Features that matter: feature selection for on-line stalling prediction in encrypted video streaming

Michael Seufert, Pedro Casas, Nikolas Wehner, Li Gang, Kuang Li

Angaben zur Veröffentlichung / Publication details:

Seufert, Michael, Pedro Casas, Nikolas Wehner, Li Gang, and Kuang Li. 2019. "Features that matter: feature selection for on-line stalling prediction in encrypted video streaming." In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 29 April - 2 May 2019, Paris, France*, 688–95. Piscataway, NJ: IEEE. https://doi.org/10.1109/infcomw.2019.8845109.

Features that Matter: Feature Selection for On-line Stalling Prediction in Encrypted Video Streaming

Michael Seufert*†, Pedro Casas*, Nikolas Wehner*, Li Gang‡, Kuang Li‡
*AIT Austrian Institute of Technology GmbH, Vienna, Austria
†Now at: University of Würzburg, Würzburg, Germany
‡Huawei Technologies, Department of R&D, Shenzhen, P.R. China
michael.seufert@uni-wuerzburg.de, pedro.casas@ait.ac.at,
nikolas.wehner@ait.ac.at, lig619@huawei.com, kuangli@huawei.com

***Abstract*—Despite the vast literature and major developments in HTTP adaptive video streaming (HAS) technology, stalling events due to re-buffering still are by far the worst Quality of Experience (QoE) degradation, and thus, represent a major issue for ISPs. In this paper, we address the problem of real-time QoE monitoring of HAS, focusing on the detection of re-buffering events and QoE-relevant metrics, for the particular case of YouTube. Given the wide adoption of end-to-end encryption, we resort to machine learning models to predict these metrics directly from the analysis of the encrypted traffic. The salient feature of our approach ViCrypt is its ability to perform QoE predictions in real-time, during the course of an ongoing YouTube streaming session, relying on constant memory, stream-like inputs continuously extracted from the encrypted stream of packets. We show through empirical evaluations that ViCrypt can predict the occurrence of re-buffering events with a time granularity as small as one second with very high accuracy. By aggregating independent predictions, ViCrypt is able to accurately estimate per-video-session QoE-relevant metrics such as initial playback delay, number of re-buffering events and stalling ratio. In this situation, we investigate if a decent prediction performance can also be reached by selecting reduced feature sets based on the relevance of the features. Moreover, we explore the potential of including recurrent features, namely, stalling predictions from past time slots, to improve the prediction performance.**

I. Introduction

Video streaming has become the most popular and most demanding application of the Internet due to the high number of requests, high bit rates of the video content and strict real-time requirements of the video playback. Still, the delivered streaming service has to meet the expectations of the end users. To understand and eventually improve Internet services like video streaming, content, application, and Internet service providers are now more often relying on the concept of Quality of Experience (QoE) to quantify the subjective experience and satisfaction of their customers. The intensifying competition among operators is forcing ISPs to integrate QoE into the core of their network management systems, from network monitoring and reporting to traffic engineering.

When it comes to video streaming performance, it is widely accepted that the most severe QoE degradations are the interruptions of the playback or *stalling*, caused by re-buffering events [1]–[3], and the waiting time until the start of the playback. While these degradations have been partially mitigated by adapting the video bit rate to the network conditions, e.g., HTTP Adaptive Streaming (HAS), re-buffering is still the most annoying and prevalent QoE degradation [1], [4]. Previous studies have also shown that re-buffering is not only detrimental for the overall user experience, but is also highly correlated to viewer engagement. As a consequence, ISPs are highly interested in solutions able to detect the occurrence of such events as soon as they happen to take appropriate countermeasures. In the QoE-aware traffic management cycle [4], network operators continuously monitor the QoE of their customers and apply network traffic management, such as bandwidth shaping or re-routing, to avoid the stalling of video streams or to relief users from ongoing re-bufferings. The trend towards end-to-end encryption (e.g., HTTPS) has significantly reduced the visibility of network operators on their customers' traffic, making the monitoring process more challenging and cumbersome. It is no longer possible to rely on Deep Packet Inspection (DPI) based approaches to analyze the video data contained in each packet to reconstruct the streaming process and the video buffer [5]. The encrypted stream of packets only offers very basic information about the streaming process, such as packet sizes and inter-arrival times.

In this paper, we present ViCrypt, a machine-learning based approach for monitoring QoE-relevant metrics in YouTube, capable of predicting the occurrence of re-buffering events in real-time from such basic features. To do so, ViCrypt analyzes ongoing streaming sessions using fine grained time slots of 1 s length. Multiple snapshot-like statistical features are computed from the video traffic in a stream-based fashion with constant memory consumption for these time slots independently. Additionally, two macro windows consisting of multiple time slots are considered to capture trend and progression properties of the streaming session. More precisely, ViCrypt considers a first sliding window aggregating the last t time slots to compute trend features, and a second sliding window aggregating all past time slots since the start of the streaming session to compute session-progression features. At the end of each time slot, its features and the features of the corresponding macro windows are fed into a random forest model, which predicts whether the current time slot of 1 s contains stalling or not. This is by now the finest granularity of real-time prediction. As initial results promised a very good prediction results using all extracted features [6], we will investigate in this paper if that high level can also be reached by considering only reduced feature sets. At the same time, we identify, which features are especially relevant for the accurate prediction of stalling in independent time slots. As the independent stalling predictions for each consecutive time slot of a streaming session can be aggregated to obtain session-based QoE-relevant metrics, including initial delay, the total number of stallings, and the stalling ratio, i.e., the ratio between total stalling time and total playback time, we will further investigate the potential of including recurrent features, i.e., whether adding predictions of past time slots as features can improve the prediction results.

The remainder of the paper is organized as follows. Sec. II describes related work on QoE of HAS and QoE-based network monitoring approaches. Sec. III describes ViCrypt, introducing the concepts of time slots and sliding windows used in this work and the computed features. Sec. IV explains how the relevance of different features is explored. The prediction performance of ViCrypt with different feature sets is evaluated in Sec. V; finally, Sec. VI concludes this paper.

II. Related Work

The most important results on Quality of Experience (QoE) of HTTP adaptive streaming (HAS) were summarized in [1]. Also more recent publications confirmed the findings that stalling, initial delay, and quality adaptation are the most dominant QoE factors. Stalling, i.e., the playback interruptions due to buffer depletion, is considered the worst QoE degradation [2], [3]. This is why the real-time prediction of stalling is the most important goal of this work. Moreover, the played out video quality and the time on each quality layer also impact the QoE [7], but they are out of focus of this work.

Several works focused on estimating stalling of video streaming, which can be mapped to QoE, e.g., with the model presented in [8]. [5], [9], [10] transferred the approach of [11] and proposed an in-network system based on DPI to extract downloaded playtimes. They could be used to estimate the buffered playtime at the client, and thus, the corresponding stalling events. Similar approaches were followed by [12], which supported more video encodings and container formats, and by [13], which predicted stalling in LTE networks. [14] estimated stalling events based on the ratio of playback time and download time, but needed the total size of the video for real time estimation of stalling. Recently, a quality assessment model for HAS was standardized (P.1203, [15]), which predicts the MOS from stream inspection.

The trend towards end-to-end encryption, which prevents DPI-based approaches, has motivated a recent shift in QoE-based network monitoring to using low-level network measurements rather than relying on application-layer metrics. While some approaches explicitly tackle the QoE of mobile apps, including [16]–[18], there are also general approaches for QoE analyses based on network-layer monitoring of encrypted video streaming traffic. Authors in [19] evaluate machine learning-based architectures that estimate YouTube QoE from features derived from packet sizes, inter-arrival times, and throughput measurements. A similar approach is presented in [20], where authors rely on real cellular network measurements to predict QoE factors for video streaming (e.g., played resolutions, stalling events), based on features such as round-trip times, packet loss, and chunk sizes. Here, authors also used machine learning for large-scale quality monitoring and prediction. [21] focuses on the reconstruction of buffered playtime at the video player side, as previously done in [9], but for encrypted traffic. This is leveraged to estimate video QoE metrics in [22]. [23], which is the most similar work, used machine learning to predict initial delay, stalling, and video quality from the network traffic in windows of 10 s based on features derived from IP or TCP/UDP headers only.

Different from all these papers, our approach ViCrypt [6] detects QoE degradations on encrypted video streaming traffic in real-time by using a stream-like analysis approach. It considers three windows (current, trend, session) with only a minimal memory footprint, i.e., the windows store only a small set of features, which can be computed with constant memory consumption. The features are based on packet-level statistics of the network traffic, and allow to accurately recognize stalling of the streamed video within time slots of 1 s. This is by now the finest granularity of real-time prediction. Finally, with ViCrypt, the individual predictions of each time slot can be aggregated to accurate stalling statistics on a session level.

III. The ViCrypt Approach

The salient feature of ViCrypt is its ability to perform predictions in real-time, during the course of an ongoing YouTube streaming session. For doing so, a video streaming session is subdivided into a sequence of consecutive, short-duration time slots of fixed length. After a time slot has ended, a binary stalling prediction is performed, indicating whether stalling is present or not in this time slot. In this paper, ViCrypt uses 1 s time slots, which provides a proper trade-off between stalling detection delay and accuracy.

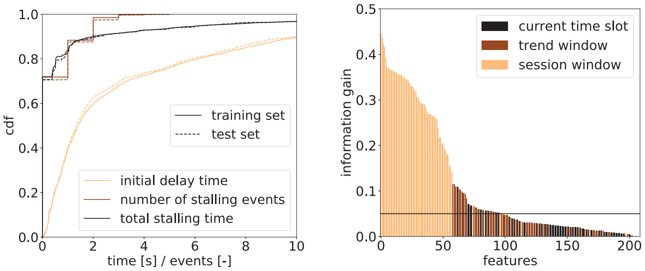
The stalling prediction for the current time slot can only rely on features extracted from the traffic of the current or past time slots. As there is a possibly large amount of previous time slots, which would lead to a high memory consumption, the past streaming information has to be compressed and structured. For this, in addition to the current time slot, the proposed system keeps track of only two additional windows, namely, the trend window and the session window. The trend window comprises t time slots, and thereby, contains all traffic of the current slot and the $t-1$ most recent slots. In this work, we use a trend window size of $t = 3$; thus, the trend window contains the traffic of the current slot and the two previous slots. The session window is a macro window, which covers all traffic of the session so far, i.e., it includes the current and all previous time slots. The features of each individual time slot, trend window, and session window are computed in an on-line fashion without the need to store information for each packet, which significantly reduces the memory consumption from linear to constant.

First, simple count-based features are computed from the traffic observed in the time slot. These consist of the number of total, uplink, and downlink packets, and the amount of transferred bytes (total, uplink, downlink). The number and byte volume of TCP and UDP packets are counted separately, and the upload ratio, download ratio, TCP ratio, and UDP ratio are computed from these counters for both number of packets and amount of bytes. Next, time-based features are computed. These include time from the start of a time slot until the first packet, the time after the last packet until the time slot end, and the burst duration, i.e., the time between the first packet and the last packet of the slot. All features are again computed for the total traffic, as well as for uplink and downlink traffic. The average throughput of the slot (i.e., traffic volume divided by slot length) and the burst throughput (i.e., traffic volume divided by burst duration) can be subsequently derived for total, uplink, and downlink traffic. A covariance-based algorithm[1] is used to compute a linear regression for the cumulative traffic volume over time in an on-line fashion. Two regressions are performed for uplink and downlink traffic, and the slope and intercept of the corresponding regression lines are also added as features. Finally, several characteristics of the traffic can be described by a distribution. An algorithm is utilized, based on [24], which can compute the first four moments of any distribution in an on-line fashion, i.e, the mean, the variance, the skewness, and the kurtosis. This algorithm was trivially augmented to additionally output the standard deviation, the coefficient of variance, as well as the minimal and the maximal values. These distribution-based features are computed for the packet size and the inter-arrival time between packets for both uplink and downlink traffic.

This results in a total of 69 basic features for the traffic of a time slot. As mentioned above, two windows are additionally considered for the feature extraction, namely, the trend and the session windows, for which the same 69 basic features are computed. Together with the ordinal sequence number of the current time slot, which basically tracks the current stream time, this sums up to 208 features, which are associated to each 1 s length time slot. Note again that, as described, all input features used by ViCrypt are computed with constant memory, using a stream-like one-pass approach.

IV. Exploration of Feature Relevance

The results for ViCrypt (see below) show that a high prediction performance can be achieved when considering the features of all three window types. Although all of the 208 features are computed with constant memory, this could still result in a high computational effort for extracting features for a huge number of streaming sessions in parallel. Thus, in this work, we investigate if that high level can also be reached by considering only reduced feature sets. At the same time, we identify, which features are especially important for the accurate prediction of stalling in independent time slots. Moreover, we investigate the idea of a recurrent random forest prediction. For this, previous prediction results are fed back as a feature for the prediction of stalling in the current slot. In the following, we will first present the data set. Then, we will describe the creation of the investigated feature sets and the training of the model for the stalling prediction.



(a) Distribution of QoE metrics. (b) Features ranked by IG.

Fig. 1: Characterization of the data set used for training and testing ViCrypt.

A. Data Set

For training and testing purposes, we generated a data set consisting of 4,714 YouTube video sessions, streamed and recorded over a period of several weeks in summer 2018 using a Chrome browser automation tool. Additionally, we added 135 YouTube app video sessions from the recently published open dataset [25], [26] to our data set. In each network trace, both TCP- and QUIC-based YouTube video flows were identified based on the domain name, and features were only extracted for the combined flows of each session, discarding all non-YouTube traffic. The ground truth, i.e., whether the video was stalling or not, was obtained from a JavaScript-based monitoring script [4], [27].

80% of the video sessions (randomly selected) were considered for training, and the remaining 20% were added to the test set. This means 3,879 sessions were used for training, and the test set consisted of 970 sessions. Fig. 1a shows the stalling characteristics of the training set (solid lines) and the test set (dashed lines) as cumulative distribution functions (CDFs). The initial delay time, excluding page load time and stalling during playback, is mostly small; 68% of the sessions have an initial delay below 3 s. This is partially due to short advertisement clips before some videos on YouTube, which require few data to be downloaded and can start very fast. However, some sessions faced serious initial delays of several seconds, as the 95-percentile is 15.3 s and the 99-percentile is 28.3 s. For stalling, excluding initial delay, the figure shows that 72% of the sessions do not face stalling at all. If stalling is present, the highest number of stalling events is 5, while the total stalling time can be quite high, having a 95-percentile of 6.4 s and a 99-percentile of 19.5 s. While these numbers correspond to the training set, the shapes of the distributions for training and test sets are similar, confirming that training and test sets show the same characteristics.

The following steps were executed with the well-known open source machine learning software Weka[2]. The training

[1] https://stats.stackexchange.com/questions/23481/are-there-algorithms-for-computing-running-linear-or-logistic-regression-param

[2] https://www.cs.waikato.ac.nz/ml/weka/

data consisted in total of 635,209 time slots. As the training set contained much less slots with stalling (18.46%) than slots without stalling, bootstrapping was applied. For this, the training set was doubled and re-sampled in Weka by drawing uniformly random with replacement from each class to obtain balanced classes with 635,209 instances each. During the preparation of the training set, the order of the slots was randomized to avoid any serial-position effects during training.

B. Investigated Feature Sets

Five different feature sets are used throughout this work. The first set (CTS feature set) is straightforward and uses all 208 extracted features of all three windows - current time slot, trend window, and session window. The second feature set was created based on domain knowledge using the information gain (IG) of all features towards the class as a quantitative criterion (CTS/IG feature set). Weka's information gain ranking filter was applied to the training set to obtain a list of all features ordered by their respective information gain. After a manual inspection of the list, an IG threshold of 0.05 was selected to ensure that many subjectively relevant features were included. 97 features with an IG above the threshold were selected for the second feature set.

Fig. 1b shows all features ranked by IG towards the classification target, i.e., stalling in the current time slot. Mainly features of the session window have a high IG, the best being session volume (0.445), session download volume (0.437), number of session packets (0.422), number of session download packets (0.416), and number of session upload packets (0.393). The first features of the trend window appear on rank 59 (trend window download throughput, trend window download volume) with IG 0.115, just before the first feature of the current time slot (ordinal sequence number) with information gain 0.109 on rank 63. The next most important features of the current time slot are download volume (0.068), download throughput (0.068), download throughput during burst (0.059), and number of packets (0.058). After the threshold of 0.05, the IGs further drops for many features of the trend window and the current slot. The end of the ranking is occupied again by several features of the session window, which are almost constant with a very small IG, e.g., session's time until first packet, session's minimum upload packet size, and session's minimum inter-arrival time of uplink/dowlink packets. The whole IG feature set contains 11 features of the current time slot, 25 features of the trend window, and 61 features of the session window. A full list of all features and their IG can be found in the appendix (cf. Table V).

The third feature set was based on a purely algorithmic feature selection using Weka's CFS algorithm (CTS/CFS feature set). It selects a small set of relevant features, which have a high correlation to the stalling class but a low correlation between each other. Only 18 features are included in this set, which radically decreases the overhead for feature generation. They include the ordinal sequence number, three features of the trend window, and 14 features of the session window.

Finally, the idea of a recurrent random forest prediction is investigated. For this, previous prediction results are fed back as a feature for the prediction of stalling in the current slot. In the first case, only the binary prediction result for the previous slot is added as feature, i.e., whether the previous time slot was predicted to be in class "stalling" or "no stalling" (CTS+P feature set). Moreover, session level stalling metrics can be computed by aggregating individual stalling predictions in a sequence of consecutive slots of a session, which is described in the next subsection. These metrics include the initial delay, number of stalling events, duration of the current stalling event, total stalling time, stalling ratio, and two binary indicators whether the session is the initial delay phase or in a stalling phase. For each slot, these seven metrics are computed considering the whole past streaming session until the current slot, and are also added as features. Thus, the resulting CTS+PS feature set has a total of 216 features, eight of which are based on previous predictions.

C. Stalling Prediction

For each feature set, a random forest model using 25 trees is trained on the training set. The random forest model outperformed other models on the same classification task in terms of accuracy and training speed, and the number of trees was chosen to avoid overfitting of the model. The classification accuracy of the model was computed on the test set.

First, the classification accuracy is evaluated per individual time slot. This means, the results of the bare classification task for any time slot of the test set will be reported. This situation allows to obtain a real-time prediction for any video session, which indicates if there is stalling in the current slot or not.

Finally, the classification accuracy is evaluated per session. For this, individual and independent predictions of the sequence of time slots within a session are aggregated on the session level. This allows to obtain stalling information for the whole session, such as the initial delay, the number of stalling events, and the total stalling time (excluding initial delay), which can be converted into the stalling ratio (ratio of total stalling time and total playback time). The trained model predicts for each consecutive slot of the session if there is stalling or not. The initial delay in seconds is given by the number of slots (slot length is 1 s) at the start of the session, for which the model predicted stalling. After the initial delay, a stalling event is counted if two or more consecutive slots are predicted as "stalling" to make the aggregated stalling metrics more robust towards false predictions of individual time slots. In this case, the number of consecutive slots with stalling is added to the total stalling time in seconds. Thus, by simple counting of slot predictions, this aggregation method allows to obtain the initial delay, the number of stalling events, the total stalling time, and the stalling ratio of the whole streaming session. Note that the granularity of the initial delay and total stalling time prediction is limited by the length of a time slot, i.e., it is 1 s. However, this granularity should be sufficient for most use cases.

TABLE I: Confusion matrix of slot prediction for CTS features

prediction →	CTS features		CTS/IG features		CTS/CFS features	
↓ actual class	no stalling	stalling	no stalling	stalling	no stalling	stalling
no stalling	130969	2424	130706	2687	131187	2206
stalling	5768	18010	5525	18253	7246	16532

TABLE II: Evaluation of slot prediction for CTS features

	CTS features			CTS/IG features			CTS/CFS features		
	prec.	recall	F1	prec.	recall	F1	prec.	recall	F1
no stalling	0.958	0.982	0.970	0.959	0.980	0.970	0.948	0.983	0.965
stalling	0.881	0.757	0.815	0.872	0.768	0.816	0.882	0.695	0.778
weighted avg.	0.946	0.948	0.946	0.946	0.948	0.946	0.938	0.940	0.937

V. Performance Evaluation

In this section, we evaluate the stalling prediction performance of the ViCrypt system for the investigated feature sets.

A. Real-time Prediction of Stalling

The real-time prediction of stalling assigns each of the time slots to class "no stalling" or "stalling". After training the random forest models with the different feature sets, their performances were checked on the test set, which contained 157,171 time slots.

In the following, the results of the full feature set CTS and the reduced feature sets based on feature selection algorithms are presented. The model using all features reached a very high accuracy of 94.79%, which gives the best performance overall. Also for the reduced features sets, the accuracy was still very high, i.e., 94.78% for the CTS/IG feature set and 93.99% for the CTS/CFS feature set. This means that it is possible to reach a high performance with a very small set of features. Table I presents the confusion matrices with the actual classes of the slots in the rows and the predicted classes in the columns, and Table II gives detailed evaluation results. These include precision (ratio of actual "stalling" slots among all predicted "stalling" slots), recall (ratio of predicted "stalling" slots among all actual "stalling" slots), and F1-measure (harmonic mean of precision and recall). Note that the definitions are analogously in terms of class "no stalling", and that the weighted average of these metrics was computed from the per-class scores weighted by the number of instances of each class. Prediction errors are generally higher for false negatives. While prediction, recall, and F1-measure are very high for class "no stalling", the lower recall of class "stalling" caused by the false negatives is visible. The lower values could be attributed to the very unbalanced measurement data, which only showed very little stalling, and would probably increase after more measurements with very bad network conditions. Still, the results for all models already show a very decent performance in predicting stalling. However, the worse results of the CTS/CFS feature set for stalling prediction indicate that this feature set tends to become too small to accurately describe the instances of the "stalling" class.

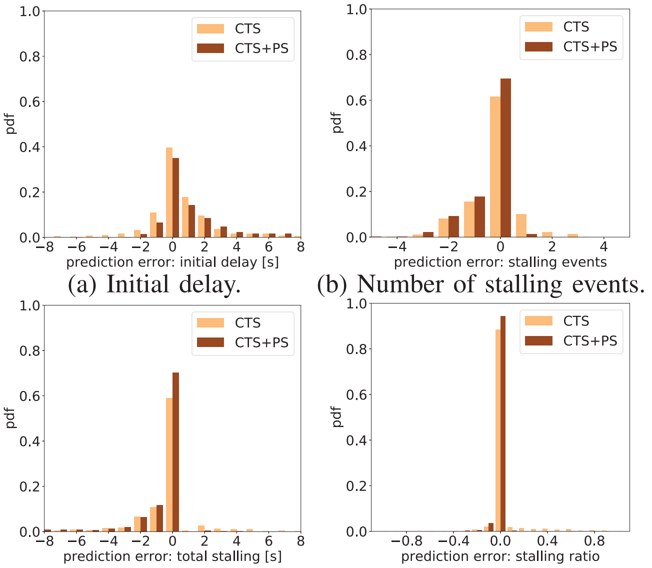
TABLE III: Confusion matrix for CTS+P and CTS+PS

prediction →	CTS+P		CTS+PS	
↓ actual class	no stalling	stalling	no stalling	stalling
no stalling	130884	2509	122992	10401
stalling	6475	17303	6699	17079

TABLE IV: Evaluation of prediction for CTS+P and CTS+PS

	CTS+P			CTS+PS		
	prec.	recall	F1	prec.	recall	F1
no stalling	0.953	0.981	0.967	0.948	0.922	0.935
stalling	0.873	0.728	0.794	0.622	0.718	0.666
weighted avg.	0.940	0.943	0.942	0.899	0.891	0.895

Next, the models with the full feature sets including recurrent features are evaluated. In the CTS+P feature set, only the stalling prediction for the previous slot is considered additionally. After training the random forest model, it was evaluated on the test set, while feeding back the actual prediction of the previous slot as a feature. The first columns of Table III show the corresponding confusion matrices, and first columns of Table IV shows the detailed evaluation results. The model reaches only an accuracy of 94.28% when using the actual previous predictions, which is slightly lower than the performance of the CTS model. If, in theory, perfect previous predictions were available, the model could reach a very high accuracy of 97.74% (F1-measures 0.987 for "stalling", 0.923 for "no stalling", and 0.977 for the weighted average). This shows that propagated prediction errors from previous slots are responsible for the deteriorated performance. For the CTS+PS model, the situation even worsens. The actual performance of 89.12% accuracy is significantly worse than the CTS and CTS+P model, and shows a very high discrepancy from a theoretical performance of 98.98% accuracy (F1-measures 0.994 for "stalling", 0.966 for "no stalling", and 0.990 for the weighted average), which it could have achieved with perfect previous predictions. Also, CTS+PS has much more false positives than CTS+P. As both models with recurrent features perform worse than the CTS model without recurrent features, we can conclude that recurrent features are not beneficial for the prediction. Instead, they propagate prediction errors to subsequent time slots and thereby lower the overall performance. This is quite surprising, as one would a-priori believe that recurrent features would improve prediction performance, by tracking transient effects. We therefore conclude that stalling



(a) Initial delay.

(b) Number of stalling events.

(c) Total stalling time.

(d) Stalling ratio.

Fig. 2: Prediction errors of session-based stalling metrics.

prediction for individual time slots is better realized by relying exclusively on the snapshot, per-slot independent features.

B. Session-based Stalling Prediction

In the following, the results of the session-based QoE prediction are presented. Individual predictions of consecutive time slots are aggregated to obtain stalling metrics on a session level, namely, initial delay, number of stalling events, total stalling time, and stalling ratio. For the stalling prediction of individual slots, the model with the full feature set (CTS) is used. It is compared to the model with all recurrent features (CTS+PS), which showed a very good performance for session-based stalling prediction, even better than CTS+P. This comes as a surprise given that CTS+PS had a worse performance on the stalling prediction for individual slots.

Fig. 2 shows the prediction errors of the session-based stalling prediction, i.e., the distribution of the difference between the predicted metric and the actual metric. Fig. 2a shows the prediction error for the initial delay. The prediction error of the CTS model is plotted in orange, the brown bars represent the CTS+PS model. The initial delay of 39.48% of the sessions can be predicted exactly for CTS. For 68.14% of the sessions, the prediction error is 1 s or less, and for 81.03% it is at most 2 s. As the prediction error is mostly positive, the model tends to overestimate the actual initial delay. For CTS+PS, the numbers are worse. The initial delay can be predicted exactly for 34.98% of the sessions, with an error of 1 s or less for 55.93%, and with an error of 2 s or less for 65.94%.

Fig. 2b shows the distribution of the prediction error for number of stalling events. No error occurs for 61.54% of the sessions. For 24.85%, the number of stalling events was underestimated. Here, CTS+PS performs better and can predict the number of stalling events exactly for 69.48% of the sessions. However, it underestimates for 29.28% of the sessions. In contrast, when considering the prediction error of the total stalling time in Fig. 2c, stalling duration can be predicted with high accuracy. With CTS, for 58.93% of the sessions there is no error, for 70.28% the error is 1 s or less, and for 82.56% it is 3 s or less. CTS+PS performs better and has no error for 70.14%, an error of 1 s or less for 81.92%, and an error of 3 s or less for 90.91% of the sessions.

Finally, Fig. 2d considers the prediction error for the stalling ratio, i.e., the ratio of total stalling time and total playtime of the video. Predicted stalling ratio is within ± 0.05 of the actual stalling ratio (difference in interval from -0.05 to 0.05) for 88.54% of the sessions with CTS, and for 94.43% of the sessions with CTS+PS. For the remaining sessions, CTS tends to overestimate the stalling ratio, while CTS+PS rather underestimates the stalling ratio.

The results show that it is possible to aggregate the individual stalling predictions of consecutive time slots to obtain stalling metrics on session level. The CTS model already showed proper performance with sufficiently small prediction errors. When considering the prediction errors of the number of stalling events, total stalling time, and stalling ratio, the CTS+PS model with recurrent features performed better than CTS, although it showed a worse performance for individual, per-slot stalling predictions. This suggests that, when aggregating individual predictions on the session level, it can be actually beneficial to consider recurrent features for the stalling prediction of the consecutive slots. Only for initial delay, the CTS model could outperform CTS+PS.

VI. Conclusion

We presented ViCrypt, a machine-learning based approach for real-time prediction of QoE-relevant metrics with a fine granularity of only 1 s and a high accuracy. It monitors the encrypted video traffic in three windows (current slot, trend window, session window), from which statistical features are computed and updated with constant memory consumption. By testing on a YouTube data set, we observed that the full feature set (CTS) is not necessary for predicting re-buffering occurrence in real-time, as a similar performance could be reached with half of the features (CTS/IG). Here, especially features of the session window proved to be relevant. However, stalling prediction slowly suffers when more features are excluded (CTS/CFS). Also adding recurrent features degraded the stalling prediction of individual time slots.

We also showed how per-slot individual re-buffering predictions could be aggregated to predict stalling metrics at the session level with low prediction errors. Surprisingly, recurrent features could reduce the prediction error of session-level QoE-relevant metrics (CTS+PS) when aggregating individual stalling predictions. In future works, this effect has to be investigated in more detail. Moreover, the system has to be trained on more slots with stalling to improve performance. Finally, as the approach is very general, it will be transferred to other video streaming services, to predict other QoE factors, such as the visual quality of the streaming, and to extend the prediction to future time slots, which could be beneficial for proactive QoE-aware traffic management.

References

[1] 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.

[2] D. Ghadiyaram, J. Pan, and A. C. Bovik, “A Time-varying Subjective Quality Model for Mobile Streaming Videos with Stalling Events,” in *Proceedings of SPIE Applications of Digital Image Processing XXXVIII*, San Diego, CA, USA, 2015.

[3] K. Zeng, H. Yeganeh, and Z. Wang, “Quality-of-experience of Streaming Video: Interactions between Presentation Quality and Playback Stalling,” in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, USA, 2016.

[4] 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

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

[6] 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 *3rd International Workshop on Quality of Experience Management*, 2019.

[7] M. Seufert, T. Hoßfeld, and C. Sieber, “Impact of Intermediate Layer on Quality of Experience of HTTP Adaptive Streaming,” in *Proceedings of the 11th International Conference on Network and Service Management (CNSM)*, Barcelona, Spain, 2015.

[8] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, and P. Tran-Gia, “Quantification of YouTube QoE via Crowdsourcing,” in *Proceedings of the International Workshop on Multimedia Quality of Experience - Modeling, Evaluation, and Directions (MQoE)*, Dana Point, CA, USA, 2011.

[9] R. Schatz, T. Hoßfeld, and P. Casas, “Passive YouTube QoE Monitoring for ISPs,” in *Proceedings of the 2nd International Workshop on Future Internet and Next Generation Networks (FINGNet)*, Palermo, Italy, 2012.

[10] P. Casas, R. Schatz, and T. Hoßfeld, “Monitoring YouTube QoE: Is Your Mobile Network Delivering the Right Experience to Your Customers?” in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, 2013.

[11] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, “YoMo: A YouTube Application Comfort Monitoring Tool,” in *Proceedings of the 1st Workshop of Quality of Experience for Multimedia Content Sharing (QoEMCS)*, Tampere, Finland, 2010.

[12] M. Eckert, T. M. Knoll, and F. Schlegel, “Advanced MOS Calculation for Network Based QoE Estimation of TCP Streamed Video Services,” in *Proceedings of the 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, Gold Coast, Australia, 2013.

[13] P. Ameigeiras, A. Azcona-Rivas, J. Navarro-Ortiz, J. J. Ramos-Munoz, and J. M. Lopez-Soler, “A Simple Model for Predicting the Number and Duration of Rebuffering Events for YouTube Flows,” *IEEE Communications Letters*, vol. 16, no. 2, pp. 278–280, 2012.

[14] P. Szilágyi and C. Vulkán, “Network side Lightweight and Scalable YouTube QoE Estimation,” in *Proceedings of the IEEE International Conference on Communications (ICC)*, London, UK, 2015.

[15] International Telecommunication Union, “ITU-T Recommendation P.1203: Parametric Bitstream-based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services over Reliable Transport,” 2016. [Online]. Available: https://www.itu.int/rec/T-REC-P.1203/en

[16] 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 (HotMobile)*, Santa Barbara, CA, USA, 2014.

[17] 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.

[18] 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.

[19] 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 *Proceedings of the 5th IEEE International Workshop on Quality of Experience for Multimedia Communications (QoEMC)*, Washington, DC, USA, 2016.

[20] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, “Measuring Video QoE from Encrypted Traffic,” in *Proceedings of the ACM Internet Measurement Conference (IMC)*, Santa Monica, CA, USA, 2016.

[21] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, “Buffest: Predicting buffer conditions and real-time requirements of http(s) adaptive streaming clients,” in *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys)*, Taipei, Taiwan, 2017.

[22] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, “emimic: Estimating http-based video qoe metrics from encrypted network traffic,” in *TMA Conference 2018 - Network Traffic Measurement and Analysis Conference*, 2018.

[23] M. H. Mazhar and M. Z. Shafiq, “Real-time video quality of experience monitoring for https and quic,” in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018.

[24] P. Pébay, “Formulas for Robust, One-Pass Parallel Computation of Covariances and Arbitrary-Order Statistical Moments,” Sandia National Laboratories, Tech. Rep., 2008.

[25] T. Karagkioules, D. Tsilimantos, S. Valentin, F. Wamser, B. Zeidler, M. Seufert, F. Loh, and P. Tran-Gia, “A Public Dataset for YouTube’s Mobile Streaming Client,” in *Proceedings of the 2nd Workshop on Mobile Network Measurement (MNM)*, Vienna, Austria, 2018.

[26] M. Seufert, B. Zeidler, F. Wamser, T. Karagkioules, D. Tsilimantos, F. Loh, P. Tran-Gia, and S. Valentin, “A Wrapper for Automatic Measurements with YouTube’s Native Android App,” in *Proceedings of the 2nd Network Traffic Measurement and Analysis Conference (TMA)*, Vienna, Austria, 2018.

[27] 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 *Proceedings of the European Conference on Networks and Communications (EuCNC)*, Paris, France, 2015.

Appendix

Table V presents the list of all features sorted by their information gain towards class “stalling”, and indicates in which of the reduced feature sets they were included. Prefix letter ‘t’ indicates a feature of the trend window, prefix letter ‘s’ indicates a feature of the session window, the other features are computed on the current time slot. Additionally, prefix letters ‘d’ and ‘u’ indicate features for downlink and uplink traffic, respectively.

TABLE V: List of features sorted by information gain towards class “stalling”

information gain	name	feature set
0.445414	s_volume	IG, CFS
0.436852	sd_volume	IG, CFS
0.421901	s_num_packets	IG
0.415934	sd_num_packets	IG
0.392734	su_num_packets	IG
0.37201	sd_packet_size_skew	IG, CFS
0.369883	su_packet_size_skew	IG
0.366606	sd_packet_size_cvar	IG, CFS
0.366449	sd_packet_size_kurt	IG
0.364596	su_packet_size_mean	IG
0.362772	sd_iat_kurt	IG, CFS
0.362404	su_packet_size_kurt	IG, CFS
0.36028	sd_packet_size_mean	IG
0.357725	sd_iat_skew	IG, CFS
0.355849	sd_packet_size_var	IG
0.355849	sd_packet_size_stddev	IG
0.354076	sd_volume_ratio	IG
0.354076	su_volume_ratio	IG
0.348692	su_packet_size_var	IG
0.348683	su_packet_size_stddev	IG
0.345504	sd_regression_intcpt	IG, CFS
0.344576	su_volume	IG
0.341763	sd_burst_throughput	IG
0.336476	su_iat_kurt	IG, CFS
0.331278	s_burst_duration	IG
0.328932	s_volume_tcp	IG
0.32738	su_iat_skew	IG
0.320215	s_burst_throughput	IG
0.319094	su_burst_duration	IG
0.306033	sd_regression_slope	IG
0.304495	sd_iat_mean	IG
0.297394	sd_burst_duration	IG
0.292278	su_regression_intcpt	IG, CFS
0.291357	su_iat_mean	IG
0.290869	su_num_packets_ratio	IG
0.290842	sd_num_packets_ratio	IG
0.283743	sd_iat_max	IG
0.27512	sd_iat_stddev	IG
0.268584	sd_iat_var	IG
0.268054	su_iat_stddev	IG
0.267332	sd_iat_cvar	IG
0.266696	su_iat_max	IG
0.264366	s_throughput	IG, CFS
0.263358	sd_throughput	IG, CFS
0.259922	su_iat_var	IG
0.257448	su_packet_size_cvar	IG
0.239231	su_iat_cvar	IG
0.216976	s_num_packets_tcp_ratio	IG
0.216976	s_num_packets_udp_ratio	IG
0.203673	s_volume_tcp_ratio	IG
0.203673	s_volume_udp_ratio	IG
0.192402	s_volume_udp	IG
0.188676	su_burst_throughput	IG
0.180672	s_num_packets_tcp	IG
0.178991	s_num_packets_udp	IG
0.156065	su_throughput	IG
0.151182	su_regression_slope	IG
0.141154	su_packet_size_max	IG, CFS
0.1147	td_volume	IG
0.1147	td_throughput	IG
0.10937	t_volume	IG
0.10937	t_throughput	IG
0.108723	ordinal_sequence_number	IG, CFS
0.104279	td_burst_throughput	IG, CFS
0.101206	t_burst_throughput	IG
0.098643	t_num_packets	IG
0.095487	td_num_packets	IG
0.09491	tu_num_packets	IG
0.093312	t_volume_tcp	IG, CFS
0.082858	t_num_packets_tcp	IG
0.070875	d_throughput	IG
0.070875	d_volume	IG
0.067616	volume	IG
0.067616	throughput	IG
0.067504	sd_time_after_last_packet	IG
0.065118	tu_packet_size_mean	IG
0.062839	tu_packet_size_stddev	IG
0.062839	tu_packet_size_var	IG
0.059359	td_volume_ratio	IG
0.059359	tu_volume_ratio	IG
0.059044	d_burst_throughput	IG
0.058235	tu_packet_size_kurt	IG
0.058018	packets	IG
0.057303	s_time_after_last_packet	IG
0.056921	u_num_packets	IG
0.056885	tu_packet_size_skew	IG
0.056182	burst_throughput	IG
0.055997	td_regression_slope	IG
0.055465	tu_throughput	IG
0.055465	tu_volume	IG
0.054434	tu_packet_size_max	IG
0.053616	td_packet_size_mean	IG
0.053234	td_packet_size_skew	IG
0.052582	d_num_packets	IG
0.052532	volume_tcp	IG
0.052429	td_packet_size_kurt	IG
0.050595	su_time_after_last_packet	IG
0.049489	td_packet_size_cvar	
0.048145	t_volume_udp	
0.047044	td_packet_size_stddev	
0.047044	td_packet_size_var	
0.046475	tu_iat_mean	
0.046467	td_iat_mean	
0.043254	packets_tcp	
0.042758	tu_packet_size_cvar	
0.040324	t_num_packets_udp	
0.038509	u_volume	
0.038509	u_throughput	
0.037025	td_num_packets_ratio	
0.037025	tu_num_packets_ratio	
0.036336	volume_udp	
0.033925	t_burst_duration	
0.033287	u_packet_size_max	
0.031948	sd_packet_size_max	
0.030887	td_packet_size_min	
0.030404	u_iat_mean	
0.030395	u_packet_size_mean	
0.030206	td_iat_stddev	
0.030201	td_iat_var	
0.030051	tu_burst_duration	
0.029879	tu_iat_stddev	
0.02985	tu_iat_var	
0.029392	td_burst_duration	
0.029374	t_num_packets_tcp_ratio	CFS
0.029374	t_num_packets_udp_ratio	
0.029005	packets_udp	
0.028376	tu_iat_max	
0.028296	t_time_until_first_packet	
0.028226	d_regression_slope	
0.0279	burst_duration	
0.027374	t_volume_tcp_ratio	
0.027374	t_volume_udp_ratio	
0.027118	tu_burst_throughput	
0.02658	u_packet_size_var	
0.02658	u_packet_size_stddev	
0.025533	tu_time_until_first_packet	
0.025499	u_packet_size_kurt	
0.024635	d_volume_ratio	
0.024635	u_volume_ratio	
0.02449	u_packet_size_skew	
0.023849	d_burst_duration	
0.023735	d_packet_size_mean	
0.023654	u_iat_stddev	
0.023478	u_iat_var	
0.023054	u_burst_throughput	
0.022819	u_burst_duration	
0.022002	d_iat_mean	
0.02192	d_iat_var	
0.02189	u_iat_max	
0.021875	d_iat_stddev	
0.021664	tu_packet_size_min	
0.020924	d_packet_size_skew	
0.020446	td_packet_size_max	
0.020321	d_num_packets_ratio	
0.020321	u_num_packets_ratio	
0.020233	td_time_until_first_packet	
0.020179	d_packet_size_kurt	
0.019905	d_packet_size_cvar	
0.019887	d_packet_size_min	
0.019689	td_iat_max	
0.019135	u_packet_size_min	
0.017708	td_iat_kurt	
0.017417	d_packet_size_stddev	
0.017417	d_packet_size_var	
0.016183	tu_iat_skew	
0.015154	td_regression_intcpt	
0.014949	u_packet_size_cvar	
0.014605	d_iat_max	
0.013792	sd_time_until_first_packet	
0.013619	packets_tcp_ratio	
0.013619	packets_udp_ratio	
0.013329	tu_iat_kurt	
0.013144	tu_regression_intcpt	
0.012993	tu_iat_cvar	
0.012927	tu_regression_slope	
0.012821	td_iat_cvar	
0.012286	volume_udp_ratio	
0.012286	volume_tcp_ratio	
0.012104	td_iat_skew	
0.011111	t_time_after_last_packet	
0.009766	u_iat_min	
0.009712	u_regression_slope	
0.009594	d_packet_size_max	
0.009285	tu_time_after_last_packet	
0.009176	td_time_after_last_packet	
0.008269	u_iat_cvar	
0.00826	time_after_last_packet	
0.00781	time_until_first_packet	
0.007713	u_regression_intcpt	
0.007657	u_time_until_first_packet	
0.006812	u_time_after_last_packet	
0.006741	d_time_after_last_packet	
0.006722	d_iat_kurt	
0.006528	u_iat_skew	
0.006521	d_time_until_first_packet	
0.006388	u_iat_kurt	
0.00631	tu_iat_min	
0.005805	su_time_until_first_packet	
0.004922	sd_packet_size_min	CFS
0.004626	d_iat_cvar	
0.004466	d_regression_intcpt	
0.003081	d_iat_skew	
0.001196	d_iat_min	
0.001016	td_iat_min	
0.000805	su_iat_min	
0.000753	sd_iat_min	
0	su_packet_size_min	
0	s_time_until_first_packet	