single dataset), *ii)* the performance of general models (trained and tested on all datasets merged), and *iii)* the cross-applicability performance of network-specific models (trained on a single dataset and tested on all other datasets separately). The models are trained and their performance is analyzed for session-level (per-video) QoE/KPI estimation (Section VI).

4) **Analysis of methods for improving the model cross-applicability.** In order to reduce the exhaustiveness of data collection for model training purposes, it would be valuable if once trained models could be reused for other use-cases (e.g., different networks/locations) besides the one they were specifically trained for. Thus, we aim to investigate potential solutions for improving the cross-applicability of the models (Section V-C). We apply and evaluate methods based on scaling, decomposition, manifold learning, ML-based feature representation transfer, drift elimination, and enrichment (Section VI).

The initial results of the study were published in [22], where we trained and tested network-specific, cross-network, and general models for session-level QoE/KPI classification on a smaller sample of around 5000 videos collected in 2020. This paper presents a much more comprehensive analysis of cross-network model applicability and extends the initial study by using a 10 times larger and more diverse dataset, giving a deeper insight into the differences in data across measurement scenarios. This paper builds on top of previous work by assessing various additional methods for improving model cross-applicability.

The paper is organized as follows. Section II outlines background and related work on ML–based in-network QoE/KPI estimation models, applicability of such models across different usage settings, and potential for model adaptation and transfer. Section III describes the measurement setup, data collection campaigns, and data processing. The dataset is then portrayed and analyzed in Section IV. Section V presents the modeling procedure and methods for improving model transfer. Results are shown and discussed in Section VI, followed by a conclusion and outlook in Section VII.

II. Background and Related Work

A. In-network video streaming QoE/KPI estimation

The idea of applying ML for estimating QoE and video streaming KPIs in the network was suggested in [11]. The proposed Prometheus approach outperformed existing solutions that required control over the app services and domain expertise. Soon thereafter, many of the popular video streaming services started introducing encryption to their flows, thus making some of the network traffic features needed as input inaccessible. Hence, the study in [15] relied exclusively on features obtainable from the encrypted traffic, but the application-level ground-truth was still derived from non-encrypted flows captured at a web-proxy. Building on top of these ideas, the studies published in [23], [24] resulted with approaches fully applicable in the context of encrypted traffic, i.e, not needing the access to packet payloads at any phase of the model training. These studies have proven the feasibility of classifying YouTube video streams into QoE classes in a per-video (session-level) manner based only on the statistical properties of the encrypted traffic volume.

In parallel with the further development of session-level QoE/KPI estimation solutions [25], [26], a number of real-time KPI estimation approaches were proposed [13], [16], [18]–[20], [27]–[29]. Such solutions, as opposed to session-level ones, might be more appropriate for network operators looking to dynamically manage QoE and optimize resource allocation. Session-level approaches, on the other hand provide per-video session metrics and may be more appropriate for network planning purposes.

Focusing on the deployment of these ML–based models in 5G networks, in [30], [31] the authors present their simulations used to assess such models when embedded in NWDAF (Network Data Analytics Function) – an analytics entity in 5G with machine learning capabilities [32], [33].

Another interesting research avenue on QoE monitoring for HTTP adaptive streaming (HAS) is the inclusion of user behavior and its impact on traffic patterns and, consequently, on model performance. In [21], [34], the authors explore this important issue, given its relation to deploying robust QoE monitoring solutions in the network.

B. Applicability of QoE/KPI estimation models across different scenarios

Related work has barely scratched the surface of the problem of inherent dimensionality originating from the variety of

[1] https://urn.nsk.hr/urn:nbn:hr:168:227338

possible video streaming usage scenarios. The cross-testing efforts described in [25], [35] 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 [25]. 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 [20].

Similar conclusions, but focused on different services and not platforms, have been found in [18]. 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 [36], [37], 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.

C. Adaptation and transfer of QoE/KPI estimation models

In real-life applications, data used as input for prediction models often changes over time, making the performance of the models degrade as newly generated data is presented to them. Similarly, there might be slight differences between the same type of data but generated by different sources. In general, this phenomenon is known as the *dataset shift* or *dataset drift* [38], [39]. Dataset shift can be divided into three categories: 1) *covariate shift* (shift in independent variables, 2) *prior probability shift* (shift in the target variable), and 3) *concept drift* (target concept depends on hidden contexts that are not explicitly provided to the learner algorithm) [40], [41]. Depending on the domain, concept drift can appear suddenly (sudden, abrupt, instantaneous concept drift), gradually (gradual concept drift), and temporarily disappear and reappear (reoccurring concept drift) [42]–[44].

The strategies of handling the drift differ among these types. Covariate shift can be detected using ML and the drifting features can be excluded from model training later on. This is done in the *Drift elimination* method, applied in this paper. For sudden drifts, a common approach is to retrain the model on instances captured both before and after the sudden drift occurrence [38], [45]. This principle is adopted in another method for improving transfer proposed in this paper: *Enrichment*. Aiming to reduce the drift between the datasets, we also test various feature transformation and dimensionality reduction techniques, including *Scaling*, *Decomposition*, *Manifold learning* [46], [47], and *ML–based feature representation transfer*. These methods are described in more detail in Section V-C.

III. Dataset Preparation

A. Measurement setup

The measurements were conducted using a Java–based framework similar to [16], [48]. The measurement framework is able to automatically start a Chrome browser using the Selenium browser automation tool[2]. 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 video streaming service 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[3]. Next, the browser accesses a single video page and injects a JavaScript–based monitoring script [49], [50] 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 videos 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, in a cable broadband network of an ISP in Würzburg, Germany, and in a cable broadband network of an ISP in Zagreb, Croatia. In all 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 Würzburg and Zagreb, in all three network conditions, both in 2020 and 2021, which results in a dataset of roughly 48000 streamed video sessions. The measurement runs were conducted over five months in 2020 (Jul. – Nov.) and over three months in 2021 (Apr. – Jun.). A summary of the datasets is given in Table I.

B. Dataset preprocessing

For each measurement run denoted with year (2020, 2021), location (Wue, Zag), bandwidth limitation setting (unlimited, 1 Mbps, stochastic), and ad-blocking plugin status (on, off), the raw datasets contain HTTP logs, measurement logs

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

TABLE I: Summary of collected datasets.

Label	Network description	Videos	Measurement scenarios
Wue_2020	Ethernet, high speed optical fiber, campus network	11373	Datasets include the same set of 2000 videos played under all combinations of following conditions. **Bandwidth limitations**: 1) unlimited, 2) 1 Mbps, 3) stochastic **Ad-blocking plugin**: 1) on, 2) off
Zag_2020	WiFi, cable broadband, home network	10993	
Wue_2021	Ethernet, cable broadband, home network	9927	
Zag_2021	Ethernet, cable broadband, home network	10914	

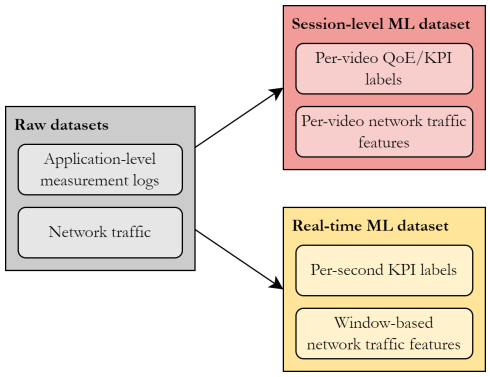


Fig. 1: The collected raw datasets are processed into two ML datasets that can be used for 1) session-level QoE/KPI estimation and 2) real-time KPI estimation.

(application-level events), and network traffic traces. From these logs, we generate two datasets that can later on be used to train the QoE/KPI estimation models. We refer to these datasets as *session-level ML dataset* and *real-time ML dataset*, as opposed to the term *raw datasets* which we use to describe initial logs. This is summarized in Figure 1. Both datasets are .csv files with rows containing network traffic features (statistical properties of the encrypted traffic) and QoE/KPI labels. While in the session-level ML dataset each row represents a single video with features and labels describing the whole video session, in the real-time ML dataset a row represents one second of a video streaming session. These two ML datasets are publicly accessible online[4] and briefly described in the continuation of the paper. Due to paper length limitations and limitations with regards to the computation environment, in this study we train models using session-level variant of the dataset only.

IV. Dataset Characteristics

Out of the scheduled 48000 video streams, 43207 were played, e.g., not skipped due to video being deleted or its availability settings changed over the course of the measurement campaigns. Out of these videos, for further analysis we used: 8833 videos from Wue_2020, 9410 from Zag_2020, 5310 from Wue_2021, and 6640 from Zag_2021. The videos were selected based on log consistency criteria in order to exclude incomplete logs and measurement errors. The ML datasets that we provide to the research community are filtered based on these criteria.

[4]https://urn.nsk.hr/urn:nbn:hr:168:227338

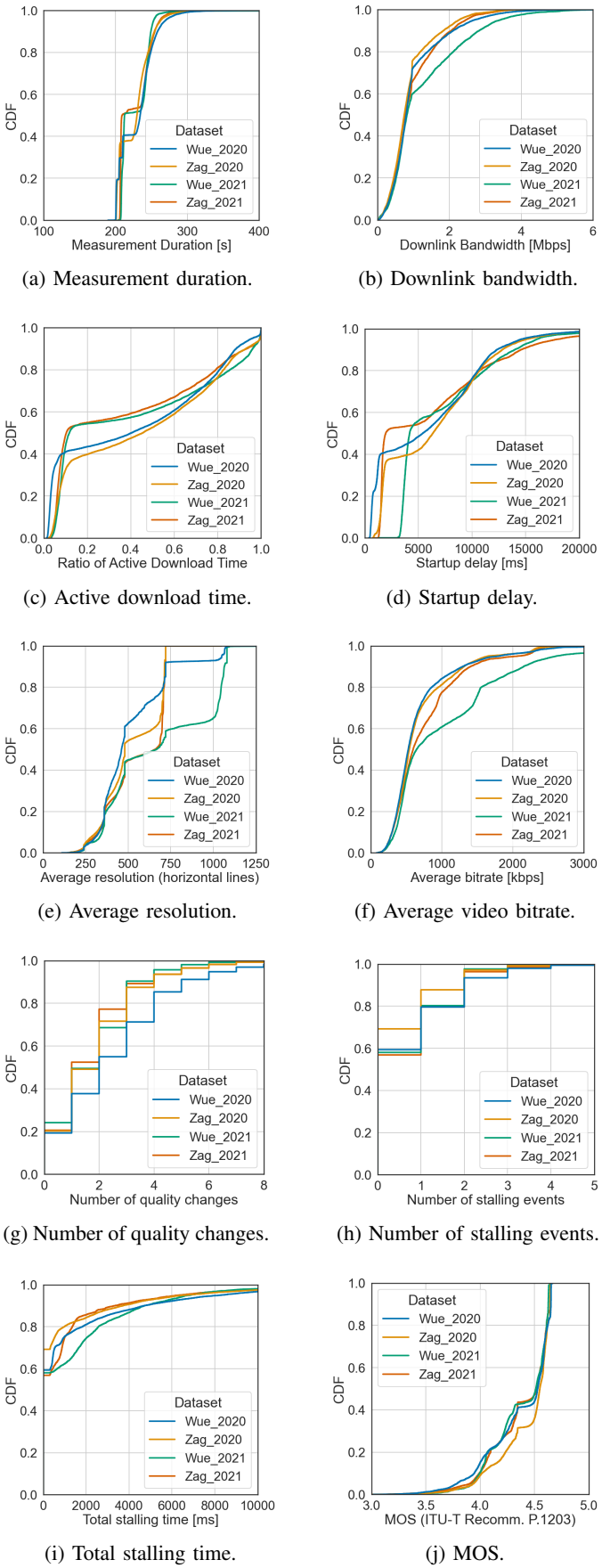
A. Application-level data

We observe in Figure 2a that measurement durations follow similar distributions in all datasets. The only notable difference is around 20 s offset for 10-12% of the videos, which could be attributed to a change of advertisement strategy between 2020 and 2021. In Figure 2b it can be seen that the bandwidth limitations were applied as configured, as roughly two thirds of the sessions have an average downlink bandwidth of at most 1 Mbps with overlapping CDFs. Regarding the sessions streamed with unlimited bandwidth, it can be seen that Wue_2021 had slightly higher average bandwidths than the other three datasets. However, the highest average bandwidth was be observed in the Wue_2020 data with 36.84 Mbps compared to 10.61 Mbps for Wue_2021, 4.37 Mbps for Zag_2021, and 4.33 Mbps for Zag_2020. When comparing the ratio of active download time and session duration in Figure 2c, we see a clear difference between 2020 and 2021 measurements, which indicates changes in the adaptive bitrate logic and resulting download behavior of the streaming service. Nevertheless, within each year, the distribution of active download ratios is similar across both locations.

We now look in more detail into the application-layer KPIs among the four collected datasets. While most curves have similar shapes across datasets, an interesting observation is that average video resolutions were often higher in Wue_2021 as compared to the other datasets (Figure 2e), consequently resulting in longer startup delays (Figure 2d) and higher average bitrates (Figure 2f). With the exception of Wue_2021, which has a large portion of videos played in 1080 p, most commonly observed resolutions were 480 p and 720 p (Figure 2e). Most of the average video encoding bitrates are under 1000 kbps and the values are very rarely exceeding 3000 kbps (Figure 2f). Stalling was noted in approximately 40% of the played videos (Figure 2h), rarely occurring multiple times in a single video, and half of the videos with stalling events had a total stalling time of less than 2 seconds (Figure 2i). MOS values in all datasets cover a range from roughly 3.0 to 4.7, with half of the values above 4.5 (Figure 2j).

B. Session-level network traffic features

We aggregated the packet-level traces - consisting of timestamp, src/dst IP address, src/dst port, protocol type, packet size - of each video session into a set of 109 features, which characterize the traffic using statistical descriptors, see Table II. These features were selected based on domain knowledge and related works, e.g., [27] and include session duration, as well as packet count, count of packets greater than



(a) Measurement duration. (b) Downlink bandwidth.

(c) Active download time. (d) Startup delay.

(e) Average resolution. (f) Average video bitrate.

(g) Number of quality changes. (h) Number of stalling events.

(i) Total stalling time. (j) MOS.

Fig. 2: Distributions of KPIs/QoE across datasets.

TABLE II: 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 III: Overview of classification targets per QoE metric and respective split conditions. *MOS and startup delay are only considered in session-level estimation

QoE metric	definition
MOS*	regression
startup delay*	short (<5 s), long (≥5 s)
avg. resolution	high (≥700 p), low (<700 p)
stalling	false (no stalling), true (contains stalling)
avg. bitrate	high (≥500 kbps), low (<500 kbps)

100 bytes, volume, and average throughput for both uplink and downlink traffic. We also add the active download ratio and the average downlink throughput in the first 1/5/10 seconds. Moreover, we consider the distribution of packet size, packet size of packets greater than 100 bytes, packet inter-arrival time, and data volume in 0.1/1/10-seconds time slots for both uplink and downlink traffic. From these distributions, we compute mean, variance, standard deviation, coefficient of variation, skewness, kurtosis, minimum, and maximum and also add these descriptors to the set of features.

The distribution of each feature was compared between all six pairs of datasets using the univariate Wald-Wolfowitz runs test and the univariate Kolmogorov-Smirnov test. We found that at most 14% (WW runs test), or 6% (KS test), respectively, of the columns do not show a significant difference with respect to the 5% significance level, and thus, can be considered similar. These findings persist even when scaling the data to standard scores, both when using a common scaling for both datasets or when using individual scaling per dataset. We additionally conducted the multivariate Friedman-Rafsky runs test and the multivariate Kolmogorov-Smirnov test according to [51] over all features. The tests were con-

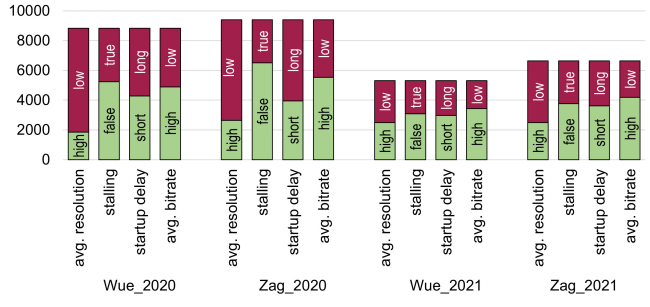


Fig. 3: Distribution of samples across datasets and classes for session-level QoE/KPI classification.

ducted on a 5% random sample of the datasets due to the computational complexity of the multivariate tests. Here, when using Euclidean pairwise distance, we find high p-values for some pairs, namely, for `Wue_2020` and `Wue_2021` ($p = 0.60$), `Wue_2020` and `Zag_2021` ($p = 0.34$), `Zag_2020` and `Wue_2021` ($p = 0.57$), `Zag_2020` and `Zag_2021` ($p = 0.65$), and `Wue_2021` and `Zag_2021` ($p = 0.09$). This suggests that these pairs of datasets are not too different from a statistical point of view. However, considering Mahalanobis distance or applying standard scaling minimizes p-values for all pairs to below 0.01, and thus, confirms that the session-level features become highly different when streaming videos from a different network. This can cause problems when applying ML methods trained on one dataset to another dataset, as we will showcase below. Thus, it is required to research and apply mechanisms which can improve the model transfer.

Considering the labels, the datasets contain all KPIs in Figure 2 (d)-(j), namely, startup delay, average resolution, average video bitrate, number of quality changes, number of stalling events, total stalling time, and MOS as estimated by the standardized QoE model from ITU-T Recomm. P.1203 [9]. We additionally apply a simple binary classification into high ($\geq$ 700p) or low average resolution, existing (true) or non-existing (false) stalling, short ($<$ 5 s) or long startup delay, and high ($\geq$ 500 kbps) or low average bitrate according to thresholds listed in Table III. The resulting distribution of samples across datasets and classes is depicted in Figure 3. It can be seen that all classes have a substantial amount of instances, with the minimum of 1861 instances.

C. Real-time network traffic features

The real-time network traffic features are based on the same raw data as the session-level features. However, here, the packet-level traces are split into small time slots of 1 second, which results in a dataset containing more than 5 million time slots. For each time slot, the traffic is described with a set of features and labeled with the corresponding application-level KPIs, namely, the current resolution and average bitrate of the played out representation and whether the video is currently stalling or not. When we apply binary classification using the same thresholds as above (Table III), this results in the distribution of samples across datasets and classes, which is depicted in Figure 4. Having these features and labels with one second granularity allows to train ML models, which can estimate the video streaming KPIs every second, and thus, allows for a fine-granular real-time tracing of the QoE.

The feature set is the same set that was used in [16], [17]. First, we compute packet count (total, uplink, downlink), traffic volume (total, uplink, downlink), uplink and downlink ratio of packet count and traffic volume, and number and volume as well as ratio of TCP and UDP packets. In addition, we consider the time from time slot start until the first uplink and downlink packet and time from the last uplink and downlink packet until time slot end, which give the burst duration (total, uplink, downlink) within the time slot. We compute the time slot throughput (total, uplink, downlink), burst throughput (total, uplink, downlink), as well as distributional statistics (mean, variance, standard deviation, coefficient of variation, skewness, kurtosis, minimum, maximum) for both the packet size and inter-arrival time distribution for uplink and downlink traffic. Finally, we compute slope and intercept of the regression line, which fits the cumulative upload and download volume and cumulative uplink and downlink inter-arrival times within the time slot. This results in a total of 69 features per time slot.

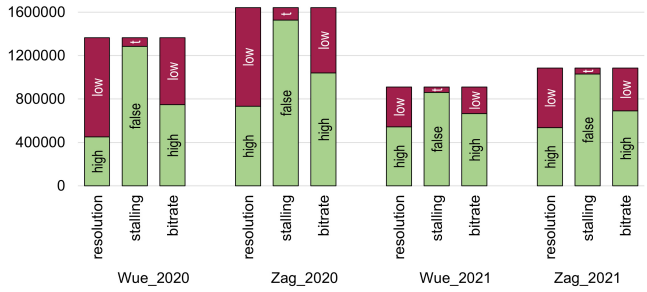


Fig. 4: Distribution of samples across datasets and classes for real-time KPI classification.

Additionally, we compute features to track trends as well as the overall state of the ongoing session. The trend features use a sliding window of length 3 seconds, i.e., three consecutive time slots. The time slots are considered as a single trend macro time slot of 3 seconds for which the same 69 features are computed as for the micro time slot of 1 second. In a similar fashion, we compute features of a session macro time slot, which are based on all past time slots including the current one. This results in a total of 208 features (time slot number + 69 time slot features + 69 trend features + 69 session features).

V. Methodology for Testing Cross-Applicability

The methodology for testing cross-applicability of models is summarized in Fig. 5, while the following subsections provide more detailed information about the depicted steps.

A. Model selection

To find the best model and hyperparameters, we follow a two stage approach using `scikit-learn`. First, we do a broad study on different model types and very few hyperparameter combinations. For this, we focus on each dataset individually and additionally filter out all videos, which contain an advertisement. We compare the performance of different classifiers for the binary classes described above. We undersample the classes to obtain a balanced dataset and avoid any preprocessing of the features, such as scaling. We perform an 80:20 training/test split, apply 3-fold cross-validation on the training set, and compare the performance of Gaussian Naive Bayes, Stochastic Gradient Descent, k-Nearest Neighbors, Decision Tree, Random Forest, Extremely Randomized Trees, Gradient Tree Boosting, XGBoost, and Support Vector Machine classifiers. Similarly, we study the performance for continuous P.1203 MOS estimation on the session-level datasets using Bayesian Ridge, Stochastic Gradient Descent, k-Nearest Neighbors, Decision Tree, Random

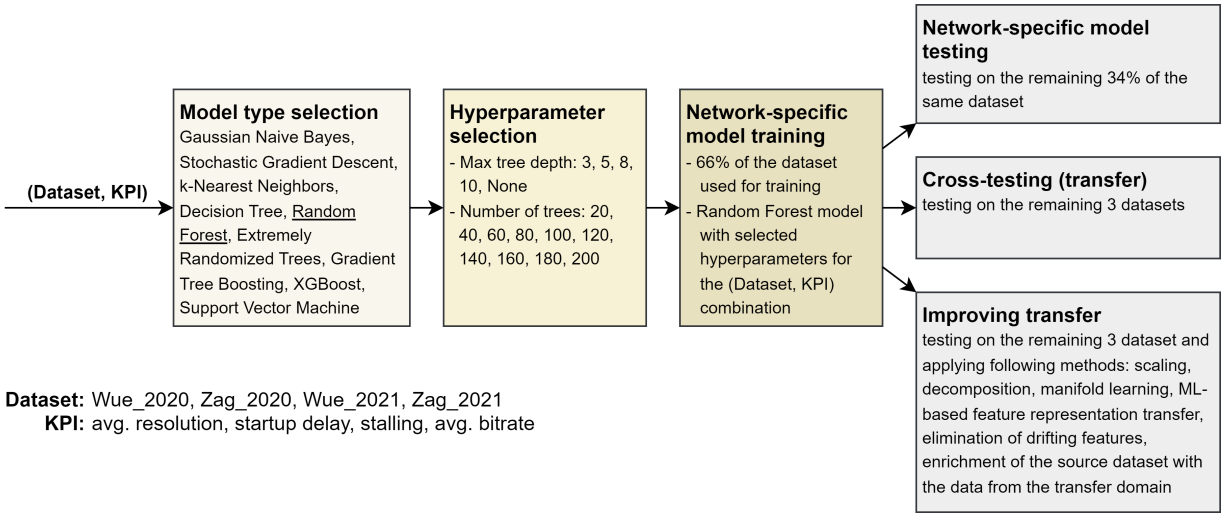


Fig. 5: Methodology for testing model cross-applicability, repeated for each combination of dataset and KPI.

Forest, Extremely Randomized Trees, Gradient Tree Boosting, XGBoost, and Support Vector Machine regressors.

Our results confirmed earlier findings, e.g., [17], that tree-based models provide the best trade-off between estimation accuracy and training and inference speed. In the study, we compared the obtained F1 scores for all algorithms. The results have shown that Random Forest was either outperforming all other algorithms or sharing the top result (when F1 score is rounded to 2 decimals) with one of the following: Support Vector Machine, XGBoost, Gradient Tree Boosting. For consistency purposes and to reduce the number of test iterations, we decided to test the subsequent steps of the methodology using Random Forest only. We also note that some algorithms, e.g., Support Vector Machine, gave comparable results to Random Forest, but are computationally much more intensive, additionally motivating not considering them further. Thus, we focus on Random Forest models for the rest of this work. As a second step of the hyperparameter study, we then focus exclusively on Random Forest and explore a larger set of hyperparameter combinations for that algorithm. This includes different max tree depth values of 3, 5, 8, 10, or None (no limit), and different numbers of trees, ranging from 20 to 200 in steps of 20. Additionally, we allow a feature selection using the SelectKBest algorithm, selecting either all features or the features with the k highest scores. Here, parameter k ranges from 5 to 50 in steps of 5, and scores are computed either based on mutual information or F-score.

B. Baseline establishment

Using Random Forest with maximum tree depth and number of trees determined in the hyperparameter study, we evaluate the performance in the following cases:

- Network-specific evaluation – Each model is trained on a subset (66%) of the dataset and tested on the rest (34%) of the same dataset. This results in 20 models, estimating 5 QoE metrics for 4 datasets.
- Cross-testing – Each of the 20 network-specific models (trained as defined above) are tested on the remaining 3 datasets.
- General model evaluation – Models are trained on a subset (66%) of the merged datasets (containing samples from all 4 datasets) and tested on the rest of the merged dataset (34%).

These results are then used as a baseline for assessing the methods for improving the transfer. Our expectation is that our methods for improving transfer would yield model performance between the one achieved with cross-testing and network-specific evaluation. The goal of the transfer improvement is to get as close as possible to the results obtained with network-specific and general models without needing all the data that was used for baseline models.

C. Improving transfer

For the remainder of this work, we will refer to the dataset, on which a model was originally trained, as the source dataset or source domain. We will consider the other dataset, on which the trained model is tested, as the transfer dataset or transfer domain. Aiming to improve the model transfer in comparison to the results obtained via cross-testing, we identify and assess methods based on scaling, decomposition, manifold learning, ML-based feature representation transfer, elimination of drifting features, and enrichment of the source dataset with the data from the transfer domain. This section describes the used methods.

Scaling. We apply min-max-scaling and standard scaling to each feature individually based on three modes. First, for a pair of datasets (source and transfer), we train the scalers only on the source dataset, and apply them to both source and transfer datasets (*source (S)*). Second, we train and apply two scalers independently of each other, one on the source dataset and one on the transfer dataset (*source & transfer, (S&T)*). Finally, we add 10% of the transfer dataset to the source dataset only for training a single scaler. Afterwards, the trained scaler is

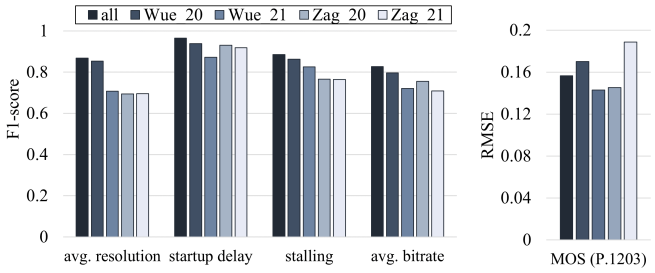
applied to both original source and transfer datasets (*source merged (SM)*).

Decomposition. We use the decomposition-based methods Principal Component Analysis (PCA) and Canonical Correlation Analysis (CCA), which allow to reduce the dimensionality of datasets by concentrating most of the dataset's variance in fewer dimensions using a linear transformation. For PCA, we linearly transform the datasets keeping all components, i.e., without reducing dimensionality. Note that the subsequent feature selection can act as dimensionality reduction in this case. CCA is a supervised approach requiring labels. It reduces the datasets to a single component, i.e., a single dimension. For both PCA and CCA, we again apply all three modes *source*, *source & transfer*, *source merged*.

Manifold learning. We apply manifold learning, which is a non-linear dimensionality reduction technique, in particular Local Linear Embedding (LLE) and Isomap. While LLE computes a lower-dimensional projection of the dataset that preserves distances within local neighborhoods, Isomap computes a lower-dimensional embedding which considers distances between all data points. For both LLE and Isomap, we again apply all three modes *source*, *source & transfer*, *source merged*. Additionally, we implemented semi-supervised manifold alignment (SSMA) [46], [47] in mode *source & transfer*, which considers distances of pairs of corresponding instances between source and transfer datasets. We construct these pairs from the video sessions in both datasets, which streamed the same video under the same bandwidth limitation, and compute their distance according to their application-layer KPIs, namely initial delay, number of stalling event, total stalling time, number of quality changes, average quality, and average bitrate. While the embedding is learned from the corresponding pairs, it can then be computed also for all other data points.

ML-based feature representation transfer. We implemented an ML-based feature representation transfer (MLFRT) in mode *source & transfer*, which again considers the same pairs of corresponding instances between source and transfer datasets as above. It trains a multi-output Random Forest regressor to transform the features from the transfer dataset into the corresponding features of the source dataset. It performs a 3-fold cross-validation to find the best hyperparameters from the considered Random Forest hyperparameter set described above. Afterwards, it transforms the transfer dataset into its source domain representation.

Drift elimination. For a pair of datasets (source and transfer), we find the features that differ among the two datasets and eliminate them prior to model training on the source dataset. The assumption here is that the model would perform better on the transfer dataset if it relies only on features that are similar in the two datasets. To identify drifting features, we evaluate how well can models based on one feature classify the origin of the sample (source or transfer dataset). Concretely, we take network traffic features of both datasets, and label the samples with their origin (dataset). Then, for each feature, we train a Random Forest model that classifies the origin based on that feature only and evaluate it through 2-fold cross-validation. If the model performs well, the feature is considered as drifting.



(a) Classification. (b) Regression.

Fig. 6: Performance of baseline and general models.

In the analysis described later, we eliminate the features based on the area under the receiver operating characteristic curve (ROC_AUC), which is a measure for the goodness of the classification performance. This means, we eliminate features that exceed the drifting threshold set to $ROC\_AUC > 0.7$ and $ROC\_AUC > 0.8$.

Enrichment. The source dataset is enriched with a labeled subsample from the transfer dataset. We evaluated three intensities of enrichment that we later refer to as 10%- 20%- and 30%-enrichment, depending on how much data from the transfer domain is added to the source domain training set. Note that 10% enrichment means that to the balanced source dataset that has the size of N samples, we add $0.1N$ samples from the transfer dataset. The rest of the transfer dataset samples are used for testing.

VI. Evaluation

We established the baseline by training and testing models in a network-specific fashion. The performance of such models is depicted in Figure 6. The figure also shows the performance of general models trained and tested on the merged dataset consisting of samples from all four datasets (denoted by "all" in the Figure). For all KPIs, general models outperformed network-specific models. This indicates that introducing data from different sources into model training results in better coverage of feature space, thus increasing the performance over all datasets. The estimation of P.1203 is an exception in that regard, which could be explained by MOS generally being a more complex concept, but also poor results obtained on individual datasets (e.g., `Zag_2021`). The figure shows that models trained and tested on `Wue_2020` perform better in comparison to other network-specific models. This may be attributed to somewhat clearer separation between the classes for that particular dataset (cf. Figure 2).

Figure 7 evaluates the classification performance of the investigated transfer methods on the transfer dataset. For this, we compare the performance of models, which are trained on the source dataset and transferred to the transfer dataset, to the performance of the best model trained on the transfer dataset, which acts as baseline. The different methods to improve transfer are depicted on the x-axis. The plot shows boxplots for each method, which describe the distribution of difference in F1 score compared to baseline over all 12 source/transfer dataset combinations and all 4 classification targets, i.e., in

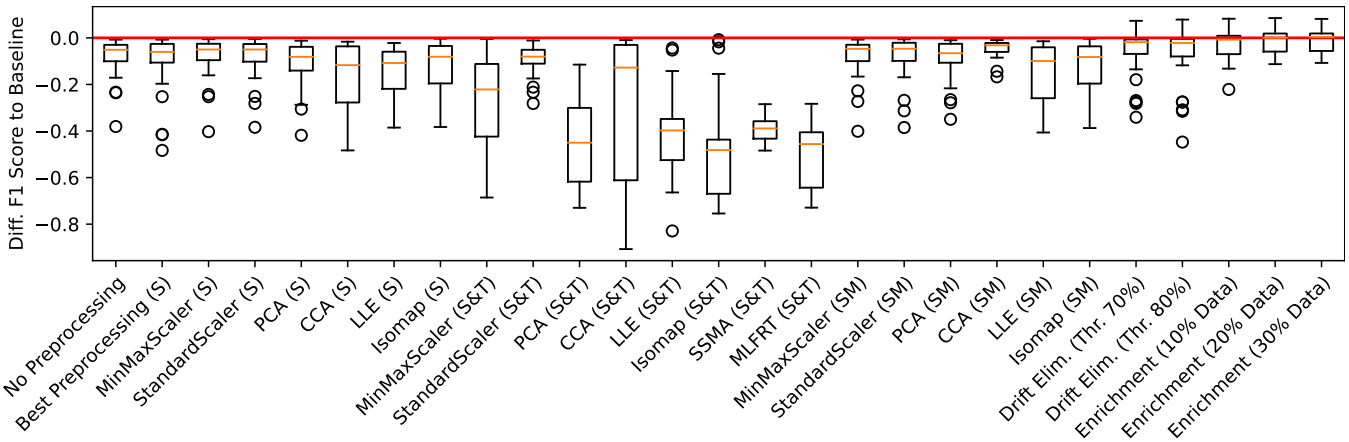


Fig. 7: Transfer dataset classification performance of models trained on source dataset using different methods for improving transfer compared to best model trained on transfer dataset (baseline). Positive values indicate improvement wrt. to baseline.

total 48 combinations. The median (50-percentile) is depicted as an orange line, the box extends from the lower to the upper quartile (25- to 75-percentile), whiskers reach from 5- to 95-percentile, and extreme values beyond the whiskers are plotted as individual points. Positive values indicate improvements with respect to the baseline while negative values show a decrease in F1 score. For example, considering the leftmost box for "No Preprocessing", the lowest point at -0.38 indicates that there was a combination of source dataset, transfer dataset, and target for which the transferred model had an F1 score, which was 0.38 lower than the baseline result. This shows that the performance of trained models can substantially drop when models are applied to other datasets for which they have not been trained and no method for improving the transfer is applied. Nevertheless, the median difference when applying no preprocessing is at -0.05 and the upper whisker extends to the maximum value of -0.01, which suggests that performance of the transferred models without any preprocessing can be also close and only slightly below the baseline.

We see a similar range of performance differences for most of the investigated *source mode (S)* methods. In particular, we find that more complex methods cannot outperform simple scaling approaches. When investigating those distributions more closely, we could not find any regularity concerning source dataset, transfer dataset, or target. Thus, we infer that the actual combination itself and its peculiarities in the concrete training and test sets define whether a certain methods performs close to the baseline or not. When investigating the methods in *S&T mode*, we see that, except for standard scaling, the boxes and whiskers extend to much lower values, which means that those methods can result in much higher performance degradation compared to the baseline. In addition, LLE and Isomap have a strongly negative 95-percentile (upper whisker), while PCA, SSMA, and MLFRT even have a strongly negative maximum value in *S&T mode*, such that they will almost always (LLE, Isomap) or always (PCA, SSMA, MLFRT) result in a substantial performance loss, and thus, cannot be not recommended. Considering the methods in *SM mode*, where some transfer data is merged to the source data before training the methods, we see similar performance than for the methods in *S mode*. Thus, also here, transfer performance can be close to the baseline, but can also lead to a substantial F1 score reduction. The most consistent performance is reached by CCA *(SM)*, which reaches values close to the baseline in almost all cases, but requires labels from the transfer domain. In contrast, all unsupervised approaches cannot outperform simple scaling methods, and thus, cannot ensure high performance when models are transferred.

Finally, we investigate the performance when eliminating drifting features or enriching the source dataset with a labeled subsample of the transfer dataset, which is depicted in the five rightmost boxes in Figure 7. Here, we can see that all boxes intersect the red zero line, having values closer to the baseline and in some cases even extending to positive values. Here, again the fluctuations around the zero line might be depending on the peculiarities of the actual data. Nevertheless, generally, boxes are smaller and we see a much more consistent performance with respect to the baseline, which means that they almost always avoid severe performance degradation when using the transferred models. Thus, we conclude that both methods are well suited for improving the transfer performance. While enrichment gives the best results, which was expected, it has the drawback that a labeled subsample of the transfer dataset is required, which might typically not be available. However, drift elimination provides an almost equally good performance without requiring labels from the transfer dataset. Therefore, it is much more applicable in typical transfer scenarios, and thus, proves to be the most valuable method for improving transfer here.

Similarly to Figure 7, Figure 8 depicts the transfer dataset performance for P.1203 MOS estimation, which is a regression task. The figure shows a boxplot for the distribution of difference in root mean square error (RMSE) between transfer methods and baseline result over all 12 source/transfer dataset combinations. Note that RMSE results are on the same scale as MOS values, and that, as the differences (errors) between estimated and actual P.1203 MOS values need to become smaller, thus, in this figure, negative values indicate an improvement with respect to the baseline. The best results on the transfer dataset are again reached by enrichment. We

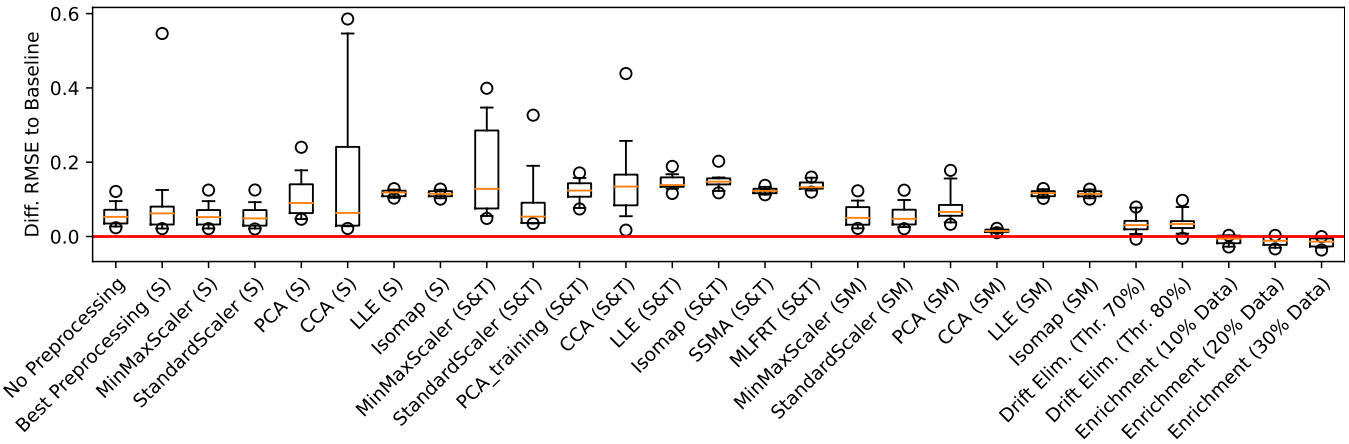


Fig. 8: Transfer dataset regression performance of models trained on source dataset using different methods for improving transfer compared to best model trained on transfer dataset (baseline). Negative values indicate improvement wrt. to baseline.

can see that these boxes are located around the baseline and in some cases extending to negative values, i.e., to smaller RMSEs than the baseline models.

For other methods, it can be seen that the results generally resemble the previously shown results for classification performance. In particular, we see that most methods result in a modest increase of RMSE with respect to the baseline between 0.05 and 0.15 when the models were trained on the source dataset with only few outliers. We also can observe that *S* and *SM mode* models perform similarly, and that *S&T mode* models perform typically worse. Again, the most consistent performance is given by CCA *(SM)*, which can reach RMSE values closely above the baseline in all cases, but requires labels from the transfer domain. Looking at the results for drift elimination, we can see that it gives slightly worse RMSE compared to the baseline in most cases, and can only improve the RMSE in a single source/transfer dataset combination. This is different from the classification results above, and shows that this method not always can reach a transfer performance on a par to baseline models. Still, it provides the best performance when considering only unsupervised methods, which do not require labels from the transfer domain.

In the following, we investigate drifting features in more detail. Figure 9 shows the most severely drifting features across all datasets. The severity of drift is measured as ROC_AUC in the 2-fold cross-validation of the model classifying the dataset origin. Since feature drift is tested for distinct dataset pairs, this gives 6 evaluations of ROC_AUC (one for `Wue_2020` and `Wue_2021`, one for `Wue_2020` and `Zag_2020`, etc.). The Figure shows the number of evaluations (dataset pairs) in which the feature was identified as drifting and the average value of ROC_AUC. Note that the average only takes into account evaluations in which ROC_AUC was higher than 0.7. For example, the maximum size of uplink packets greater than 100B was drifting in all 6 evaluation, with average drift across 6 evaluations equal to 0.9577. The maximum download data volume in slots of 100ms was drifting in 2 evaluations, with average drift across those 2 evaluations being 0.8068.

The most severely drifting features are related to maximum length of uplink and downlink packets. As these features

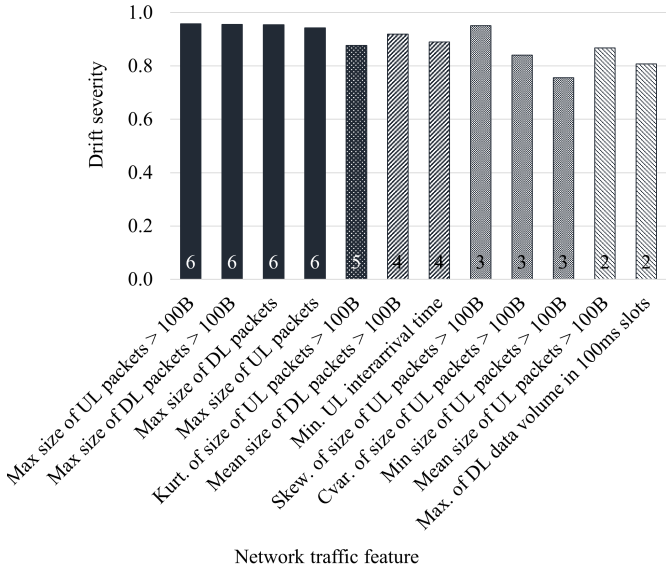


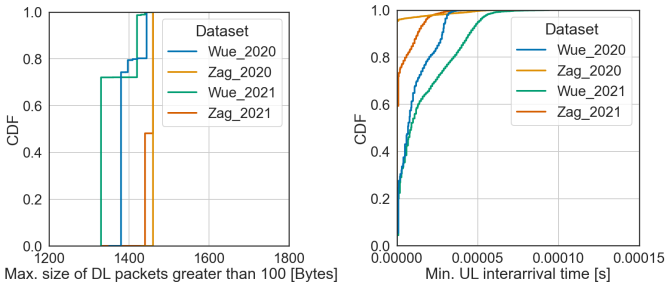
Fig. 9: Features identified as most drifting.

correspond to a largest packet for each video, the values are expected to be defined by link properties and should not vary a lot across video samples originating from the same dataset. We confirm that with Figure 10a where it is visible that the max downlink packet length is typically consistent within a dataset, but the feature clearly separates distinct datasets.

In Figure 10b, we inspect the drift of the minimum inter-arrival time of uplink packets across datasets. The minimum value of inter-arrival time is typically reached when there are two consecutive packets on the uplink, which are very small, and the link is not congested. In this case, this feature can be considered a proxy for the available link capacity. The figure shows that, especially for larger values, the feature clearly separates the datasets.

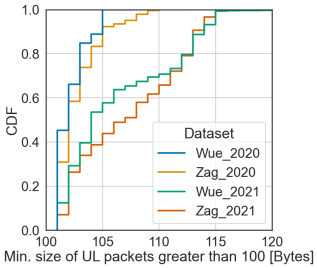
Finally, we look into the distributions of the minimum size of non-trivial uplink packets in Figure 10c, i.e., packets greater than 100KB, which excludes pure acknowledgement packets. It can be seen that datasets collected in the same year are more

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2023.3326664



(a) Maximum size of downlink packets greater than 100 B.

(b) Minimum of uplink inter-arrival time.



(c) Minimum size of uplink packets greater than 100 B.

Fig. 10: Distributions of selected drifting features.

similar when it comes to this feature. Thus, while our previous examples showed drifts in the network characteristics, this hints at a drift happening on the application side. In particular, we assume that the size of the non-trivial uplink packets has increased by a few bytes most likely due to having introduced a new field into the uplink data before our 2021 measurements.

To sum up, our evaluations showed that a consistent performance on the transfer dataset close to the baseline, i.e., a model specifically trained on the transfer dataset, can only be reached by using labelled data from the transfer dataset. The best method is enrichment of the source dataset with labelled data from the transfer domain, which can even improve on the baseline in some cases. The reason is that this method not only allows to use a larger dataset for training the model but also allows to consider examples collected in the transfer domain, which help to learn meaningful and more general concepts that apply to both source and transfer domains.

However, in a typical transfer scenario, such labels from the transfer domain will not be available, which prevents to use enrichment. In this case, while most methods could not outperform simple scaling, we found drift elimination to be the best method for reaching performance close to the transfer baseline. By eliminating drifting features, which does not require labels from the transfer domain, the trained models need to focus on features that are similar in both domains. This removes domain-specific peculiarities, such as maximum downlink packet size from the data, and thus, forces models to ignore them and focus on learning more general concepts instead. Consequently, models trained only on similar features should also give similar performance on both datasets. Our results showed that, for MOS regression, the performance of this method was only slightly worse than the baseline performance, while for the classification tasks, its performance was mostly on a par with the baseline or the enrichment method. This shows that drift elimination can be recommended for typical transfer scenarios.

VII. Conclusion and Outlook

Machine learning has been heavily used for estimating KPIs and QoE of video streaming flows in the network. While providing promising results in terms of estimation model performance, such approaches typically require extensive data collection - both of network traffic features and application-level performance information. Considering the wide variety of scenarios in which video streaming could be used (e.g., different devices, networks, operating systems), it is clear that including all of the possible scenarios in the data collection is demanding, if not impossible. In this paper, we tackled the problem of data collection extensiveness by evaluating whether an existing model trained on data from one network could be adapted and used on data obtained from a different network. The adaptation methods used in the paper require only network traffic features without labels (which can be calculated from the traffic generated by real users, without test devices), or require a significantly smaller labelled dataset from the network for which the model is adapted.

The paper evaluated adaptation methods based on scaling, decomposition, manifold learning, ML–based feature representation transfer, drift elimination, and enrichment. While the evaluation pointed out that enrichment of the source dataset with labelled data from the transfer domain and the elimination of drifting features are the most promising methods for improving the transfer of models in this context, subsequent studies are needed to address related research questions. An interesting way forward may be to explore combinations of the described methods. For example, eliminated drifting features could be transformed and introduced back into the dataset.

Apart from the session-level QoE/KPI estimation models considered in this work, we will investigate whether the same methods are as promising in the case of real-time KPI estimation using the collected real-time ML dataset, or when considering transfer between different bandwidth limitation conditions. Moreover, the methodology could be repeated for new datasets obtained in scenarios differing in aspects other than network (e.g., mobile operating system, streaming service).

When it comes to collecting new datasets, a big challenge is obtaining a good coverage of the feature space. For the model to learn to detect QoE degradations, the training set needs to include such degradations. Existing literature, however, does not provide guidelines on how to collect data to ensure a good coverage of scenarios that may occur in an operational network. Another challenge is related to assessing the needed size of the dataset to train robust QoE/KPI estimation models.

Additional questions arise when considering practical application of the models. The research focus in the area has mainly been on developing methodologies, while deploying solutions based on proposed methodologies would require experimentation, adaptation, and customization. The amount of resources

(compute, storage, network) needed for 1) real-time processing of the traffic and calculating the traffic features, and 2) executing the model on calculated features will depend on the available infrastructure. There is a number of solution design choices that may be considered in that regard. For example, an ISP may be willing to sacrifice some model performance (e.g., using simpler models) if that would significantly reduce the computation cost. Moreover, they might not be interested in assessing the QoE of each and every session, but rather in sampling the traffic in a meaningful way.

With respect to the methods that may be considered in future work, we also see deep learning based methods as worth exploring when large datasets are available. In particular, this includes approaches using pre-trained and frozen layers obtained from the source dataset and newly added trainable layers, which can adapt the model decisions to the transfer dataset. Another promising direction are models for learning more meaningful representations, such as transformers, which can potentially learn embeddings that are more independent of certain network peculiarities, and thus, allow for a better transfer performance. We will investigate these approaches in future works.

Acknowledgment

This work was partly funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) under grant SE 3163/3-1, project number: 500105691 (UserNet) and partly funded by the Croatian Science Foundation under the project IP-2019-04-9793 (Q-MERSIVE). The authors alone are responsible for the content.

References

[1] Ericsson, "Ericsson Mobility Report," Ericsson, Tech. Rep., 2021. [Online]. Available: https://www.ericsson.com/4ad7e9/assets/local/reports-papers/mobility-report/documents/2021/ericsson-mobility-report-november-2021.pdf

[2] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez *et al.*, "A year in lockdown: how the waves of covid-19 impact internet traffic," *Communications of the ACM*, vol. 64, no. 7, pp. 101–108, 2021.

[3] Sandvine, "The Global Internet Phenomena Report," Sandvine, Tech. Rep., 2021. [Online]. Available: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2022/Phenomena%20Reports/GIPR%202022/Sandvine%20GIPR%20January%202022.pdf

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

[5] 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.

[6] I. Sodagar, "MPEG-DASH: The Standard for Multimedia Streaming Over Internet," International Organisation for Standardisation, WA, USA 98052, White paper ISO/IEC JTC1/SC29/WG11 W13533, apr 2012.

[7] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc)," *Proceedings of the IEEE*, vol. 109, no. 9, pp. 1463–1493, 2021.

[8] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori, "Qoe management of multimedia streaming services in future networks: a tutorial and survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 526–565, 2019.

[9] 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

[10] ——, "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

[11] 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 *15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014, pp. 1–6.

[12] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "BUFFEST: Predicting Buffer Conditions and Real-time Requirements of HTTP(S) Adaptive Streaming Clients," in *8th ACM on Multimedia Systems Conference*, 2017, pp. 76–87.

[13] 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.

[14] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying Flows and Buffer State for YouTube's HTTP Adaptive Streaming Service in Mobile Networks," in *ACM Multimedia Systems Conf. (MMSys 2018)*, June 2018.

[15] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring Video QoE from Encrypted traffic," in *Internet Measurement Conference*, 2016, pp. 513–526.

[16] 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.

[17] 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.

[18] 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," *Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–25, 2019.

[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 *10th ACM Multimedia Systems Conference*, 2019, pp. 48–59.

[20] 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.

[21] I. Bartolec, I. Orsolic, and L. Skorin-Kapov, "Impact of user playback interactions on in-network estimation of video streaming performance," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3547–3561, 2022.

[22] I. Orsolic and M. Seufert, "On Machine Learning Based Video QoE Estimation Across Different Networks," in *16th International Conference on Telecommunications (ConTEL)*. IEEE, 2021, pp. 62–69.

[23] 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 *2016 IEEE Globecom Workshops*. IEEE, 2016, pp. 1–6.

[24] ——, "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.

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

[26] 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 *9th International Conference on Quality of Multimedia Experience (QoMEX)*, Erfurt, Germany, 2017.

[27] 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 *Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 688–695.

[28] 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 *Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2019, pp. 199–200.

[29] ——, "I See What you See: Real Time Prediction of Video Quality from Encrypted Streaming Traffic," in *4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks*, 2019, pp. 1–6.

[30] 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 *4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks*, 2019, pp. 7–12.

[31] S. Schwarzmann, C. C. Marquezan, R. Trivisonno, S. Nakajima, and T. Zinner, "Accuracy vs. Cost Trade-off for Machine Learning Based QoE Estimation in 5G Networks," in *International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[32] European Telecommunications Standards Institute, "TS 123 501 System architecture for the 5G System, version 17.8.0," Standard, 2023.

[33] ——, "TS 123 288 Architecture enhancements for 5G System (5GS) to support network data analytics services, 17.8.0," Standard, 2023.

[34] M. Seufert, S. Wassermann, and P. Casas, "Considering user behavior in the quality of experience cycle: towards proactive qoe-aware traffic management," *IEEE Communications Letters*, vol. 23, no. 7, pp. 1145–1148, 2019.

[35] 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 *14th International Conference on Network and Service Management (CNSM)*. IEEE, 2018, pp. 233–239.

[36] 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.

[37] S. Ickin, M. Fiedler, and K. Vandikas, "QoE Modeling on Split Features with Distributed Deep Learning," *Network*, vol. 1, no. 2, pp. 165–190, 2021.

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

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

[40] G. Widmer and M. Kubat, "Learning in the Presence of Concept Drift and Hidden Contexts," *Machine learning*, vol. 23, no. 1, pp. 69–101, 1996.

[41] A. Tsymbal, "The Problem of Concept Drift: Definitions and Related Work," *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004.

[42] A. Bifet, J. Gama, M. Pechenizkiy, and I. Zliobaite, "Handling Concept Drift: Importance, Challenges and Solutions," The University of Waikato, Tech. Rep., 2011. [Online]. Available: https://www.cs.waikato.ac.nz/~abifet/PAKDD2011/PAKDD11Tutorial_Handling_Concept_Drift.pdf

[43] K. O. Stanley, "Learning Concept Drift with a Committee of Decision Trees," *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003.

[44] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An Overview of Concept Drift Applications," in *Big data analysis: new algorithms for a new society*. Springer, 2016, pp. 91–114.

[45] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early Drift Detection Method," in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, 2006, pp. 77–86.

[46] J. Ham, D. Lee, and L. Saul, "Semisupervised alignment of manifolds," in *International Workshop on Artificial Intelligence and Statistics*. PMLR, 2005, pp. 120–127.

[47] J. Wang, X. Zhang, X. Li, and J. Du, "Semi-supervised manifold alignment with few correspondences," *Neurocomputing*, vol. 230, pp. 322–331, 2017.

[48] 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.

[49] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "YoMoApp: A Tool for Analyzing QoE of YouTube HTTP Adaptive Streaming in Mobile Networks," in *European Conference on Networks and Communications (EuCNC)*. IEEE, 2015, pp. 239–243.

[50] 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

[51] J. H. Friedman and L. C. Rafsky, "Multivariate generalizations of the wald-wolfowitz and smirnov two-sample tests," *The Annals of Statistics*, pp. 697–717, 1979.

Michael Seufert is a Full Professor at the University of Augsburg, Germany, heading the Chair of Networked Embedded Systems and Communication Systems. He received the Bachelor's degree in econamathematics and the Diploma, PhD, and Habilitation degrees in computer science from the University of Würzburg, Germany, and holds the First State Examination degree in mathematics, computer science, and education for teaching in secondary schools. His research focuses on user-centric communication networks, including QoE of Internet applications, AI/ML for QoE-aware network management, as well as group-based communications.

Irena Orsolic received the M.Sc. degree in information and communication technology and the Ph.D. degree in computer science from the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia in 2016 and 2020, respectively. She was a Postdoctoral Researcher and a member of the Multimedia Quality of Experience Research Lab at the same university. The focus of her research is on Quality of Experience estimation of encrypted video streaming by using machine learning methods. Since 2023 she has been working as an Experienced Core Network Researcher at Ericsson AB, Stockholm, Sweden.