Stream-based machine learning for real-time QoE analysis of encrypted video streaming traffic

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

Stream-based Machine Learning for Real-time QoE Analysis of Encrypted Video Streaming Traffic

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

*AIT Austrian Institute of Technology GmbH, Vienna, Austria

†Huawei Technologies, Department of R&D, Shenzhen, P.R. China

michael.seufert.fl@ait.ac.at, pedro.casas@ait.ac.at, nikolas.wehner@ait.ac.at, lig619@huawei.com, kuangli@huawei.com

Abstract—**As stalling is the worst Quality of Experience (QoE) degradation of HTTP adaptive video streaming (HAS), this work presents a stream-based machine learning approach, ViCrypt, which analyzes stalling of YouTube streaming sessions in real-time from encrypted network traffic. The video streaming session is subdivided into a stream of short time slots of 1 s length, while considering two additional macro windows each for the current streaming trend and the whole ongoing streaming session. Constant memory features are extracted from the encrypted network traffic in these three windows in a stream-based fashion, and fed into a random forest model, which predicts whether the current time slot contains stalling or not. The presented system can predict stalling with a very high accuracy and the finest granularity to date (1 s), and thus, can be used in networks for real-time QoE analysis from encrypted YouTube video streaming traffic. The independent predictions for each consecutive slot of a streaming session can further be aggregated to obtain stalling estimations for the whole session. Thereby, the proposed method allows to quantify the initial delay, as well as the overall number of stalling events and the stalling ratio, i.e., the ratio of total stalling time and total playback time.**

I. INTRODUCTION

Video services are the most popular and most demanding applications 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, application providers and Internet service providers use the concept of Quality of Experience (QoE) to quantify the subjective experience and satisfaction of their customers with the network and the service. For video streaming, the most severe QoE degradations are the playback interruptions (stalling) and the waiting time until the start of the playback (initial delay) [1]–[3]. While these degradations have been partially mitigated by adapting the video bit rate to the network conditions (HTTP Adaptive Streaming, HAS), stalling is still the most annoying QoE degradation, which network and service providers have to avoid.

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 occurring stalling events. For this, network operators favor real-time QoE monitoring in the network as an input to their traffic management decisions. However, the trend towards encrypted traffic (HTTPS) significantly reduced the visibility of network operators. It is no longer possible to use Deep Packet Inspection (DPI) based methods to analyze the video data contained in each packets in order to reconstruct the streaming process and the video buffer. The encrypted packets only allow to obtain basic information about the streaming process, such as packet sizes and inter-arrival times of packets.

In this work, ViCrypt, a stream-based machine learning approach is presented, which predicts stalling of YouTube streaming sessions in real-time from such basic features. The streaming session is subdivided into short time slots of 1 s length. Features are computed for each time slot, as well as for corresponding macro windows, which consider the current streaming trend and the whole ongoing streaming session. These features are computed in a stream-based fashion with constant memory consumption and 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. The independent predictions for each consecutive slot of a streaming session can further be aggregated to obtain stalling estimations for the whole session. Thereby, ViCrypt allows to quantify the initial delay, as well as the overall number of stalling events and the stalling ratio, i.e., the ratio of total stalling time and total playback time.

The remainder of the paper is structured as follows. Section II describes related works on QoE of HAS and network-based QoE monitoring. Section III introduces the concept of ViCrypt, and presents the features and training of the random forest model. A performance evaluation of the ViCrypt system is conducted in Section IV, and Section V concludes.

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 [5], 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 [6]. [7]–[9] transferred the approach of [10] 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 [11], which supported more video encodings and container formats, and by [12], which predicted stalling in LTE networks. [13] 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 parametric bit stream-based quality assessment model for HAS was standardized (P.1203, [14]), which predicts the MOS from stream inspection and supports four modes of input information.

Due to the trend towards end-to-end encryption, DPI-based approaches cannot be applied any longer. This has motivated a recent trend in QoE-based network monitoring using low-level network measurements rather than relying on application-layer metrics. While some approaches explicitly tackle the QoE of mobile apps, including [15]–[17], there are also general approaches for QoE analyses based on network-layer monitoring of encrypted video streaming traffic. Authors in [18] 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 [19], where authors rely on real cellular network measurements to predict typical QoE indicators for streaming services (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 as a promising technique for large-scale quality monitoring and prediction. [20] focuses on the reconstruction of buffered playtime at the video player side, as previously done in [7], but for encrypted network traffic. This is leveraged to estimate video QoE metrics in [21]. [22], 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. The considered features are derived from IP or TCP/UDP headers only.

Different from all these papers, ViCrypt 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 also obtain accurate stalling statistics on a session level.

III. METHODOLOGY

This work presents ViCrypt, a system for real-time QoE analysis of encrypted video streaming traffic. As stalling is still the major QoE degradation for HAS, the first version of ViCrypt focuses on the detection of stalling. The video streaming session is subdivided into a sequence of time slots, which have a constant length. After a slot ended, the stalling prediction is applied in order to detect stalling in real-time. Throughout this work, a slot length of 1 s is used, which constitutes a decent trade-off between stalling detection delay and accuracy. The individual predictions of each time slot can later be aggregated to obtain a session-level stalling evaluation, i.e., the initial delay as well as the number and length of stalling events. The remainder of this section describes the feature extraction, the data set generation, and the training of the ML model.

A. Feature Extraction

The stalling prediction for the current time slot can only rely on features extracted from the traffic of the current or past 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 slot, the proposed system keeps track of only two additional macro windows, namely, the trend window and the session window. The trend window comprises t slots in a sliding-window fashion, and thereby, contains all traffic of the current time slot and the $t-1$ most recent slots. Throughout this work, a trend size of $t = 3$ is used, which means that, for each slot, the corresponding trend window contains the traffic of the current slot and the previous two time slots. The session window is a macro window, which covers all traffic of the session so far, i.e., it includes all previous slots including the current time slot. The features of each current slot, trend window, and session window are computed in an online 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). Moreover, the number and byte volume of TCP and UDP packets is counted, 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 the slot until the first packet, the time after the last packet until the slot ends, and the burst duration, i.e., the time between the first packet and the last packet of the time slot. All features are again computed for the total traffic, as well as for uplink and downlink traffic. The average throughput of the slot (traffic volume divided by slot length) and the burst throughput (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 online 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 [23], which can compute the first four moments of any distribution in an online 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 value. These distribution-based features were computed for the packet size and the inter-arrival time of packets, i.e., the time between two consecutive packets. For this, uplink and downlink traffic were distinguished.

This results in 69 basic features for the traffic in a time slot. As described above, two macro windows are additionally considered, namely, the trend and the session window, for which the same 69 basic features are computed. Together with the ordinal sequence number of the current time slot, this sums up to 208 features, which characterize each slot of $1\,s$ length. Note again that there is a constant memory consumption for the computation of the basic features of each time slot. However, in order to keep track of the trend windows of size t, not only the current trend window, but additionally, $t-1$ future trend windows have to be maintained and updated. These future trend windows are the windows, which will become trend windows in $1, \ldots, t-1$ windows, but already have to consider the traffic in the current time slot. In contrast, only a single session window is needed, as it just hast to accumulate all traffic of the whole session. Thus, in total, $t+2$ windows with 69 features each have to be maintained and updated at all times, i.e., current time slot, trend window, session window, and $t-1$ future trend windows.

B. Data Set

Over a period of several weeks in summer 2018, 4714 YouTube video sessions were streamed and recorded. Therefore, a Java-based monitoring tool similar to [24] was used. It used the Selenium browser automation library to automatically start a Chrome browser and browse to a single random YouTube video and stream for $180\,s$ or until the video end. The chrome browser was configured such that all HTTP requests were logged to a file (`-log-net-log`) and QUIC traffic was enabled (`--enable-quic`). A JavaScript-based monitoring script [4], [25] was injected into the web page to record every $250\,ms$ the current timestamp, as well as the current video playtime, buffered playtime, video resolution, and player state. This application-layer information about the streaming session was also logged to a file.

The video sessions were streamed with highly diverse network characteristics to reach a highly generalizable model.

The videos were either streamed from a home or corporate WiFi network, or an LTE mobile network. For some sessions a firewall was enabled, which blocked all QUIC traffic, such that the videos were streamed via TCP. The maximum bandwidth was roughly $20\,Mbps$. Additionally, some streaming sessions faced bandwidth limitations, which were applied to limit both up- and downlink traffic. The bandwidth limitations were either constant on a level of $300\,Kbps$, $1\,Mbps$, $3\,Mbps$, or $5\,Mbps$, or they fluctuated between these levels every 1-5 minutes. Moreover, during some video streaming sessions, user actions were emulated via Selenium, i.e., the streaming was paused and optionally resumed later, or a jump to a different playback position was executed. The reason was to obtain some additional stalling caused by the bandwidth limitations and the emulated user actions.

During the whole streaming session, the network traffic was captured using tshark. The network trace files were parsed with a Java parser based on the fast Kaitai Struct pcap parser[2], which extracted and logged basic packet information (timestamp, source IP, source port, destination IP, destination port, size), as well as DNS lookup responses to obtain a mapping between IP addresses and domain names. In each network trace, YouTube video flows were identified based on the domain name (googlevideo.com), and features were only extracted from the traffic of all video flows, i.e., all other non-YouTube flows were ignored.

Finally, also the recently published open dataset [26], [27] was considered, which collected measurements from the mobile Android YouTube app. As the data set contains multiple measurements of the same scenario (location, network condition, video content), only three random iterations were selected from each of the 45 scenarios. Thus, in total 135 YouTube app video sessions were added to the data set.

80% of the video sessions (randomly selected) were considered for training, and the remaining video sessions comprised the test set. This means 3879 sessions were used for training, and the test set consisted of 970 sessions. Figure 1 shows the stalling characteristics of the training set (solid lines) and the test set (dashed lines) as cumulative distribution functions (CDF). It can be seen that the initial delay time (excluding page load time and stalling during playback) is mostly low. 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 face 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 high having a 95-percentile of $6.4\,s$ and a 99-percentile of $19.5\,s$. While these numbers were given for the training set, it can be seen from the similarity of the shapes that the test set shows the same characteristics as the training set.

---

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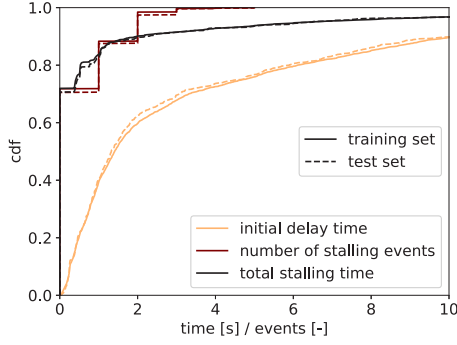
[2]https://kaitai.io/

Fig. 1: Stalling characteristics of training and test set.

C. Training

The following steps were executed with the open source machine learning software Weka[3]. The training data consisted in total of 635209 time slots. As the training set contained much less slots with stalling (class = 1, 18.46%) than slots without stalling (class = 0, 81.54%), 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 635209 instances each. During the preparation of the bootstrapped training set, the order of the slots was also randomized, which helps to avoid any serial-position effects in the data set during training.

Four different feature sets are used throughout this work. The first set (C feature set) includes only the 70 features of the current time slot. Thereby, only current information about the streaming is used. The second set (CT feature set) adds all 69 features of the trend window. Thus, it additionally takes the last 3 s of the streaming session into account. The third set (CS feature set) consists of all features of the current time slot and all features of the session window. This means, in addition to the current time slot, it considers the session window, i.e., the entire past streaming session. The last set (CTS feature set) is straightforward and uses all 208 extracted features (current slot, trend window, session window).

D. 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. The classification accuracy of the random forest model is evaluated on the test set.

First, the classification accuracy is evaluated per time slot. This means, the results of the bare classification task for any 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 time slot or not.

Finally, the classification accuracy is evaluated per session. For this, individual and independent predictions of the se-

quence 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 (time 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 time 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 stalling time prediction is limited by the time slot length, i.e., it is 1 s. However, this granularity should be sufficient for most use cases.

IV. PERFORMANCE EVALUATION

In this section, the real-time prediction of stalling per time slot, and the aggregation of individual, consecutive slot predictions to session-level stalling metrics are evaluated.

A. Real-time Prediction of Stalling

The real-time prediction of stalling assigns each of the current time slot 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 157171 time slots.

The model using only features of the current time slot (C feature set) reached already a decent accuracy of 85.86%. When adding more features, the accuracy increases to 88.34% (CT) and 94.67% (CS). Here, it can be seen that the features of the session window are better suited for the stalling prediction, as they provide a higher gain compared to the features of the trend window.

Table I shows the corresponding confusion matrices. Rows indicate the actual class of the time slot, while the columns represent the predicted classes. It can be seen that for C and CT the error is generally higher for false positives, i.e., slots without stalling were considered to contain stalling. For CS, the error changes towards more false negatives, i.e., "stalling" time slots were not recognized by the model. Table II further evaluates the prediction in terms of 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. While the metrics are
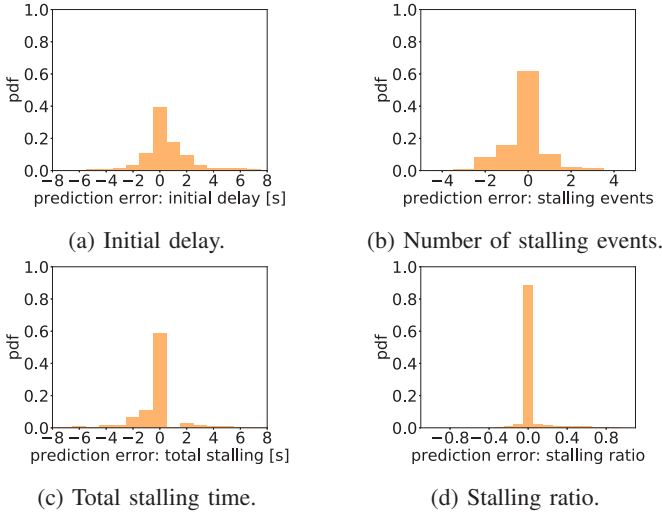
(a) Initial delay.

(b) Number of stalling events.

(c) Total stalling time.

(d) Stalling ratio.

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

generally high for the "no stalling" class, the prediction of "stalling" is suffering. 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.

In the following, the results of the full feature set CTS are presented. The model using all features reached a very high accuracy of 94.79%, which gives the best performance overall. Table III presents the confusion matrix, and Table IV gives detailed evaluation results. It can be seen that the error is 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. Still, the results show a very decent performance in predicting stalling in real-time for individual time slots.

B. Session-based Stalling Prediction

In the following, the results of the session-based stalling prediction are presented. For this, individual predictions of consecutive time slots were aggregated to obtain stalling metrics on a session level, namely, initial delay, number of stalling events, total stalling time, and stalling ratio. This also means that all results inherit the granularity of the individual slots of 1 s. For the stalling prediction, the random forest model with the full feature set (CTS) is used.

Figure 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. Figure 2a shows the prediction error for the initial delay. It can be seen that the initial delay can be predicted exactly, i.e., with the granularity of 1 s, for 39.48% of the sessions. 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.

Figure 2b shows the distribution of the prediction error for number of stalling events. No error occurs for 61.54% of the sessions, including both sessions with and without stalling. For 24.85% of the sessions, the number of stalling events was

underestimated. When considering the prediction error of the total stalling time in Figure 2c, it can be seen that the stalling duration can be predicted with high accuracy. 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. Finally, Figure 2d considers the prediction error for the stalling ratio, i.e., the ratio of total stalling time and total playtime of the video. It can be seen that the 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. For the remaining sessions, the model tends to overestimate the stalling ratio. The high accuracy here compared to the results above is a consequence of the generally small amount of stalling with respect to the length of the sessions.

All in all, the results show that it is possible to aggregate the individual stalling predictions of consecutive windows to obtain stalling metrics on session level. Moreover, the ViCrypt system showed a decent performance for obtaining stalling metrics with sufficiently small prediction errors.

V. CONCLUSION

This paper presented ViCrypt, a stream-based machine learning approach for real-time stalling prediction from encrypted video streaming traffic. The approach is based on individual stalling predictions for time slots of 1 s. A simple random forest model was trained on a dataset of monitored YouTube video streaming sessions with different sets of features. All features were computed in a stream-based fashion with constant memory consumption. While session-based features showed a higher gain than trend-based features, the feature set with all features (current slot, trend window, session window) performed best. The approach reached a very high accuracy and precision. Only for recall, the numbers were slightly lower, due to the highly imbalanced dataset, which contained relatively few stalling.

Furthermore, this paper showed that the individual predictions of consecutive time slots could be aggregated to predict stalling metrics on session level with low prediction errors. Thus, the ViCrypt system allows to monitor stalling in real-time, which is crucial to analyze and properly manage the QoE of video streaming, even with encrypted traffic. In future works, the system has to be trained on more windows with stalling to come up with an improved performance. Finally, the ViCrypt approach will be transferred to predict also other QoE factors, such as the visual quality of the streaming.

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 *SPIE Applications of Digital Image Processing XXXVIII*, 2015.

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

TABLE I: Confusion matrix for slot prediction for C, CT, CS features

	C features		CT features		CS features	
	no stalling	stalling	no stalling	stalling	no stalling	stalling
no stalling	120084	13309	123340	10053	130720	2673
stalling	8913	14865	8276	15502	5710	18068

TABLE II: Evaluation of slot prediction for C, CT, CS features

	C features			CT features			CS features		
	precision	recall	F1	precision	recall	F1	precision	recall	F1
no stalling	0.931	0.900	0.915	0.937	0.925	0.931	0.958	0.980	0.969
stalling	0.528	0.625	0.572	0.607	0.652	0.628	0.871	0.760	0.812
weighted avg.	0.870	0.859	0.863	0.887	0.883	0.885	0.945	0.947	0.945

TABLE III: Confusion matrix for slot prediction for CTS features

	CTS features	
	no stalling	stalling
no stalling	130969	2424
stalling	5768	18010

TABLE IV: Evaluation of slot prediction for CTS features

	CTS features		
	precision	recall	F1
no stalling	0.958	0.982	0.970
stalling	0.881	0.757	0.815
weighted avg.	0.946	0.948	0.946

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

[6] T. Hoßfeld, R. Schatz, M. Seufert, M. Hirth, T. Zinner, and P. Tran-Gia, "Quantification of YouTube QoE via Crowdsourcing," in *International Workshop on Multimedia Quality of Experience - Modeling, Evaluation, and Directions (MQoE)*, 2011.

[7] R. Schatz, T. Hoßfeld, and P. Casas, "Passive YouTube QoE Monitoring for ISPs," in *2nd International Workshop on Future Internet and Next Generation Networks (FINGNet)*, 2012.

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

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

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

[11] M. Eckert, T. M. Knoll, and F. Schlegel, "Advanced MOS Calculation for Network Based QoE Estimation of TCP Streamed Video Services," in *7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2013.

[12] 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.

[13] P. Szilágyi and C. Vulkán, "Network side Lightweight and Scalable YouTube QoE Estimation," in *IEEE International Conference on Communications (ICC)*, 2015.

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

[15] 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 (HotMobile)*, 2014.

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

[17] P. Casas, A. D'Alconzo, F. Wamser, M. Seufert, B. Gardlo, A. Schwind, P. Tran-Gia, and R. Schatz, "Predicting QoE in Cellular Networks using Machine Learning and in-Smartphone Measurements," in *9th International Conference on Quality of Multimedia Experience (QoMEX)*, 2017.

[18] 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 *5th IEEE International Workshop on Quality of Experience for Multimedia Communications (QoEMC)*, 2016.

[19] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring Video QoE from Encrypted Traffic," in *ACM Internet Measurement Conference (IMC)*, 2016.

[20] 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 (MMSys)*, 2017.

[21] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "eMIMIC: Estimating HTTP-based Video QoE Metrics from Encrypted Network Traffic," in *2nd Network Traffic Measurement and Analysis Conference (TMA)*, 2018.

[22] M. H. Mazhar and M. Z. Shafiq, "Real-time Video Quality of Experience Monitoring for HTTPS and QUIC," in *IEEE INFOCOM*, 2018.

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

[24] A. Schwind, M. Seufert, Ö. Alay, P. Casas, P. Tran-Gia, and F. Wamser, "Concept and Implementation of Video QoE Measurements in a Mobile Broadband Testbed," in *IEEE/IFIP Workshop on Mobile Network Measurement (MNM)*, 2017.

[25] 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)*, 2015.

[26] 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 *2nd Workshop on Mobile Network Measurement (MNM)*, Vienna, Austria, 2018.

[27] 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 *2nd Network Traffic Measurement and Analysis Conference (TMA)*, Vienna, Austria, 2018.