ViCrypt to the Rescue: Real-Time, Machine-Learning-Driven Video-QoE Monitoring for Encrypted Streaming Traffic

Sarah Wassermann, Michael Seufert, *Member, IEEE*, Pedro Casas, *Member, IEEE*, Li Gang, and Kuang Li

***Abstract*—Video streaming is the killer application of the Internet today. In this article, we address the problem of real-time, passive Quality-of-Experience (QoE) monitoring of HTTP Adaptive Video Streaming (HAS), from the Internet-Service-Provider (ISP) perspective – i.e., relying exclusively on in-network traffic measurements. Given the wide adoption of end-to-end encryption, we resort to machine-learning (ML) models to estimate multiple key video-QoE indicators (KQIs) from the analysis of the encrypted traffic. We present ViCrypt, an ML-driven monitoring solution able to infer the most important KQIs for HTTP Adaptive Streaming (HAS), namely stalling, initial delay, video resolution, and average video bitrate. ViCrypt performs estimations in real-time, during the playback of an ongoing video-streaming session, with a fine-grained temporal resolution of just one second. For this, it relies on lightweight, stream-like features continuously extracted from the encrypted stream of packets. Empirical evaluations on a large and heterogeneous corpus of YouTube measurements show that ViCrypt can infer the targeted KQIs with high accuracy, enabling large-scale passive video-QoE monitoring and proactive QoE-aware traffic management. Different from the state of the art, and besides real-time operation, ViCrypt is not bound to coarse-grained KQI-classes, providing better and sharper insights than other solutions. Finally, ViCrypt does not require chunk-detection approaches for feature extraction, significantly reducing the complexity of the monitoring approach, and potentially improving on generalization to different HAS protocols used by other video-streaming services such as Netflix and Amazon.**

***Index Terms*—Network monitoring, QoE, HTTP adaptive video streaming, machine learning, encrypted traffic.**

I. Introduction

VIDEO streaming is one of the key applications of the Internet. To satisfy end users and avoid customer churn, Internet Service Providers (ISPs) strive to deliver a high video-streaming Quality of Experience (QoE). 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. The goal of network operators is not only to operate their networks efficiently, but also to avoid severe degradations of the subjective experience.

Network-traffic monitoring has traditionally relied on the usage of Deep-Packet-Inspection (DPI) techniques to understand the performance of the services and applications used by their customers. In particular for video-streaming services, analyzing the payload of the packets containing video information could be used to understand the status of the video-player buffer [1], [2], shedding light on the video-playback performance. ISPs have no longer access to such information in the nowadays widespread scenario of TLS encryption, turning the monitoring of video-streaming QoE into a daunting and challenging task.

In this article, we present ViCrypt, an ML-driven monitoring solution able to estimate and continuously track the most relevant Key QoE Indicators (KQIs) for HTTP Adaptive Streaming (HAS), in real time, and using as input features derived from raw network-traffic measurements – basically, packet size and arrival time. The targeted KQIs include re-buffering events or stallings, initial playback delay, video resolution, and average video bitrate. To do so, ViCrypt analyzes ongoing streaming sessions using fine-grained time slots of one-second length, computing multiple lightweight, statistical features from the video traffic in a stream-based fashion. Besides per-time-slot features, ViCrypt additionally computes features for different temporal aggregations of past slots, including a short-term memory capturing the last t slots, as well as a long-term memory, aggregating all time slots since the start of the video session. At the end of each one-second time slot, all these features are fed into ML models, which estimate the video resolution, average bitrate, and stalling of this slot. To the best of our knowledge, this is by now the finest time granularity for real-time estimation of Key QoE Indicators (KQIs) from encrypted traffic. ViCrypt provides fine-grained estimations, either by building classification models with many quality classes – e.g., for video resolution analysis – or by building regression models – e.g., for video-bitrate analysis. Such a combined fine-grained temporal scope and estimation resolution allows ViCrypt to provide better and sharper insights into video HAS QoE than other solutions.

Indeed, while there have been already multiple proposals presented in the past to deal with this inference problem, the

Manuscript received May 3, 2020; revised September 24, 2020; accepted October 12, 2020. Date of publication November 6, 2020; date of current version December 9, 2020. The associate editor coordinating the review of this article and approving it for publication was G. Casale. *(Corresponding author: Pedro Casas.)*

Sarah Wassermann and Pedro Casas are with the Center for Digital Safety and Security, AIT Austrian Institute of Technology, 1210 Vienna, Austria (e-mail: sarah@wassermann.lu; pedro.casas@ait.ac.at).

Michael Seufert is with the Institute of Computer Science, University of Würzburg, 97070 Würzburg, Germany (e-mail: michael.seufert@uni-wuerzburg.de).

Li Gang and Kuang Li are with the Huawei Technologies Company Ltd., Shenzhen 518129, China.

Digital Object Identifier 10.1109/TNSM.2020.3036497

contributions brought by ViCrypt exceed the state of the art as follows:

1 – Fine-Grained, Real-Time Operation: ViCrypt estimates the most important KQIs, i.e., initial delay, stalling, and visual quality, as well as the video bitrate in real time during the streaming of a video session with a fine-grained temporal resolution of just one second. This is the smallest granularity proposed so far, enabling quick anomaly detection and troubleshooting approaches, as well as proactive traffic management.

2 – Stream-Based Feature Computation in Constant Memory Without Requiring Chunk Detection: Different from all previously presented proposals, ViCrypt continuously extracts features from the encrypted stream of packets in a stream-like manner, using a bounded and lightweight memory footprint. This enables the execution of ViCrypt on top of memory-constrained hardware, such as set-top boxes or home routers, which are nowadays the preferred devices for conducting end-customer monitoring by major vendors. Indeed, ViCrypt targets the monitoring of video-streaming QoE from devices installed near the end-user, which do not necessarily belong to the ISP – but to the vendor – and where traffic load enables real-time monitoring with limited hardware. ViCrypt features are based on packet-level statistics and their computation does not require chunk-detection mechanisms, removing the extra overheads and errors introduced by such a detection step.

3 – Fine-Grained Estimation: ViCrypt tackles substantially more precise estimation tasks than previous work [3], [4]. In fact, ViCrypt estimates the six most common different video-resolution classes – 144p, 240p, 360p, 480p, 720p, and 1080p – instead of discriminating between low and high resolution. In addition, it provides a continuous estimation of the average video bitrate by relying on regression techniques.

4 – Extensive Machine-Learning-Model Benchmarking: Also unlike previous work, we devote a significant part of this study to benchmark different ML algorithms and evaluate their performance using different sets of inputs, carefully engineered by automatic feature-selection approaches to limit the number of required input data for proper execution.

5 – Empirical Validation Over a Heterogeneous YouTube Dataset: Last but not least, we show that ViCrypt performs accurately under a very heterogeneous set of scenarios. For this, we empirically tested the system over a large dataset of more than 15,000 streaming sessions of different YouTube videos. The dataset covers different access technologies (WiFi and LTE), transport protocols (QUIC and TCP), bandwidth configurations, players, and devices (browser player in laptops and native YouTube application in smartphones), and measurements were collected at four different ISPs in four different EU countries. This is an additional advantage over the state of the art, where proposals are generally validated using fewer or less representative scenarios.

This article is based on our preliminary work on ViCrypt [5]–[7]. Here, however, we provide not only a top-level view on the ViCrypt approach, but also present in detail different procedures for the extraction of features using constant memory. In contrast to our previous work, which focused mainly on a single ML model and a single KQI, this article presents an extensive benchmark of different ML models, targeting the estimation of all relevant KQIs for video streaming, including stalling, initial delay, video resolution, and bitrate. Moreover, we investigate the impact of different feature sets on the estimation of the targeted KQIs, and present results on the practical applicability of ViCrypt for real-time feature extraction and video-QoE prediction.

The remainder of the paper is organized as follows. In Section II, we describe related work on QoE of adaptive video streaming and QoE-based network monitoring approaches. In Section III, we present ViCrypt and introduce its basic concepts, detail the features and the extraction process, and describe the collected datasets used for performance evaluation. We benchmark the prediction performance of ViCrypt for all the proposed targets in Section IV. In Section V, we analyze the importance of different input feature sets, as well as their impact on inference results, and Section VI discusses practical considerations for real-time operation. Section VI additionally presents a reference performance comparison of the results achieved by ViCrypt to the results realized by two relevant competitors in the state of the art, as reported in their corresponding papers. While only useful for referencing purposes, such as comparison allows to position ViCrypt in the space of real-time video-QoE monitoring for encrypted network traffic. Finally, Section VII concludes this article.

II. Related Work

A. Video-Streaming QoE Context

In the past, video streaming mostly suffered from waiting times, namely stalling, caused by re-buffering events [8]–[10], and also from initial delay, i.e., the time until the start of the playback [11]. In the last years, these degradations have been partially mitigated by adapting the video bitrate to the network conditions, using HAS or Adaptive Bitrate (ABR) streaming technology. To operate HAS video streaming, the video content must be available in multiple bitrates, i.e., quality levels, and split into small segments or chunks, each containing a few seconds of playtime. The client-side adaptation logic requests the next chunk of the video in an appropriate bitrate, such that the initial delay is minimized, stalling is avoided, and the quality level is maximized to best utilize the available bandwidth. The decisions of the adaptation logic are typically based on the current bandwidth and/or buffer status [12], [13], but might take into account other aspects, such as client characteristics or fairness among competing clients [14]. The HAS streaming technology is adopted by a wide range of applications and video content providers, such as YouTube, Netflix, Amazon, and has been standardized as MPEG Dynamic Adaptive Streaming over HTTP (DASH) in ISO/IEC 23009-1 [15].

Changing the video bitrate also means modifying the visual quality of the streamed video, e.g., in terms of resolution, frame rate, or compression, which introduces an additional impact on QoE. An interesting finding on the impact of HAS on QoE [8], [16]–[19] is that, rather than quality changes, the most relevant effect to monitor is the fraction of total played time during which the video is played-out at a high

visual quality; the higher this time, the better the QoE. As a consequence, ISPs are highly interested in solutions able to estimate video resolution levels, which can serve to detect events when the played out quality level drops as soon as they happen, to take appropriate countermeasures. For example, thinking towards a more proactive network-management paradigm, ISPs would like to additionally estimate and predict the video bitrate to adjust the network configuration in time. This could include appropriately shaping the allocated bandwidth or selecting suitable routes for the streaming traffic, which would avoid further QoE degradation.

The trend towards end-to-end encryption (e.g., HTTPS) has significantly reduced the visibility of network operators on the traffic of their customers, making the monitoring process more challenging and cumbersome. It is no longer possible to rely on Deep Packet Inspection (DPI) to analyze the video data contained in each packet and reconstruct the streaming process and the video buffer [1], or to intercept and analyze segment requests. The encrypted stream of packets offers only very basic information about the streaming process, such as packet sizes and inter-arrival times, which has brought machine-learning (ML) based approaches to the center of the academic and industrial attention.

B. State of the Art

Previous studies on QoE for HTTP Adaptive Streaming (HAS) [8] confirm that stalling, initial delay, and quality adaptation are the most dominant QoE factors in HAS. Although adaptation is less severe than stalling [20], its impact on QoE should not be neglected. Indeed, each adaptation dimension (e.g., resolution, frame rate, quantization) has a specific impact on the perceived quality [8].

It has been shown that a quality switch implies a QoE degradation, and that the QoE changes according to the adaptation direction, even though switching down the video quality will have a stronger negative impact on video QoE [21]. The adaptation amplitude is the most dominant factor and a high amplitude leads to a low QoE, while low amplitudes might not be detectable [22]. Although a high frequency of quality adaptations will be annoying for end users [22], the actual quality changes have little impact on QoE. Only the resulting reduction of the time on high quality causes the QoE degradation [16], [17].

Due to the trend towards end-to-end encryption, DPI-based approaches are no longer effective. 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 applications [23]–[25], there are also general approaches for QoE analysis based on network-layer monitoring of encrypted video-streaming traffic. Authors in [3] evaluate ML-based architectures that estimate YouTube QoE using features derived from packet sizes, inter-arrival times, and throughput measurements. A similar approach is presented in [26], where authors rely on measurements in cellular networks to estimate 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. In that study, authors also use ML as a promising technique for large-scale quality monitoring and analysis. The authors of [27] estimate video-quality metrics (initial delay, stalling ratio, number of stallings, total stalling time) and user engagement for YouTube videos watched on smartphones, relying on ML and network-layer features. Reference [28] focuses on the reconstruction of buffered playtime at the video player side, as previously done in [2], but for encrypted traffic. This is leveraged to estimate video-QoE metrics in [29].

The two most similar approaches to ViCrypt are Requet [30], and the system presented in [4]. ViCrypt improves on both in multiple aspects: (*i*) while both approaches claim to be real-time, there is no evaluation of the computational costs required during the feature-extraction procedures, questioning their claims; in addition, for some of the targets, Requet has a temporal resolution based on chunk lengths (typically several seconds of a video), and [4] operates at a 10-second time scale, both significantly higher than for ViCrypt. This impacts their usability in practice for critical real-time monitoring applications, such as troubleshooting; (*ii*) while Requet also provides the same fine-grained classes for estimation of video resolution as ViCrypt, estimations in [4] are coarser-grained, with just few classes. Moreover, video bitrate, which is especially relevant for ISPs, is not inferred by their systems; (*iii*) while ViCrypt operates directly on the stream of packets at the network and transport layers, Requet requires chunk-detection mechanisms to extract chunk-based features, which is error-prone and introduces additional delays and complexities.

Other relevant differences between our study and the previous studies presented in [30] and [4] are reported in Table I; the table offers a comprehensive overview on the properties of the proposed solutions in terms of type and detail of the predicted KQIs, input features, monitoring capabilities, and datasets used for training and testing purposes.

III. ViCrypt Fundamentals

This section presents the fundamental concepts behind ViCrypt. As it has been shown in previous work [3]–[7], [26], [30], it is possible to extract features from the (encrypted) stream of packets which have strong correlation to different QoE-relevant metrics, such as re-buffering or video resolution, and to build ML models bridging the gap between network features and QoE metrics. This is also the approach followed by ViCrypt. Our system subdivides a video-streaming session into a sequence of time slots having a constant length. Throughout this work, we use a slot length of 1 s, which constitutes a good trade-off between estimation delay and accuracy. Nevertheless, the slot length is a system parameter, so the design principles of ViCrypt could also be applied to other temporal resolutions. At the design phase of ViCrypt, we tested similar temporal resolutions – up to 5 s – without resulting in significant changes in performance, yet loosing monitoring resolution. Being ML-driven, ViCrypt needs datasets containing both the collected traffic traces – the *input* – and the targeted KQI metrics – the

TABLE I
VICRYPT VS. REQUET AND INFOCOM'18 [4]. OVERVIEW ON THE PROPERTIES OF THE PROPOSED SOLUTIONS IN TERMS OF TYPE AND DETAIL OF THE PREDICTED KQIS, INPUT FEATURES, MONITORING CAPABILITIES, AND DATASETS USED FOR TRAINING AND TESTING PURPOSES

		ViCrypt	**Requet**	**INFOCOM'18** [4]
KQI estimation targets	Initial delay (# classes)	✓ (continuous)	✗	✓ (binary)
	Stalling (# classes)	✓ (binary detection/continuous estimation)	✓ (binary detection)	✓ (binary detection)
	Resolution (# classes)	✓ (6 levels, 144p–1080p)	✓ (6 levels, 144p–1080p)	✓ (binary, ≷ 480p)
	Bitrate (# classes)	✓ (continuous)	✗	✗
Input features	Chunk detection required?	✗	✓	✗
	# features	208	127	226
	Feature selection	✓ (down to 20 features)	✗	✗
Network monitoring	Real time	✓	✓	✓
	Temporal resolution	1 second	5 seconds or every chunk	5-10 seconds
	Feature computational efficiency	✓	not tested	not tested
Training/Evaluation data	Streaming service	YouTube	YouTube	YouTube
	# video sessions	15,000+	600	11,000
	Access network	WiFi & Cellular	WiFi	not mentioned
	# ISPs ǀ geo-location	4 ISPs ǀ 4 EU countries	2 ISPs ǀ US/India	1 ISP
	Devices	laptop & smartphone (native app)	laptop	laptop
	Time span	9 months in 2018/2019	6 months in 2018	4 months in 2017

ground truth. In the following, we explain how the datasets used in the study were generated and present a brief statistical analysis of the collected measurements. We then detail the lightweight feature-extraction procedures and describe the ML models used for benchmarking purposes.

A. YouTube Dataset Acquisition

Over a period of several months from June 2018 to February 2019, we streamed and recorded more than 15,000 YouTube video sessions, resulting in a total of more than 4,600,000 1-second time slots. For reference, Requet [30] collected measurements for only 580 video sessions, resulting in a dataset almost 26 times smaller. As we describe next, our dataset is not only large in terms of number of video sessions, but also very diverse.

For the video-streaming and data-collection tasks, we used a monitoring tool similar to [31]. It relies on the Selenium browser-automation library to automatically start a Chrome browser and randomly select a YouTube video to stream. Chrome was configured such that all HTTP requests were logged and QUIC traffic was enabled. A JavaScript monitoring script [32], [33] was injected into the Web page to record the current timestamp every 250 ms, as well as the current video playtime, buffered playtime, video resolution, and player state.

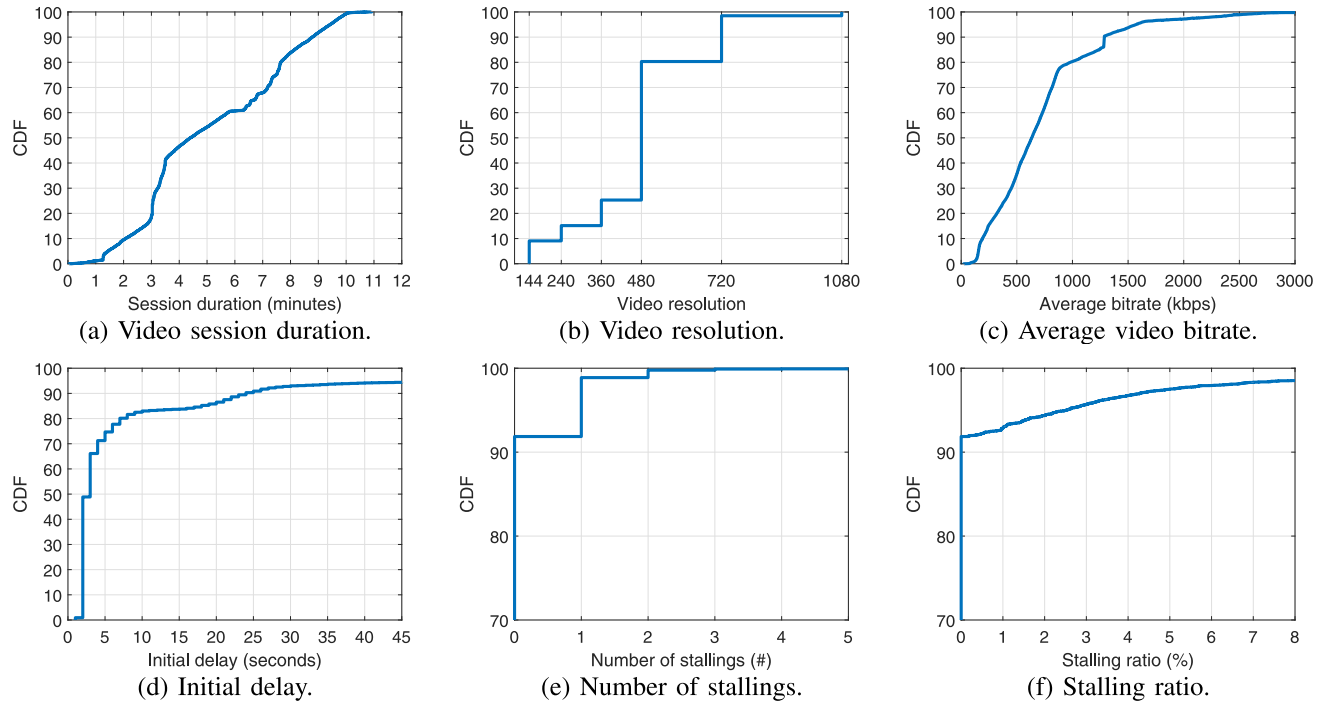
We streamed the video sessions with very diverse network setups to reach a highly generalizable model. Video sessions were collected at home (∼30% of the samples) and over corporate WiFi networks (∼50%), as well as over LTE mobile networks (∼20%). For some of the sessions, a firewall was enabled, which blocked all QUIC traffic, such that the videos were streamed via TCP (∼60%). 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. The number of video sessions per bandwidth configuration is mostly balanced among the different categories, with a slightly higher number of sessions streamed without bandwidth limitations. We collected video sessions from four different geographic locations – France, Austria, Germany, and Italy – and from four different ISPs.

Network traffic was collected for each video-streaming session, logging basic per-packet information (timestamp, source IP address, source port, destination IP address, destination port, size), as well as DNS-lookup responses to obtain a mapping between IP addresses and domain names. In each network-traffic trace, we identified YouTube video flows based on domain names (googlevideo.com), and extracted features only for these video flows, ignoring all non-YouTube traffic. Finally, we also included in our measurements the recently published YouTube open dataset [34], which includes measurements from the native, mobile Android YouTube application. While the share of app measurements is limited as compared to desktop devices (less than 10%), it contributes to the heterogeneity of the learning data.

Finally, we do not consider the challenging issue of video-traffic identification and filtering in this article, besides the aforementioned DNS-based identification approach. This is indeed a complex issue, especially when considering multiple users sharing the same IP address – e.g., NAT. The specific video-traffic identification and disentangling of concurrent video-streaming sessions is out of the scope of this article.

B. Dataset Analysis

Next, we provide some insight into the collected dataset. Figure 1 shows the characteristics of the dataset as cumulative distribution functions (CDF), with respect to duration of the video sessions, video resolution, average bitrate, initial delay, and stalling. Figure 1(a) shows that the recorded video sessions have durations between a couple of seconds and 11 minutes. The average length of a video session is approximately 5 minutes.



(a) Video session duration. (b) Video resolution. (c) Average video bitrate.

(d) Initial delay. (e) Number of stallings. (f) Stalling ratio.

Fig. 1. Characterization of the YouTube dataset, composed of more than 15,000 video-streaming sessions. There is a strong imbalance for some of the KQIs, such as occurrence of stalling events, or video resolution. Stalling is a rare event in the wild, generally traded by lower video resolutions in HAS, and this is reflected in the collected data. More than 50% of the video chunks were streamed on 480p resolution, and 1080p resolutions were rarely used.

In YouTube, the video resolution is typically indicated by the number of vertical pixels, for which standard quality classes exist. The video-resolution classes contained in the dataset can be easily observed in Figure 1(b), namely 144p, 240p, 360p, 480p, 720p, and 1080p. In some videos, the video resolution did not match exactly one of these classes, and therefore, was rounded to the nearest class. The distribution shows that the adaptation logic of YouTube decided to stream videos mostly in 480p resolution, but also very low resolutions occur (9% 144p, 6% 240p, 10% 360p). At the other end, HD resolution is rare (18% 720p, 1% 1080p). We did not observe any resolutions above 1080p during the measurements; therefore, although supported by YouTube, we kept resolution classes to the 6 observed in the dataset – the same 6 resolution levels were considered by Requet [30].

Figure 1(c) depicts the distribution of the average bitrate. Here, the average bitrate was obtained via the YouTube API, and represents the average bitrate of the full video when streamed with a given quality level (itag). Thus, this estimation target is not the momentary bitrate of the current slot, but rather the average bitrate of the quality level that was downloaded in the current slot. The average video bitrate spreads from approximately 20 kbps to about 4600 kbps. Nearly all of the slots have an average bitrate less than 3000 kbps. The CDF increases steeply, almost uniformly, until roughly 900 kbps, which corresponds to 78% of the slots. Then, it increases slower only showing a steep increase around 1280 kbps, which is the average bitrate for ∼4% of the slots, and thus, seems to be a certain target bitrate for encoding.

Figure 1(d) shows the distribution of the initial delays. Nearly 50% of the sessions have an initial delay of at most 2 seconds, and 75% of the sessions have a delay below 5 seconds. Figures 1(e) and (f) depict the distribution of two stalling metrics, namely the number of stallings per video session and the stalling ratio, i.e., the fraction of time spent in stalling mode with respect to the full session duration. Stalling events are rare: more than 90% of the videos do not stall at all, and when they do, the large majority stalls only once. The stalling ratio suggests that most of the stalling events are short with respect to the session duration: more than half of the observed stalling ratios are at most 3%.

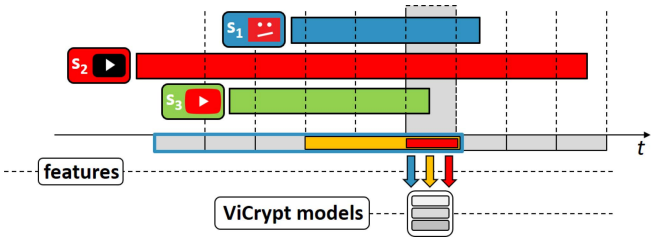


Fig. 2. ViCrypt overview. Features are continuously extracted/updated from the monitored video traffic, using different temporal aggregations of past time slots. At each new time slot, different ML models are applied to the input features, each one inferring the corresponding KQI.

C. ViCrypt Feature Extraction

Figure 2 presents a general overview on the functioning of ViCrypt. ViCrypt operates in a time-slotted, sequential manner, producing estimations for the selected KQIs at the end of each elapsed time slot. Features are extracted and continuously updated for each new packet on the stream of encrypted video traffic; at the end of each new time slot, different ML models infer the corresponding KQI from the extracted features.

ViCrypt embeds temporal notions in the construction of features, using information not only from the current time slot,

TABLE II
VICRYPT FEATURES. ALL FEATURES ARE DERIVED FROM THREE BASIC, PACKET-LEVEL METRICS, NAMELY PACKET COUNT, PACKET SIZE, AND INTER-ARRIVAL TIME (IAT), AGGREGATED AT TIME-SLOT-BASED AND WINDOW-BASED RESOLUTIONS

up/down/total	Volume	Throughput	Distribution	Protocol Shares
packets size	✓	✓	✓	✓
packet count	✓			✓
packet IAT		✓	✓	

but also from past slots. For memory and computational efficiency, the past streaming information must be compressed and structured. This is why ViCrypt keeps track of two additional macro windows, referred to as *trend* or short-term memory window, and *session* or long-term memory window. The trend window comprises the last t time slots in a sliding-window fashion, i.e., it contains all traffic of the current time slot and the $t-1$ most recent slots. For this article, ViCrypt uses a trend size of $t=3$, and thus, the features of the trend window of each time slot are computed from the traffic of the current slot and the previous two time slots. The value of t is set empirically on the evaluated datasets, but different from the time-slot length, the trend window length has a non-negligible impact on the model performance. The proposed value provides the best results in terms of inference performance for the analyzed data.

The second macro window is the session window, which includes all traffic of the session so far observed, and its features are therefore extracted from the traffic in all previous slots including the current time slot. All features of each current slot, trend window, and session window are computed in an online, stream fashion, without the need to store the previous traffic or detailed information about traffic packets observed in the past. This significantly reduces the memory consumption of the feature extraction process – from linear to constant, enabling a lightweight monitoring solution. Next, we dig deeper into the feature-extraction process, elaborating on the different computation steps.

Table II briefly summarizes the features computed per time slot. All features are derived from three basic, packet-level metrics, namely packet count, packet size, and inter-arrival time (IAT). Different aggregations are done on these metrics, based on individual time slots and aggregation windows – trend and session ones. Features are computed for uplink, downlink, and total traffic. The rationale behind the computed features is rather straightforward: adaptive video-streaming protocols employ closed-loop algorithms to achieve synchronization and adaptability between server and video player, thus traffic patterns on both uplink and downlink direction might reveal different behaviors at the player side.

Firstly, we compute simple count-based features from the traffic observed in the time slot. These consist of the number of total, uplink, and downlink packets, and the number of transferred bytes (total, uplink, downlink). We also count the number and byte volume of TCP and UDP packets, and compute the upload ratio, download ratio, TCP ratio, and UDP ratio from these counters, for both number of packets and number of bytes. Next, we extract time-based features. These include the 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 procedure is used to obtain a linear regression for the cumulative traffic over time, in an online fashion [35]. The Procedure 1 stores the origin of the regression (start time of the time slot $\texttt{slot}_{\texttt{start}}$), and keeps updates of the number of packets $\texttt{num}_{\texttt{P}}$ and the cumulative packet size $\texttt{cum}_{\texttt{size}}$, to compute the regression for the cumulative traffic. In addition, it stores and updates the current means of the abscissa (time, $\texttt{mean}_{\texttt{T}}$) and ordinate (cumulative packet size, $\texttt{mean}_{\texttt{S}}$), the number of packets $\texttt{n}$, as well as the temporal variance $\texttt{var}_{\texttt{T}}$ and covariance $\texttt{cov}_{\texttt{TS}}$. These are the only permanently stored variables, updated whenever a new packet of size $\texttt{s}$ arrives at time $\texttt{t}$.

Procedure 1 Online Regression Computation

procedure COMPUTEREGRESSION($\texttt{s}$, $\texttt{t}$)
- $\texttt{n} \leftarrow \texttt{n} + 1$
- $\texttt{cum}_{\texttt{size}} \leftarrow \texttt{cum}_{\texttt{size}} + \texttt{s}$
- $\texttt{diff}_{\texttt{slot}} \leftarrow \texttt{t} - \texttt{slot}_{\texttt{start}}$
- $\texttt{d}_{\texttt{t}} \leftarrow \texttt{diff}_{\texttt{slot}} - \texttt{mean}_{\texttt{T}}$
- $\texttt{d}_{\texttt{s}} \leftarrow \texttt{cum}_{\texttt{size}} - \texttt{mean}_{\texttt{S}}$
- $\texttt{var}_{\texttt{T}} \leftarrow \texttt{var}_{\texttt{T}} + \frac{\frac{\texttt{n}-1}{\texttt{n}} \cdot \texttt{d}_{\texttt{t}}^2 - \texttt{var}_{\texttt{T}}}{\texttt{n}}$
- $\texttt{cov}_{\texttt{TS}} \leftarrow \texttt{cov}_{\texttt{TS}} + \frac{\frac{\texttt{n}-1}{\texttt{n}} \cdot d_t \cdot d_s - \texttt{cov}_{\texttt{TS}}}{\texttt{n}}$
- $\texttt{mean}_{\texttt{T}} \leftarrow \texttt{mean}_{\texttt{T}} + \frac{\texttt{d}_{\texttt{t}}}{\texttt{n}}$
- $\texttt{mean}_{\texttt{S}} \leftarrow \texttt{mean}_{\texttt{S}} + \frac{\texttt{d}_{\texttt{s}}}{\texttt{n}}$
- $\texttt{slope} \leftarrow \frac{\texttt{cov}_{\texttt{TS}}}{\texttt{var}_{\texttt{T}}}$
- $\texttt{intercept} \leftarrow \texttt{mean}_{\texttt{S}} - \texttt{slope} \cdot \texttt{mean}_{\texttt{T}}$

The updates to these statistics only use three additional temporal variables: $\texttt{diff}_{\texttt{slot}}$, $\texttt{d}_{\texttt{t}}$, and $\texttt{d}_{\texttt{s}}$. At the end of the slot, we compute the final slope ($\texttt{slope}$) and intercept ($\texttt{intercept}$) values from the regression curve. We perform two regressions for uplink and downlink traffic, and the slope and intercept of these regression curves are added as features.

Finally, we extract multiple features derived from the empirical distribution of the traffic. We use an algorithm based on [36], 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. We extend this algorithm to additionally output the standard deviation, the coefficient of variation, as well as the minimal and the maximal values.

Here again, only few statistics are stored in memory and updated, namely the number of packets $\texttt{n}$, the mean value $\texttt{mean}$, the second, third, and fourth power of the sum of differences from the mean value ($\texttt{sdm}_2$, $\texttt{sdm}_3$, $\texttt{sdm}_4$), as well as the minimal ($\texttt{min}$) and the maximal ($\texttt{max}$) values. The update of these statistics occurs whenever a new value $\texttt{x}$ is observed for the corresponding statistic. The approach is explained in Procedure 2.

Procedure 2 Online Update of Distribution Metrics, Used for Computation of Distribution Features. The Procedure is Executed When New Values for the Corresponding Statistics are Observed

procedure UPDATEDISTRIBUTIONS(x)
  $\mathtt{n} \leftarrow \mathtt{n} + 1$
  $\mathtt{d_x} \leftarrow \mathtt{x} - \mathtt{mean}$
  $\mathtt{d_n} \leftarrow \frac{\mathtt{d_x}}{\mathtt{n}}$
  $\mathtt{mean} \leftarrow \mathtt{mean} + \mathtt{d_n}$
  $\mathtt{sdm_4} \leftarrow \mathtt{sdm_4} + \left[\mathtt{d_x}\,\mathtt{d_n}\,(\mathtt{n}-1)\,\mathtt{d_n^2}\left(\mathtt{n}^2 - 3\mathtt{n} + 3\right)\right] + \left(6\,\mathtt{d_n^2}\,\mathtt{sdm_2}\right) - \left(4\,\mathtt{d_n}\,\mathtt{sdm_3}\right)$
  $\mathtt{sdm_3} \leftarrow \mathtt{sdm_3} + \left[\mathtt{d_x}\,\mathtt{d_n}\,(\mathtt{n}-1)\,\mathtt{d_n}\,(\mathtt{n}-2)\right] - \left(3\,\mathtt{d_n}\,\mathtt{sdm_2}\right)$
  $\mathtt{sdm_2} \leftarrow \mathtt{sdm_2} + \left[\mathtt{d_x}\,\mathtt{d_n}\,(\mathtt{n}-1)\right]$
  if $\mathtt{x} < \mathtt{min}$ **then**
    $\mathtt{min} \leftarrow \mathtt{x}$
  if $\mathtt{x} > \mathtt{max}$ **then**
    $\mathtt{max} \leftarrow \mathtt{x}$

The computed updates allow to directly obtain the mean (mean), minimal (min), and maximal (max) values. Moreover, the variance (var), standard deviation (std), coefficient of variation (cvar), skewness (skew), and kurtosis (kurt) of the distributions can be computed as stated in Procedure 3. These distribution-based features are computed for the packet size and the IAT, for both uplink and downlink traffic.

This results in a total of 69 basic features for the traffic in a time slot, plus the same 69 basic features for each of the two additionally considered macro windows, namely the trend and the session windows. Together with the sequence number of the current time slot, which is also included as a feature, this sums up to a total of 208 features, characterizing each slot of 1 s length. 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 and aggregate the traffic of the current time slot. In contrast, only a single session window is needed, as it only needs to accumulate the full traffic of the complete 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.

D. ML Models Benchmarking

Solutions so far proposed in the state of the art such as [3]–[7], [26], [30] rely mostly on random-forest models as the underlying ML approach. In this article, we provide further insights on the performance of different types of models, benchmarking 11 different ML models within ViCrypt. Nine out of these 11 models are fit for both classification and regression tasks, while the other two are designed for anomaly detection, hence our selection.

Procedure 3 Computation of Distribution Features

procedure COMPUTEDISTRIBUTIONFEATURES
  $\mathtt{var} \leftarrow \frac{\mathtt{sdm_2}}{\mathtt{n}-1}$
  $\mathtt{std} \leftarrow \sqrt{\mathtt{var}}$
  $\mathtt{cvar} \leftarrow \frac{\mathtt{std}}{\mathtt{mean}}$
  $\mathtt{skew} \leftarrow \sqrt{\frac{\mathtt{n}}{\mathtt{sdm_2^3}}} \cdot \mathtt{sdm_3}$
  $\mathtt{kurt} \leftarrow \mathtt{n} \cdot \frac{\mathtt{sdm_4}}{\mathtt{sdm_2^2}} - 3$

Most of the selected models also rely on decision trees, not only because of their proven high accuracy and low computational cost, but also due to a series of embedded properties, such as model visibility, robustness to input noise, and embedded feature selection. We consider both bagging and boosting ensembles based on trees, which brings robustness, increased accuracy, and improved generalization of the training. To increase training speed, reduce model complexity, and therefore reduce the chances of over-fitting, we favor small-sized ensembles, using 10 to 50 models.

Model parameters are calibrated through standard grid-search optimization. Finally, **all evaluations throughout the paper are done through 5-fold cross-validation**. For the classification tasks, we apply stratified cross-validation, i.e., we ensure that the five folds preserve the percentage of samples for each class. The list of benchmarked models includes **Decision trees** (DT), **Random Forest** with 10 trees (RF10), **AdaBoost** using 50 trees (ADA), **ensembles with 10 extremely randomized trees** [37] (ERT10), **bagging** with 10 trees (BAGGING), **naïve Bayes** (BAYES), ***k*-nearest neighbors** with $k = 5$ (KNN), **Neural Networks** (NN) (three hidden layers, the first one containing 200 neurons, the second one 100 neurons, and the last one 50 neurons, using sigmoid activation and softmax at the output), and **support vector machines** (SVM) - a regression version also exists, which is called support vector regression (SVR). As stalling can be considered as an anomaly of the streaming process, we additionally evaluate two anomaly-detection algorithms for stalling *detection*:

Isolation Forests With 10 Trees [38] (ISO10): An unsupervised model which behaves similarly to ERT10, but at each node, both the feature and the cut point are chosen randomly. The number of nodes a sample needs to traverse to reach a leaf is the normality measure, such that the fewer nodes a sample has to visit, the more abnormal it is. This is intuitive, as outliers have unusual characteristics, and thus, are rapidly distinguishable from the normal samples.

Local Outlier Factor [39] (LOF): An algorithm relying on the concept of local density. The local density of a sample is estimated from the distance to its *k* nearest neighbors (in our case, we set *k* to 20). The algorithm compares the local densities of the given sample and its *k* nearest neighbors. If a sample has a much lower density than its neighbors, it is considered as an outlier.

When considering DT, RF10, and ERT10 models, and to counterbalance the impact of imbalanced classes, we assign weights to each sample *i* of $class_i$ based on the occurrence

frequency of its class:

$$\mathtt{W_i} = \frac{\#\ \mathtt{samples}}{\#\ \mathtt{classes} \times (\#\ \mathtt{samples\ in\ class_i})}$$

This implies that the estimation errors for samples of rare or under-represented classes are significantly penalized, which improves the estimation accuracy for these classes. We further exploit the fact that RF10, ERT10, BAGGING, and KNN models can be parallelized for speed improvement, and run them in a parallelized fashion. For NN, we use TensorFlow on GPU, while we use the scikit-learn library for the remaining models. The benchmark of the models is executed on a high-end desktop computer, equipped with two Intel Xeon Silver 4116 processors including 12 physical cores each (a total of 48 virtual cores thanks to Intel HyperThreading), 128 GB of RAM, and a NVIDIA GeForce RTX 2080 Ti graphics card (with 11 GB of VRAM).

IV. ViCrypt in Action—Performance Evaluation

We now present the performance evaluation results of ViCrypt for all the described KQIs. We take the full set of 208 features as input, i.e., we do not explicitly consider feature-selection approaches. We devote Section V to feature selection. For each of the tested ML algorithms, we report both performance metrics, as well as the total running time for training and inference, i.e., the time needed to compute the 5-fold cross-validation on the whole dataset. This helps to better understand the practical trade-offs when it comes to real-time analysis.

A. Stalling Estimation

As stalling is the most severe QoE degradation, our first goal is to estimate whether the video is stalling or not. More precisely, for each 1-second time slot, ViCrypt infers whether the video is being played or stalling; this is therefore a binary classification problem. We consider only time slots which contain network traffic, and end up with almost 1,283,000 samples.

The binary stalling-estimation results obtained for each of the time slots can be further combined to obtain stalling metrics at a video-session level, such as the initial playback delay, the number of stalling events, and the stalling ratio – ratio of total stalling time to total playback time. The initial delay is given by the number of slots predicted as stalling at the start of the session. After the initial delay, a stalling event is counted only if two or more consecutive time slots are predicted as stalling, making the aggregated stalling metrics more robust against isolated false predictions. In this case, the number of consecutive slots with stalling is added to the total stalling time in seconds. Thus, by simple count 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. The granularity of the initial delay and stalling time estimation is limited by the time-slot length, one second. Nevertheless, such a fine-grained resolution is sufficient for most monitoring use cases.

We therefore present evaluation results for stalling at two different temporal granularities: per-slot, binary stalling classification (stalling/no stalling), and per-session, continuous estimation of initial delay, number of stalling events, and stalling ratio.

Table III summarizes the overall accuracy, recall, and precision for the stalling class per model, as well as the overall cross-validation times. Results show that stalling detection is a challenging task, especially due to the high imbalance of the data (see Figure 1(e)). Indeed, let us take the NN model as example: the trained model was not able to identify a single stalling slot, still achieving a high accuracy of 94.3%, due to the imbalance. This recalls that the overall accuracy can be misleading with highly imbalanced data, and that it is particularly important to look in detail at recall and precision results. Here, we observe that the tree-based models achieve a high precision of around 90%, but only a recall of around 60%. BAGGING is an exception, with a recall of 65%.

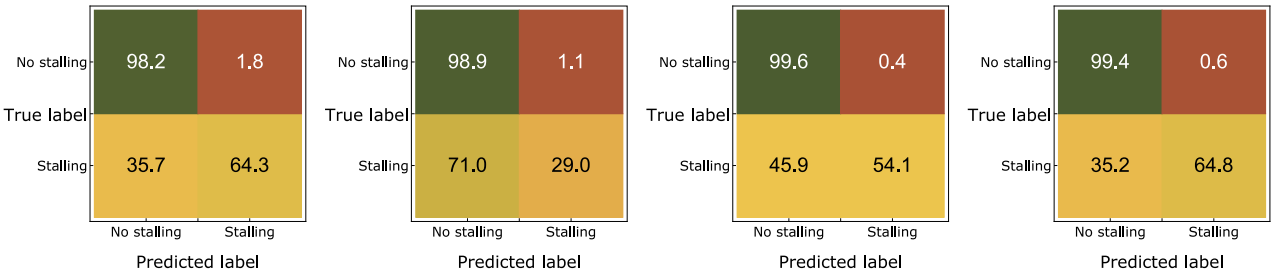
To dig deeper into these results, Figure 3 presents the confusion matrices for the different models. We left out the confusion matrix of the neural network as discussed above, and the ones for RF10 and LOF, as the matrices were nearly identical to those of ERT10 and ISO10, respectively. The confusion matrices underline that stalling detection is a rather difficult task. Surprisingly, BAYES is the algorithm yielding the highest stalling-class accuracy (i.e., recall), achieving 86%. However, its very poor precision turns the approach inapplicable. The outlier-detection algorithms (ISO10 and LOF) also perform poorly for this estimation task, which might indicate that the features do not deviate much between the stalling and no-stalling classes.

ERT10 seems to be a good model choice for stalling detection: it runs very quickly and realizes a decent recall-precision combination. Only BAGGING achieves an overall better performance, especially a higher recall, but at the cost of a much higher cross-validation processing time of roughly one hour versus one minute for ERT10. Of course, as training is usually done offline, this would in principle not be a limitation for BAGGING. However, if one would consider adaptive learning approaches, or applying the model in scenarios with strong video traffic variations, low training times become paramount.

We now explore the inference performance of ViCrypt for stalling metrics at the session level. More precisely, we estimate the initial playback delay of the videos, the number of stalling events, and the stalling ratio. Figure 4 shows the distribution of estimation errors for the three considered targets. Regarding initial delay, ViCrypt perfectly infers the real playback delay for about 40% of the video sessions, and achieves an error of at most 2 seconds for 70% of the sessions. The number of stallings is perfectly estimated for about 50% of the sessions, and an error of at most 2 stallings is realized for about 75% of the sessions. The stalling ratio is perfectly estimated for about 60% of the sessions, and the error is below 3% for more than 85% of the sessions. Errors related to stalling estimation usually correspond to overestimations, which is always preferred, for the sake of safety margins and over-provisioning. Also, stalling ratio has been the preferred metric in the state of

TABLE III
BENCHMARKING OF ML MODELS FOR THE STALLING DETECTION—OVERALL ACCURACY, RECALL/PRECISION ONLY INDICATED FOR THE STALLING CLASS

	Accuracy (%)	Recall (%)	Precision (%)	5-CV time (minutes)
DT	96	64	68	57
RF10	97	55	88	3
ADA	95	29	61	154
ERT10	97	54	88	1
BAGGING	97	65	87	63
BAYES	50	86	9	1
KNN	96	48	71	10
NN	94	0	0	600
SVM	84	62	21	36
ISO10	86	13	8	4
LOF	86	11	6	46

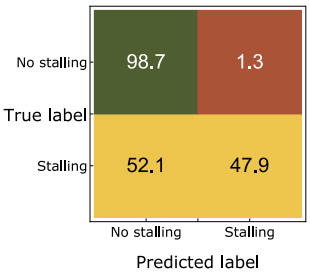


(a) DT confusion matrix.

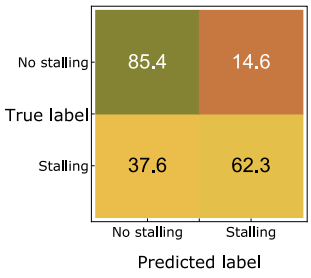
(b) ADA confusion matrix.

(c) ERT10 confusion matrix.

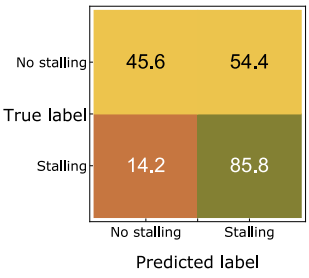
(d) BAGGING confusion matrix.



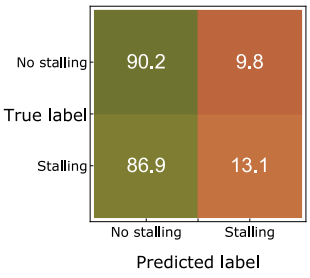
(e) KNN confusion matrix.



(f) SVM confusion matrix.

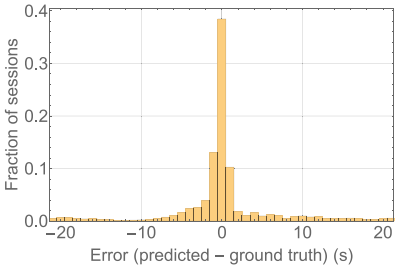


(g) BAYES confusion matrix.

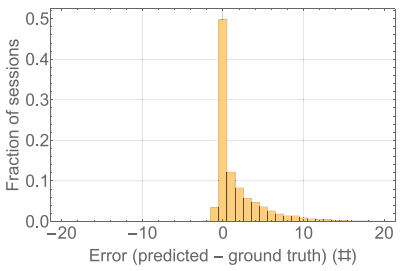


(h) ISO10 confusion matrix.

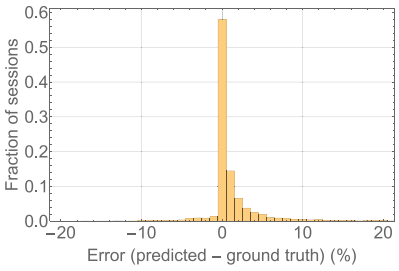
Fig. 3. Normalized confusion matrices obtained by the benchmarked ML models for the estimation of stalling.



(a) Initial delay.



(b) Number of stallings.



(c) Stalling ratio.

Fig. 4. Prediction performance for session-based stalling metrics, using ERT10 as the underlying model. Initial delay, number of stalling events, and stalling ratio are perfectly estimated for about 40%, 50%, and 60% of the video sessions, respectively.

the art when it comes to session-based stalling estimation [3], [26], and achieved results are in line with or even better than the state of the art [3], [26], even when dealing with such a strong imbalance in the data.

Finally, for visualization purposes, Figure 5 shows the real-time stalling estimation produced by ViCrypt for an exemplary YouTube video-streaming session, using the ERT10 as underlying model. ViCrypt can track in real time the overall stalling pattern of the video session, from the initial playback delay to the occurrence of stalling events.

B. Video-Resolution Estimation

The video resolution is highly linked to the visual quality of the streamed video, and is therefore a crucial QoE metric. The estimation of the resolution is treated as a

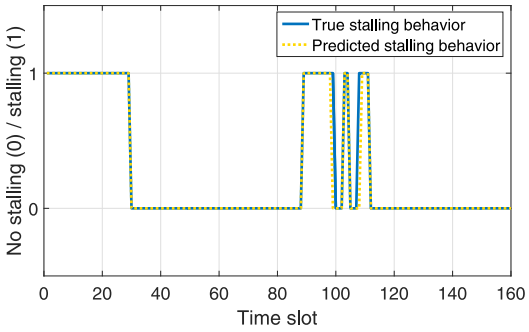


Fig. 5. Example of ViCrypt real-time stalling detection.

TABLE IV
BENCHMARKING OF DIFFERENT ML MODELS FOR THE RESOLUTION ESTIMATION

	Accuracy (%)	5-CV time (minutes)
DT	92	43
RF10	92	2
ADA	68	125
ERT10	90	1
BAGGING	95	37
BAYES	42	1
KNN	73	9
NN	58	507
SVM	54	194

multi-class classification problem. The considered classes correspond to the typical YouTube video resolutions: 144p, 240p, 360p, 480p, 720p, and 1080p. Thus, the classification problem is based on six classes, which is substantially more precise than other approaches, e.g., [3], [4], [26]. After considering only time slots with a valid resolution and containing traffic, we end up with a dataset including almost 1,160,000 time slots.

We report the accuracy achieved by the different models and the corresponding total processing times for cross-validation in Table IV. Except for AdaBoost, all the tree-based methods provide very high overall accuracy, above 90%. KNN also achieves encouraging results, with an accuracy of 73%. The accuracy of BAYES is by far the worst, which is most probably due to its underlying hypothesis that the different features are independent from each other, which does not seem to be satisfied for the video resolution. NN and SVM also yield disappointing results, especially when considering that they needed significantly more time than the other models. Here, we can also verify the benefit of parallelization: besides BAYES, the fastest algorithms are the parallelizable ones, which is a non-negligible advantage for these models. For instance, RF10 and ERT10 were done in at most two minutes, while ADA, NN, and SVM took several hours.

As the video-resolution classes are also strongly imbalanced (see Figure 1(b)), we take a closer look at the per-class accuracy (i.e., recall) and precision. Results are depicted in Figures 6 and 7. The recall indicates the percentage of time slots of a given quality for which ViCrypt correctly inferred the resolution. In contrast, the precision for a given quality expresses the proportion of the class estimations which are correct. We left BAYES out of this analysis because of its poor performance.

Figure 6 reveals that the 480p class is accurately detected by all of the eight models, with SVM being the worst with an accuracy below 70%. DT, RF10, ERT10, and BAGGING achieve a near perfect score for this video resolution. This comes as no real surprise, as more than 50% of the time slots have a resolution of 480p. However, it is interesting to note that most models accurately estimate the 144p class, even though it is a significantly underrepresented class with only 9% of the slots having that resolution. For all the models, these two classes are the ones that are the most accurately detected. For instance, NN obtained an accuracy of more than 60% for 144p and more than 80% for 480p, while for the other classes its accuracy is below 30%. For a couple of models, and especially for ADA and NN, it was challenging to accurately classify the 240p and 360p resolutions; for NN, the accuracy for 360p is even close to 0%. In case of ADA and NN, the two classes were very frequently detected as either 144p or 480p.

Figure 7 shows that the precision of the benchmarked models is similar to the recall: it is highest for DT, RF10, ERT10, and BAGGING (always higher than 80%), while it is relatively low for ADA, NN, and SVM. Contrary to the recall, the precision is not systematically high for the 144p and 480p classes. Overall, the per-class analysis gives us interesting insights into the performance of the models and indicates that only DT, RF10, ERT10, and BAGGING provide consistently excellent estimation throughout all the six video-resolution classes. For example, even though the total accuracy of KNN is decent, it is mostly due to its performance for the 144p and 480p classes.

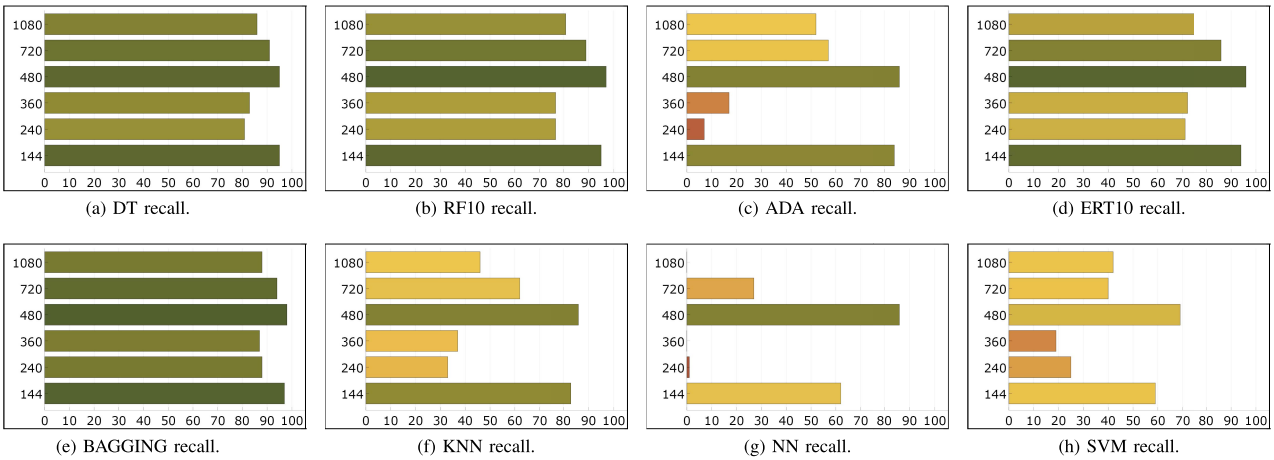
Results suggest that RF10 is the most appropriate model for the video resolution estimation task: this model is extremely lightweight, executes fast, and presents an excellent performance, with a recall and precision close to or above 80% for each resolution class. Similar to the stalling-inference results, BAGGING is the best model in the benchmark, with recall and precision close to or above 90% for all resolution classes, but using a more complex underlying structure, as reflected by the cross-validation execution times.

Again, for visualization purposes, Figure 8 shows the real-time estimation and tracking of the video resolution produced by ViCrypt for an exemplary YouTube video-streaming session using multiple resolution levels (720p, 360p, 480p, and 144p), using the RF10 as underlying model.

C. Average-Bitrate Estimation

The last estimation target is the average video (encoding) bitrate, which is highly relevant for proactive network management. ViCrypt infers the average video bitrate of the video contents monitored at each 1-second time slot. As the bitrate is per-se continuous, the estimation of the average bitrate is tackled as a regression task. Again, we consider only those time slots with actual traffic and a valid average bitrate label, obtained from the YouTube API. The resulting dataset consists of more than 933,000 samples.

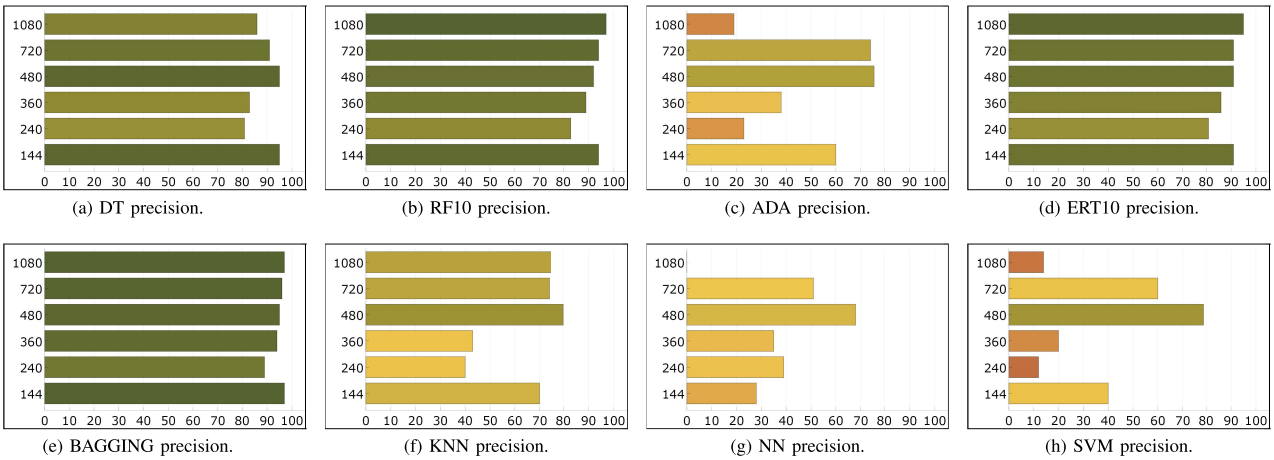
We benchmark the same models as before, using 5-fold cross-validation. The only exception is the Naïve Bayes model, which can only handle classification tasks; we thus replace it



(a) DT recall. (b) RF10 recall. (c) ADA recall. (d) ERT10 recall.

(e) BAGGING recall. (f) KNN recall. (g) NN recall. (h) SVM recall.

Fig. 6. Accuracy per class (i.e., recall) obtained by the benchmarked ML models for the resolution estimation.



(a) DT precision. (b) RF10 precision. (c) ADA precision. (d) ERT10 precision.

(e) BAGGING precision. (f) KNN precision. (g) NN precision. (h) SVM precision.

Fig. 7. Precision per class obtained by the benchmarked ML models for the resolution estimation.

by the Bayesian ridge regression (BAYES). For each model, we report the mean absolute error (MAE) $= \texttt{mean}(|\hat{X} - X|)$, the root mean squared error (RMSE), the mean relative error MRE $= \texttt{mean}(|\hat{X} - X|/X)$, and the Pearson linear correlation coefficient (PLCC), where X and $\hat{X}$ are the real and inferred values, respectively. As before, we also report the total processing times for the 5-fold cross validation. While the MAE metric penalizes all the errors equally, the RMSE puts a relatively high weight on larger errors. A PLCC value close to 1 indicates that the real and estimated values are strongly positively correlated, a negative value shows a negative correlation, and a PLCC close to 0 indicates that there is no linear correlation.

Results are summarized in Table V. We note that the models that worked well for the video-resolution estimation, namely DT, RF10, BAGGING, and ERT10, perform also very well for the inference of the average bitrate. Indeed, these four tree-based models yield the lowest errors, achieving a MAE of below 100 kbps, and very high PLCCs close to 1. RMSE and MRE values are relatively low for these algorithms, suggesting that they only rarely make large errors. However, it is interesting to see that RF10 needs significantly more time to process the whole dataset than for the video-resolution estimation. As for the video-resolution inference, BAYES and especially SVM provide disappointing results. With BAYES, ViCrypt obtained a negative PLCC as well as very high error metrics, which underlines the bad performance of the model. With SVM, the system output errors of an unacceptable order of magnitude.

Figure 9 depicts the distributions of the inference errors for the different regression models. SVM errors are not reported, as they are simply too large. Overall, the CDFs confirm our observations from Table V. The most promising tree-based methods present errors very close to 0 for a non-negligible fraction of the dataset; this is especially true for DT, which realizes an almost perfect estimation for 60% of the samples. However, DT presents a large RMSE compared to its tree-based competitors, indicating that it yields larger errors than the other tree algorithms. ADA is the only tree-based method where estimation errors are most often quite high, higher than 500 kbps for about 45% of the time slots. Absolute errors are below 100 kbps for approximately 80% of the time slots when using DT, RF10, BAGGING, or ERT10 as underlying models.

Based on these results, and again considering the outperformance in terms of computational times, ERT10 seems to be the best algorithm for the estimation of the average bitrate with ViCrypt. Even though error metrics are slightly worse for ERT10 than those for RF10 or BAGGING, differences

TABLE V
BENCHMARKING OF DIFFERENT ML MODELS FOR THE ESTIMATION OF AVERAGE BITRATE

	MAE (kbps)	RMSE (kbps)	MRE (%)	PLCC	5-CV time (minutes)
DT	94	246	18	0.88	31
RF10	89	179	18	0.93	36
ADA	492	573	130	0.59	126
ERT10	93	182	19	0.93	7
BAGGING	89	179	17	0.93	22
BAYES	2,540	6,530	545	-0.14	3
KNN	229	353	42	0.70	6
NN	333	489	70	0.20	305
SVM	10^{23}	$2 \cdot 10^{23}$	$2 \cdot 10^{23}$	0.12	143

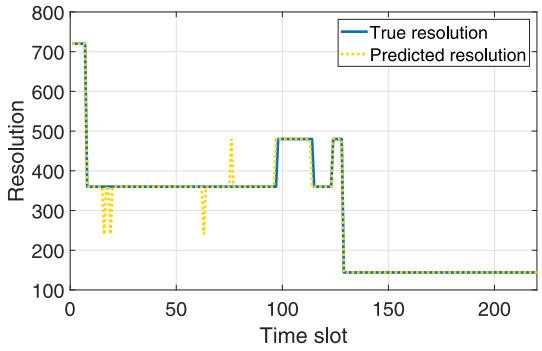


Fig. 8. Example of ViCrypt real-time video-resolution estimation.

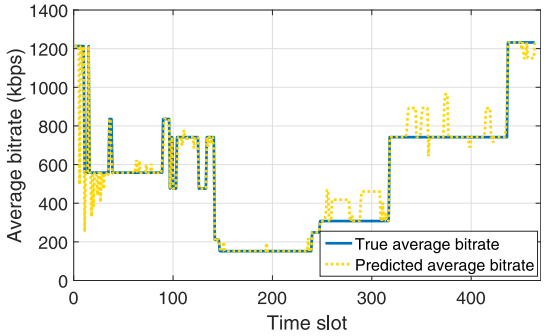


Fig. 10. Example of ViCrypt real-time average video-bitrate estimation.

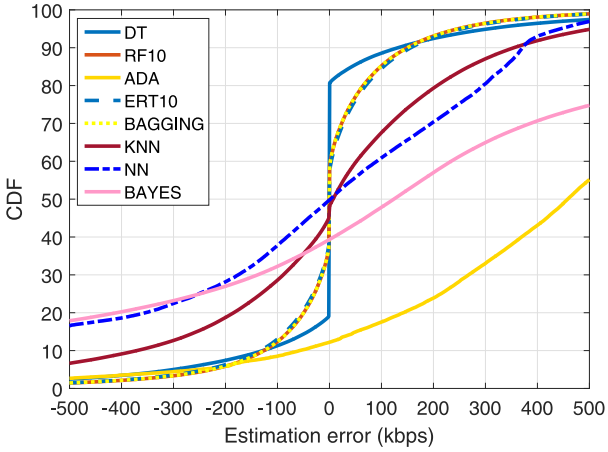


Fig. 9. Errors (estimated value—true value) obtained by the benchmarked ML models for the average video-bitrate inference.

are not significant enough and lightweight models should be preferred.

Finally, Figure 10 shows ViCrypt's estimation of the average bitrate for an exemplary video with several bitrate changes. Again, ViCrypt estimates the average bitrate with high precision throughout the whole video. Rather than estimating a too low bitrate, ViCrypt coupled with ERT10 overestimates the ground truth in more than half of the time slots (54%). This overestimation of ERT10 together with the generally low estimation error is advantageous from the point of view of the ISP, as overestimating the video bitrate helps them to avoid allocating insufficient bandwidth in the context of traffic shaping. This could cause the video to stall, which is a major QoE degradation. This behavior could be even forced by adding a safety margin to the estimations of ViCrypt.

V. FEATURE-IMPORTANCE ANALYSIS

Results presented so far correspond to ViCrypt models using the full set of 208 features as input for the estimations. In this section, we analyze the importance of different feature sets and their impact on inference performance. Using an extensive list of input features is not always the best strategy, as it may negatively impact estimation performance. Using more features increases the dimensionality of the feature space, introducing sparsity issues. In addition, using irrelevant or redundant features may lower model performance in practice. Last but not least, working in higher-dimensional spaces usually results in higher computational times.

We resort to standard automatic feature-selection techniques to identify the most relevant input features for our three estimation targets. Moreover, we consider additional feature subsets which might have a significant impact, considering for example the difference between snapshot features – i.e., those computed for the same slot where the estimation takes place – and trend- or session-based features. Based on these guidelines, we divide the full input-feature set into the following six feature subsets:

(1) F_C *subset:* the features representing the current time slot, i.e., the time slot for which we want to infer the video resolution (69 features).

(2) F_T *subset:* the features collected for the trend window (69 features).

(3) F_S *subset:* the features summarizing the characteristics of the session since the beginning of the streaming (69 features).

(4) F_{DOWN} *subset:* the features related to the download traffic (81 features).

(5) F_{UP} *subset:* the features representing the upload traffic (81 features).

TABLE VI
TOP-5 MOST IMPORTANT FEATURES FOR THE THREE ESTIMATION TARGETS TACKLED BY VICRYPT, WITH THEIR CORRESPONDING WINDOW (CURRENT, TREND, OR SESSION) AND GINI IMPORTANCE SCORES

	Stalling	**Video resolution**	**Average bitrate**
#1 feature	maximum upload packet size (trend) [0.03]	throughput (session) [0.04]	throughput (session) [0.07]
#2 feature	standard deviation of upload packet size (session) [0.02]	burst throughput (session) [0.03]	burst throughput (session) [0.05]
#3 feature	upload volume (session) [0.02]	mean IAT of download packets (session) [0.02]	skewness of upload packet-size distribution (session) [0.04]
#4 feature	standard deviation of download packet size (session) [0.02]	burst throughput of download traffic (session) [0.02]	mean IAT of download packets (session) [0.04]
#5 feature	skewness of upload packet-size distribution (session) [0.01]	coefficient of variation of IAT of download packets (session) [0.02]	download burst throughput (session) [0.04]

(6) F_{TOP20} *subset:* the 20 most important features, determined using automatic feature-selection techniques (20 features).

To select the 20 most relevant features (F_{TOP20}), we take the best-performing ML algorithm, which is always a tree-based method, and apply an embedded feature-selection technique, i.e., an approach ranking the features based on their importance for the algorithm: for each run of the 5-fold stratified cross-validation, we fit the model on the training folds and detect the 20 most discriminative features with the Gini importance measure [40]. In a tree, this measure is defined as the weighted sum of the impurity reduction at each node of the tree testing feature *f*. For a forest, the resulting importance of f is the average over all trees. Then, we re-train the model only on those 20 features and test its performance on the test fold. We determine the overall top 20 features based on their importance score averaged over the five folds, as well as the average accuracy of the algorithm over the folds with only the selected features.

Before going into the specific performance results achieved with these subsets, let us take a look at the five most relevant features according to the aforementioned feature-selection approach. Table VI reports the five most important features for the three investigated KQIs. For each of the selected features, we additionally report the corresponding temporal window (current, trend, or session) and the importance score. For stalling detection, the most important features come from different subsets. The most important feature for stalling inference comes from the F_T subset (trend window). Nevertheless, almost all of the most important features actually come from the set of session-based features. F_S can be generally considered as the most relevant feature set for stalling estimation, which is in line with our previous results in [6], using a different feature importance metric, namely, the information gain. For video resolution, the top 5 features are all session-related and include statistics about throughput patterns and information related to the inter-arrival times between packets. The results for the average bitrate confirm that session-based features are often relevant ones, followed by the features of the trend window and the current time slot.

As we show next in the specific comparison results, session-related features – the F_S subset – are the most important ones, and have the most discriminative performance, followed by trend features – F_T. This is coherent with the overall nature of adaptive-video-streaming algorithms, where stronger variations tend to occur at the beginning of the video session, and conditions tend to remain constant over the course of the streaming – as long as the connection remains stable. Features computed for the current time slot generally achieve the lowest importance scores. This suggests that snapshot-like approaches as the one proposed in [4] are less powerful and more prone to over-fitting, and probably have poorer generalization capabilities.

TABLE VII
VICRYPT PERFORMANCE FOR STALLING ESTIMATION WITH ERT10, USING DIFFERENT FEATURE SUBSETS (OVERALL ACCURACY AND RECALL/PRECISION ONLY INDICATED FOR THE STALLING CLASS)

Features	**Accuracy (%)**	**Recall (%)**	**Precision (%)**
All	97	54	88
F_C	96	10	30
F_T	97	17	51
F_S	99	72	91
F_{DOWN}	98	41	87
F_{UP}	98	47	74
F_{TOP20}	97	56	86

A. Stalling

Table VII reports the estimation performance for stalling in terms of recall and precision for the stalling class, using ERT10 as the underlying model, which is the one we selected in Section IV-A for this task. As before, results correspond to 5-fold stratified cross-validation. Stalling-detection results when using only F_C and F_T subsets are poor for both recall and precision. However, performance dramatically improves when considering F_S features only; indeed, the recall for ERT10 increases from 54% (using all features, see Table III) to 72%, and even the precision increases from 88% to 91%, taking the overall accuracy to 99%. This confirms the paramount importance of session-progression features, which is in line with our above findings and discussion.

When relying exclusively on the 20 most important features, ViCrypt obtains a recall score of 56% and a precision score of 86%, almost on a par with using all features. This tells us the following: (*i*) the overall statistics describing the entire history of the session are very insightful metrics for detecting stalling; (*ii*) ViCrypt produces highly accurate estimations even with a reduced set of features: instead of using 208 attributes, 69 or even 20 would be sufficient. This shows that a substantial number of features can be removed with only a minor performance

TABLE VIII
VICRYPT PERFORMANCE FOR INFERRING THE VIDEO RESOLUTION WITH RF10, USING DIFFERENT FEATURE SUBSETS

Features	Accuracy (%)
All	92
F_C	70
F_T	73
F_S	96
F_{DOWN}	90
F_{UP}	90
F_{TOP20}	95

degradation, which even increases the practical applicability of ViCrypt.

B. Video Resolution

For the case of video-resolution inference, we study the performance of RF10, the model that produces the most promising outcome, based on the different feature groups. The results in Table VIII also reveal that the features of F_C and F_T yield the poorest results in terms of accuracy, indicating that they are insufficient to infer the video resolution with high precision. Indeed, their accuracy is more than 15 percentage points below the accuracy obtained by ViCrypt based on the entire feature set. However, ViCrypt performs very well when used with either F_S, F_{DOWN}, or F_{UP}, with F_S performing even better than the entire feature set. The average accuracy of ViCrypt when using only the top 20 features is highly encouraging: it is equal to 95%, confirming the above finding that a substantial number of features can be removed.

C. Average Bitrate

Table IX reports the results obtained for the estimation of the average bitrate, using ERT10 as underlying model. The differences in terms of performance between the considered subsets and the whole set of features are much more significant than in the video-resolution case. As a matter of fact, with F_C and F_T, the value of the MAE is nearly three times higher than when relying on the whole feature set; the other error metrics underline the poor performance. In case ViCrypt bases itself on the download- or upload-traffic information, the obtained errors are only slightly higher than when inferring from all the features. However, as for the stalling detection and video-resolution inference, ViCrypt yields excellent results when coupled with F_S features only, showing once again that information about the session history is the most valuable one. Again, using only the top 20 selected features for the ERT10 model yields slightly more precise estimations than when using the whole feature set. Indeed, the MAE and the RMSE decrease to 81 kbps and 175 kbps, respectively.

VI. PRACTICAL CONSIDERATIONS FOR REAL-TIME OPERATION & DISCUSSION

Finally, we elaborate on the presented results and discuss the applicability of ViCrypt in practice. To ensure that ViCrypt is scalable and can be deployed in the wild, we analyze several key aspects in terms of computation time, and execute multiple tests on two machines with completely different

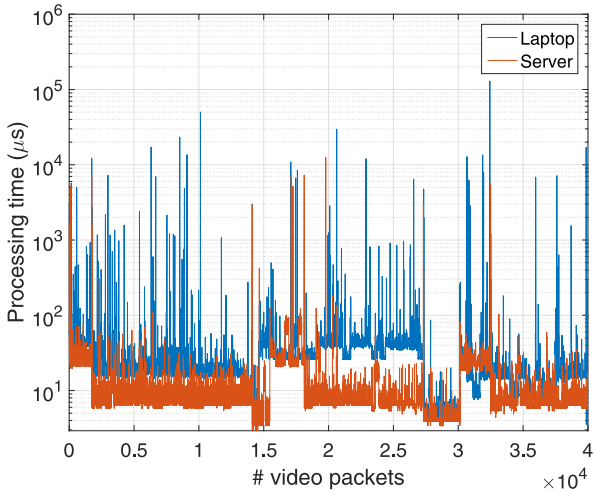


Fig. 11. Time Needed to Update the ViCrypt Features Each Time a New Packet Arrives (Log Scale).

technical specifications: on *server*, the high-end computer presented in Section III, and *laptop*, which includes a Intel Core i5-4200U CPU with two physical cores and a total of four virtual ones, eight gigabytes of RAM, and an integrated GPU Intel HD Graphics 4400. We show that ViCrypt runs extremely fast, with minimal memory footprint. The evaluation is not done at scale, but considering the end-to-end processing of single video sessions. Still, the main target of ViCrypt is video-streaming-QoE monitoring at devices installed near the end-user (e.g., home routers), where traffic load enables real-time monitoring with limited hardware capabilities.

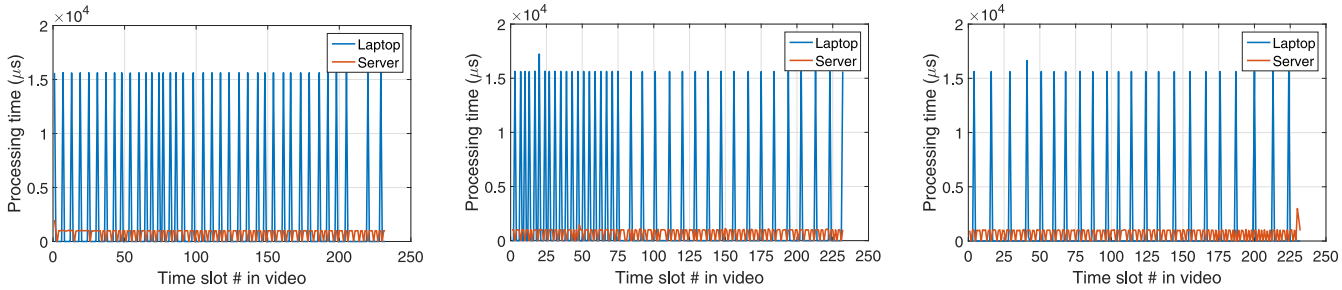
Feature Extraction: To demonstrate the real-time properties of ViCrypt for the feature-extraction process, we record at each packet arrival the time needed to update the feature set for a session lasting four minutes. The results are reported in Figure 11. Feature updates are performed extremely fast on both machines: they take only a couple of microseconds. On *server*, the peak value is about 12 ms, while the average duration is of only 13 μs. More than 90% of the updates took less than 25 μs. Even on *laptop*, the average processing duration is 37 μs, with a maximum value of 129 ms, which is still almost an order of magnitude smaller than the time-slot length of 1 s.

Stalling Detection: We measure the time needed by the most suitable model (ERT10) to detect the stalling of a time slot on both *laptop* and *server*. To do so, we train the model on 80% of the data, i.e., on four folds, and record, for one video session in the remaining fold, the time required to detect the occurrence of stalling per time slot. We use the same video as before, and, to make computations harder, we disable the parallelization of the algorithm for the estimation phase. The estimation times for this task are depicted in Figure 12(a). The model runs very fast, with an average of 740 μs and a maximum of 2 ms on *server*, and an average of 2.5 ms on *laptop*, which confirms that ViCrypt can also infer stalling in real time.

Video-Resolution Estimation: We measure the time needed by the best performing RF10 model for a single estimation of the video resolution, following the same procedure. Figure 12(b) shows the processing times for video-resolution estimation for the consecutive time slots of the exemplary

TABLE IX
VICRYPT PERFORMANCE FOR ESTIMATING THE AVERAGE BITRATE WITH ERT10 USING DIFFERENT FEATURE SUBSETS

Features	MAE [kbps]	RMSE [kbps]	MRE [%]	PLCC
All	93	182	19	0.93
F_C	275	407	55	0.58
F_T	253	377	51	0.64
F_S	68	157	14	0.95
F_{DOWN}	105	195	21	0.92
F_{UP}	106	198	21	0.92
F_{TOP20}	81	175	16	0.93



(a) Stalling. (b) Video resolution. (c) Average bitrate.

Fig. 12. Time needed to estimate video-QoE metrics from features for an exemplary YouTube video session. ViCrypt can perform all KQI estimations in real time, with an end-to-end delay significantly smaller than the time-slot length of 1 second.

TABLE X
REFERENCING PERFORMANCE COMPARISON BETWEEN VICRYPT, REQUET, AND INFOCOM'18 [4]. RESULTS CORRESPOND TO NUMBERS REPORTED IN [30] AND [4], FOR DIFFERENT DATASETS, SEE TABLE I

	sessions	Stalling		Video Resolution – P (%)						Video Resolution – R (%)					
		P (%)	R (%)	144p	240p	360p	480p	720p	1080p	144p	240p	360p	480p	720p	1080p
ViCrypt	15,000+	91.0	72.0	96.5	89.5	93.4	95.1	96.2	96.0	96.2	88.3	87.5	98.1	93.5	88.2
Requet	580	70.4	51.9	80.6	68.7	49.2	64.9	60.6	75.0	79.9	64.3	64.4	63.8	54.5	76.9
INFOCOM'18	10,863	80.9	83.9	73.2				80.1		71.1				82.1	

video. On *server*, almost all of the durations are around 1 ms, with an average of 700 μs and a maximum of around 1.4 ms; on *laptop*, the average duration is 2.5 ms and all estimations are available in less than 20 ms. Again, results confirm that ViCrypt performs video-resolution inference very fast, and significantly faster than the time slot length of 1 s.

Average-Bitrate Estimation: Finally, we analyze the time needed by the best model (here ERT10) to infer the average bitrate of a time slot. The observed estimation times are almost identical to the ones observed for the video resolution on both machines, as we can see in Figure 12(c), with an average value of 700 μs and a maximum of 3 ms on *server*, showing that ViCrypt can also infer average video bitrate in real time.

A. ViCrypt vs. State of the Art

To conclude our study, we provide some indicative results comparing the estimation performance of ViCrypt against the two most similar systems in the literature, namely Requet [30] and INFOCOM'18 [4]. While a re-implementation of both systems for benchmarking purposes is out of the scope of our study, we present in Table X the performance results reported by the authors of both approaches in the corresponding papers [4], [30]. Naturally, this is not intended as a valid or fair comparison among approaches, as the used datasets are not the same. Still, we decided to include the table to better position ViCrypt within the state of the art, and to serve as reference or baseline for the results presented in this study.

We consider the estimation of two of the KQIs, namely stalling and video resolution, as neither Requet nor INFOCOM'18 are designed to estimate the average video bitrate. In addition, while both ViCrypt and Requet tackle the video-resolution inference problem as a multi-class classification task, using exactly the same resolution levels, INFOCOM'18 considers only a binary classification task, defining *low video resolution* as all resolutions below 480p, and *high video resolution* for levels above 480p.

As already mentioned in Section II, the size and heterogeneity of the considered datasets is significantly different for the three systems. A particularly challenging issue for ViCrypt is that our dataset is highly imbalanced, especially when it comes to the occurrence of stalling, with only a small fraction of videos and time slots experiencing stalling. On the contrary, the dataset used in INFOCOM'18 is almost perfectly balanced in terms of stalling, with about 106,000 time slots corresponding to no-buffering and 94,000 slots reported as stalling.

Nevertheless, Table X shows that ViCrypt achieves similar or even better results than INFOCOM'18 in the binary detection of stalling events, and that both systems significantly outperform Requet, which presents quite poor results for stalling detection. Here, ViCrypt uses the ERT10 model

with only F_S features (see Table VII), which is still not the best of all models reported in the study – BAGGING performance using only F_S features is even higher. Regarding video resolution, ViCrypt provides highly accurate results in terms of precision and recall, while both Requet and INFOCOM'18 report lower performance on the classification task. INFOCOM'18 achieves relatively poor performance for video-resolution estimation, even if the KQI is addressed as a plain binary-classification task.

All in all, we can conclude that ViCrypt is not only able to estimate video quality metrics with high accuracy – comparable or even potentially outperforming the state of the art – but also to do it in real time, with minimal temporal computation requirements.

VII. Concluding Remarks

In this article, we presented ViCrypt, a machine-learning-driven system for real-time estimation of QoE-relevant metrics of video streaming, using a fine temporal granularity of only one second. This is, to the best of our knowledge, the finest granularity so far used for quality inference in the context of encrypted traffic. ViCrypt monitors the encrypted video traffic in a stream-like manner considering three windows (current slot, trend window, session window), from which statistical features are computed and updated with constant memory consumption.

We focused on the estimation of the most important Key QoE Indicators (KQIs), i.e., initial delay, stalling, visual quality, as well as the video bitrate, which are highly relevant for ISPs to monitor the end-user QoE and to enable proactive QoE-aware traffic management. We built a dataset containing more than 15,000 randomly chosen YouTube videos streamed under diverse network conditions, devices, ISPs, and transport protocols. We benchmarked multiple ML models and found that tree-based techniques are the most appropriate algorithms for ViCrypt.

Indeed, ERT10 (binary classification of stalling, which includes initial delay, with 99% accuracy; and continuous-valued estimation, i.e., a regression problem, of average bitrate with a mean absolute error of 68 kbps) and RF10 (classification of video resolution into six classes with 96% accuracy) provided highly promising results for all the considered QoE-relevant metrics and can be (re-)trained very fast. BAGGING performed even slightly better than RF10 and ERT10 for all three KQIs when using all features, but comes at the cost of significantly higher training times. We also analyzed the feature importance and showed that ViCrypt performed best with only a reduced feature set. Here, the features summarizing the characteristics of the session since the beginning of the streaming were the most relevant ones.

Overall, we demonstrated that ViCrypt is a very powerful tool to infer KQIs of YouTube from encrypted traffic in real time. As future work, we propose to further study to which extent the performance of ViCrypt could be improved by carefully crafting better feature subsets, or by following recurrent approaches considering also predictions from previous time slots. Furthermore, the performance of ViCrypt has to be revisited when applied to estimate KQIs for other streaming services, such as Amazon or Netflix.

References

[1] P. Casas, M. Seufert, and R. Schatz, "YOUQMON: A system for on-line monitoring of YouTube QoE in operational 3G networks," *ACM SIGMETRICS PER*, vol. 41, no. 2, pp. 44–46, 2013.

[2] R. Schatz, T. Hoßfeld, and P. Casas, "Passive YouTube QoE monitoring for ISPs," in *Proc. 6th Int. Conf. Innovat. Mobile Internet Services Ubiquitous Comput.*, Palermo, Italy, 2012, pp. 358–364.

[3] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "YouTube QoE estimation based on the analysis of encrypted network traffic using machine learning," in *Proc. GLOBECOM Workshops*, Washington, DC, USA, 2016, pp. 1–6.

[4] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for HTTPS and QUIC," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, 2018, pp. 1331–1339.

[5] M. Seufert, P. Casas, N. Wehner, L. Gang, and K. Li, "Stream-based machine learning for real-time QoE analysis of encrypted video streaming traffic," in *Proc. ICIN*, 2019, pp. 76–81.

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

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

[8] 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 Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 1st Quart., 2015.

[9] D. Ghadiyaram, J. Pan, and A. C. Bovik, "A time-varying subjective quality model for mobile streaming videos with stalling events," in *Proc. SPIE Appl. Digital Image Process. XXXVIII*, San Diego, CA, USA, 2015, Art. no. 959911.

[10] K. Zeng, H. Yeganeh, and Z. Wang, "Quality-of-experience of streaming video: Interactions between presentation quality and playback stalling," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Phoenix, AZ, USA, 2016, pp. 2405–2409.

[11] S. Egger, T. Hoßfeld, R. Schatz, and M. Fiedler, "Waiting times in quality of experience for Web based services," in *Proc. 4th Int. Workshop Qual. Multimedia Experience (QoMEX)*, Yarra Valley, VIC, Australia, 2012, pp. 86–96.

[12] C. Timmerer, M. Maiero, and B. Rainer, "Which adaptation logic? An objective and subjective performance evaluation of HTTP-based adaptive media streaming systems," 2016. [Online]. Available: arXiv:1606.00341.

[13] H. Ott, K. Miller, and A. Wolisz, "Simulation framework for HTTP-based adaptive streaming applications," in *Proc. Workshop Ns-3*, 2017, pp. 95–102.

[14] M. Seufert, N. Wehner, and P. Casas, "A fair share for all: TCP-inspired adaptation logic for QoE fairness among heterogeneous HTTP adaptive video streaming clients," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 475–488, Jun. 2019.

[15] *Information Technology—Dynamic Adaptive Streaming Over HTTP (DASH)—Part 1: Media Presentation Description and Segment Formats*, (ISO/IEC) Standard 23009-1:2012, 2012.

[16] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming," in *Proc. 6th Int. Workshop Qual. Multimedia Experience (QoMEX)*, Singapore, 2014, pp. 111–116.

[17] M. Seufert, T. Hoßfeld, and C. Sieber, "Impact of intermediate layer on quality of experience of HTTP adaptive streaming," in *Proc. 11th Int. Conf. Netw. Service Manag. (CNSM)*, Barcelona, Spain, 2015, pp. 256–260.

[18] H. T. T. Tran, T. Vu, N. P. Ngoc, and T. C. Thang, "A novel quality model for HTTP adaptive streaming," in *Proc. 6th IEEE Int. Conf. Commun. Electron. (ICCE)*, Ha Long, Vietnam, 2016, pp. 423–428.

[19] F. Wang, Z. Fei, J. Wang, Y. Liu, and Z. Wu, "HAS QoE prediction based on dynamic video features with data mining in LTE network," *Sci. China Inf. Sci.*, vol. 60, no. 4, 2017, Art. no. 042404.

[20] O. Oyman and S. Singh, "Quality of experience for HTTP adaptive streaming services," *IEEE Commun. Mag.*, vol. 50, no. 4, pp. 20–27, Apr. 2012.

[21] B. Lewcio, B. Belmudez, A. Mehmood, M. Wältermann, and S. Möller, "Video quality in next generation mobile networks—Perception of time-varying transmission," in *Proc. IEEE Int. Workshop Tech. Committee Commun. Qual. Rel. (CQR)*, Naples, FL, USA, 2011, pp. 1–6.

[22] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Flicker effects in adaptive video streaming to handheld devices," in *Proc. 19th ACM Int. Conf. Multimedia (MM)*, Scottsdale, AZ, USA, 2011, pp. 463–472.

[23] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, and H. Yan, "Prometheus: Toward quality-of-experience estimation for mobile apps from passive network measurements," in *Proc. 15th Workshop Mobile Comput. Syst. Appl. (HotMobile)*, Santa Barbara, CA, USA, 2014, pp. 1–6.

[24] 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 Trans. Netw. Service Manag.*, vol. 13, no. 2, pp. 181–196, Jun. 2016.

[25] P. Casas *et al.*, "Predicting QoE in cellular networks using machine learning and in-smartphone measurements," in *Proc. 9th Int. Conf. Qual. Multimedia Experience (QoMEX)*, Erfurt, Germany, 2017, pp. 1–6.

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

[27] S. Wassermann, N. Wehner, and P. Casas, "Machine learning models for YouTube QoE and user engagement prediction in smartphones," *SIGMETRICS PER*, vol. 46, no. 3, pp. 155–158, 2019.

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

[29] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "eMIMIC: Estimating HTTP-based video QoE metrics from encrypted network traffic," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Vienna, Austria, 2018, pp. 1–8.

[30] C. Gutterman *et al.*, "Requet: Real-time QoE metric detection for encrypted YouTube traffic," in *Proc. ACM Multimedia Syst. Conf. (MMSys)*, 2019, p. 71.

[31] 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 *Proc. IEEE/IFIP Netw. Traffic Meas. Anal. Conf.*, Dublin, Ireland, 2017, pp. 1–6.

[32] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "YoMoApp: A tool for analyzing QoE of YouTube HTTP adaptive streaming in mobile networks," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Paris, France, 2015, pp. 239–243.

[33] M. T. Seufert, "Quality of experience and access network traffic management of HTTP adaptive video streaming," Ph.D. dissertation, Fakultät für Mathematik und Informatik, Univ. Würzburg, Würzburg, Germany, 2017. [Online]. Available: https://opus.bibliothek.uni-wuerzburg.de/files/15413/Seufert_Michael_Thomas_HTTP.pdf

[34] T. Karagkioules *et al.*, "A public dataset for YouTube's mobile streaming client," in *Proc. Netw. Traffic Meas. Anal. Conf.*, Vienna, Austria, 2018, pp. 1–6.

[35] A. L. Strehl and M. L. Littman, "Online linear regression and its application to model-based reinforcement learning," in *Proc. 20th Int. Conf. Neural Inf. Process. Syst.*, 2007, pp. 1417–1424.

[36] P. Pébay, "Formulas for robust, one-pass parallel computation of covariances and arbitrary-order statistical moments," Sandia Natl. Lab., U.S. Dept. Energy, Livermore, CA, USA, Rep. SAND2008-6212, 2008.

[37] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.

[38] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.

[39] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2000, pp. 93–104.

[40] P. Geurts, A. Irrthum, and L. Wehenkel, "Supervised learning with decision tree-based methods in computational and systems biology," *Mol. BioSyst.*, vol. 5, no. 12, pp. 1593–1605, 2009.

Sarah Wassermann received the bachelor's and master's degrees from the University of Liège, Belgium, in 2015 and 2017, respectively. She is currently pursing the Ph.D. degree with TU Wien, doing research in network measurements, in particular in the field of QoE and machine learning. Her goal is to conceive intelligent systems which make the Internet smarter and able to face demanding users and an ever-growing volume of heterogeneous network traffic.

Michael Seufert (Member, IEEE) received the bachelor's degree in economathematics and the Diploma and Ph.D. degrees in computer science from the University of Würzburg, Germany. He is currently pursuing the Habilitation degree from the University of Würzburg, while leading the chair's research activities towards user-centric communication networks. From 2012 to 2013, he was with the FTW Telecommunication Research Center, Vienna, Austria. From 2013 to 2017, he was a Researcher with the Chair of Communication Networks, University of Würzburg. From 2018 to 2019, he was a Postdoctoral Fellow and a Scientist with the AIT Austrian Institute of Technology, Vienna. Since 2019, he has been a Postdoctoral Fellow and a Scientist with the Chair of Communication Networks, University of Würzburg. His research focuses on QoE of Internet applications, artificial intelligence and machine learning for networks, monitoring and analytics of (encrypted) network traffic and orchestration of edge cloud services, proactive QoE- and socially-aware traffic management solutions, and performance modeling of communication systems.

Pedro Casas (Member, IEEE) received the Electrical Engineering degree from the Universidad de la República, Uruguay, in 2005, and the Ph.D. degree in computer science from Télécom Bretagne in 2010. He is Senior Scientist in AI/ML for Networking with the AIT Austrian Institute of Technology in Vienna. He was Postdoctoral Research with the LAAS-CNRS in Toulouse from 2010 to 2011, and Senior Researcher with the Telecommunications Research Center Vienna from 2011 to 2015. He has published more than 180 Networking research papers in major international conferences and journals. His work focuses on machine-learning-based approaches for Networking, big data analytics and platforms, Internet network measurements, network security, and anomaly detection, as well as Internet QoE monitoring. He received 14 awards for his work, including seven best paper awards. He is General Chair for different actions in Network measurement and analysis, including the IEEE ComSoc ITC Special Interest Group on Network Measurements and Analytics.

Li Gang works as a Technical Pre-Research Engineer with Huawei Technologies. He joined Huawei, and has been working in multiple projects linked to network traffic monitoring and analysis, including in particular video streaming (QoE) analysis, more than ten years ago.

Kuang Li received the Ph.D. degree on computer science from Wuhan University in China. He has worked several years on researching intelligent operation and the management of communication networks in Huawei Technologies.