Unveiling YouTube QoE over SATCOM using Deep-Learning

Matthieu Petrou, David Pradas, Mickaël Royer, Emmanuel Lochin

Unveiling YouTube QoE over SATCOM using Deep-Learning

MATTHIEU PETROU[1,2], DAVID PRADAS[1], MICKAËL ROYER[3], EMMANUEL LOCHIN[3],
[1]Viveris Technologies, Toulouse, France
[2]ISAE-SUPAERO, Toulouse, France
[3]Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, France

Corresponding author: Matthieu Petrou (e-mail: matthieu.petrou@viveris.fr).

This research was funded by Viveris Technologies.

ABSTRACT The importance of stored streaming video for current Internet traffic is undeniable, even in the context of satellite communications (SATCOM). Therefore, Internet service providers aim to deliver the highest quality of experience to their end users, although they are not able to assess it directly. Some machine learning techniques proposed in the literature have demonstrated their ability to predict the quality of experience based on traffic data analysis. However, these models cannot be directly applied in a SATCOM context without considering the specific characteristics of satellite links. Furthermore, some of them may not be suitable for real-time use.

In this study, we monitored over 2,400 YouTube video sessions over an emulated satellite network to develop models capable of predicting the initial delay, played resolution, and stalling events. The collected dataset is available as an open source to the research community. The primary objective of this research is to provide a functional model for real-time applications. To achieve this, we reduced the required feature set to minimize computation time and resources, enabling a practical real-time implementation of the model while assessing its feasibility. We show that we successfully achieved a substantial reduction in the number of features while also observing a relative improvement in prediction. Our results yield prediction performance comparable to that of other studies on terrestrial networks. Using the reduced feature set, we present a real-time implementation and confirm the real-time viability of our work.

INDEX TERMS Deep Learning, HTTP Adaptive Video Streaming, Machine Learning, Network Monitoring, QoE, SATCOM.

I. INTRODUCTION

The Sandvine report of 2020 [1] revealed that video streaming plays an important role in today's internet, by representing more than 57% of global traffic share. This report looked specifically at satellite traffic as well and reported that video streaming also represents a major part of it. Specifically, YouTube represents more than 16% of the satellite traffic share and Netflix more than 9%. Therefore, the Quality of Experience (QoE) of video streaming application is important for network operators and, consequently, is widely studied in the literature [2].

One of the main challenges for Internet Service Providers (ISPs) is to provide the best QoE to end users. Nevertheless, QoE evaluation is restricted to end user applications and is not visible to ISPs, which can access only to Quality of Service (QoS) metrics. Additionally, the encryption of most today's traffic prevents the IPSs from using Deep-Packet-Inspection (DPI). Several previous studies have provided methods and approaches to predict QoE of streaming video in real time, based on QoS information. Nevertheless, to the best of our knowledge, no studies have been conducted on GeoSynchronous Orbit (GSO) satellite links.

GSO satellite links are an essential part of today's global connectivity, providing connections to isolated areas that lack communication infrastructure. Satellite networks also provide a vital solution for emergency communications, disaster-stricken areas, such as those affected by natural disasters or war, and earth and global warming observations [3]. Because of the altitude of the satellite, connections over GSO satellites suffer from a high latency, with a round-trip time (RTT) over the satellite network of about 600 ms. Therefore, it is impossible to directly apply existing QoE predictive models to satellite networks. This disparity stems from the absence of terrestrial equivalents for the entire physical and access

layers within the SATCOM context. Consequently, established standards, such as instance the ITU standard for VoIP QoE assessment (ITU-T G1011), introduce specific biases when applied to SATCOM, as highlighted by the observed increase in "advantage factor" ranging from 20 to 40 [4], [5]. Therefore, the unique characteristics of SATCOM necessitate a comprehensive reevaluation and adaptation of current QoE evaluation methodologies.

The main objective of this work is to provide a practical solution for predicting end user QoE to ISPs and offer an open dataset. To build the dataset, we monitored 2,400 video sessions over emulated GSO satellite networks under various scenarios, collecting QoE metrics from YouTube and QoS metrics from traffic data. Our second objective is to select the best Machine Learning (ML) algorithms, between Long Short-Term Memory (LSTM) and Random Forest (RF), which both have been considered in previous studies. The emulated testbed is presented in Section III-A.

The selection of both RF and LSTM algorithms for QoE evaluation results from the state-of-the-art presented in Section II-C and summarized in Table 1. While much recent research has employed RF, Loh et al. (2021) [6] demonstrated promising results using LSTMs. This interest likely stems from the popularity of LSTMs in handling time-series data. LSTMs excel at capturing and learning temporal dependencies, with a key advantage being their ability to process sequential data points and retain information over extended periods, making them well-suited for modeling time-dependent patterns. Notably, RFs also demonstrate efficiency in time-series regression [7], but with significantly lower computational cost than LSTMs. Considering our objective of developing a real-time solution (existing studies typically disregard the transition from theory to practice) and benchmarking against the state-of-the-art, we opted for this dual-algorithm approach.

Next, we focus on the most effective machine learning algorithm, specifically LSTM in our scenario, with the objective of reducing computation time and resources without significantly affecting performance. To achieve this, we employ various techniques to decrease the number of features and subsequently conduct a comparative analysis. In the end, the models using the smaller feature set achieve the best performance, comparable to the state-of-the-art results on other networks. Finally, we implement the best approach in a real-time solution and confirm the real-time application of our work.

The remaining part of the paper proceeds as follows: Section II defines background details of stored video streaming and summarizes related work. Section III describes the methodology used to collect and process the data and analyze the collected dataset. Section IV presents both compared ML algorithms, RF and LSTM, their first performance results, and concludes with a comparison of these results. In Section V, we decrease the number of features to reduce computation time and resources, and compare the performance results of LSTM trained on various feature set sizes. This section also compares the performance results with those of previous studies. Section VI offers a practical real-time implementation of our work and assesses its feasibility and viability. Finally, Section VIII concludes this study.

II. BACKGROUND

This section provides an overview of background knowledge related to stored video streaming. In addition, this section provides a review of previous studies on predicting video streaming QoE.

A. STORED VIDEO STREAMING QOE

In stored video streaming, the audio and video content are downloaded while being played to the user. Therefore, the available downlink throughput must be sufficient to match the video encoding rate. A video buffer is implemented to prevent user experience issues in the event of short mismatches between the download speed and the video encoding rate. As long as the buffer remains non-depleted, the video continues to play. Otherwise, a stalling event occurs and the video pauses while the buffer refills.

Videos are divided into segments and chunks. Segments are regular time-slices of a few seconds, whereas chunks are slices of data of regular-size. Chunks and segments are not directly related because a single chunk may contain multiple segments or fewer than one. At the network level, only one chunk is requested and downloaded at a time. Nevertheless, the video and audio chunks can be downloaded in parallel. At the application level, the player updates the buffer with complete segments, even from a chunk partially downloaded. The video buffer stores multiple video segments. Depending on the buffer state, the resolution of the next requested segments can change. Indeed, if the buffer is not filled as quickly as it is emptied, one solution to reduce the required data rate is to reduce the requested resolution.

B. QUALITY OF EXPERIENCE

The quality of experience is the user's subjective perception and expectations toward a given service and is defined by Qualinet White Paper [8] as *"the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state."*.

Previous studies [8], [9] proposed four categories of Influencing Factors (IFs) that impact QoE: Human-related, System-related, Context-related, and Content-related IFs. From an ISP's point of view, network related IFs, which are system-related IFs, are the main Key Performance Indicators (KPIs), i.e., packet loss, delay, latency, jitter, and throughput. For stored video streaming, the main Key Quality Indicators (KQIs) are stalling events, initial delay, and resolution [10]:

- ***"Initial delay"*** refers to the time between the first request and the start of the video. In this study, both page load time and video initial delay are considered. There-

fore, we use the first request to the YouTube servers as the original timestamp.

- ***"Resolution"*** describes the quality of the application. It is usually determined by the number of vertical pixels that indicate the resolution in the image ($144p$, $240p$, $360p$, $480p$, $720p$, and $1080p$). As previously mentioned, the resolution may vary during video playback. Such variation also impacts QoE for the user. A previous study [10] showed that users prefer a constant lower resolution to several resolution changes.
- ***"Stalling"*** defines the interruption of video playback. During this time, the application attempts to refill the buffer with segments. These events have the strongest negative impact on QoE. Previous study [11] showed that increasing the initial delay can help reduce or prevent stalling, which improves the QoE. Therefore, applications such as YouTube or Netflix aim to minimize and prevent stalling events from occurring, by reducing the resolution or forcing a longer initial delay, which provides more time to fill the buffer before the video starts.

C. RELATED WORK

This section covers the different ML approaches and methodologies used in prior studies to predict QoE for end users based on QoS. Table 1 summarizes relevant previous related studies that predict QoE KQIs. The table presents the algorithm and platforms used, along with the predicted KQIs. In addition, the table presents the number of video sessions collected and the real-time compatibility of these previous studies. As presented in Table 1, most of the predicting QoE studies have targeted YouTube, either on desktop or mobile devices.

Mazhar's work [12] exploited features based on TCP flags, and for QUIC flows they used network layer information to create features. Gutterman's study [13] used a chunk detection system to process chunk-based features. These chunk-based features predict the resolution, buffer filling phase, and buffer warning, which is triggered when the buffer is below a threshold of 20 seconds. In the same vein, Bronzio & Schmitt's study [14] also exploited segment detection to add application-based features, for Netflix, YouTube, Amazon, and Twitch videos. They showed that the use of these features improves prediction accuracy for the initial delay and the resolution compared with the use of only network layer and transport layer related features. Orsolic's and Wassermann's studies [15], [16] are using network related features processed over different time windows to predict resolution and bit rate. Additionally, Wassermann's work also predicts the initial delay and stalling. For the four KQIs, Wassermann's study obtained high performance with RF using 5-fold cross validation. Shen's work [17] is the only work cited here that also focused on the Bilibili application, and the only one that used a Convolutional Neural Network (CNN). Loh's work [6] revealed that the use of only uplink-related features gives predictions that are reasonably as good as using both uplink and downlink features, for the prediction of resolution, initial delay, buffer filling phase, and stalling events.

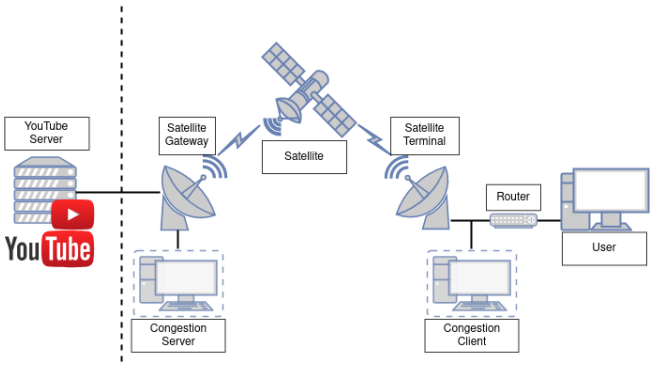


FIGURE 1. Topology of the test bench. The user is connected to the Internet through an emulated satellite link.

III. METHODOLOGY

This section details the testbed and the scenarios used to collect data. It also describes the processing of QoE and QoS data, and provides an analysis of the collected dataset. This dataset is available on github[1].

A. EXPERIMENTAL SETUP

Our testbed relies on an emulated satellite link that connects the user to the Internet. To emulate the satellite link, we use OpenSAND [18], [19], an open-source end-to-end satellite communication system emulator. OpenSAND emulates SATCOM systems with a fair representation [20]. Three components compose the satellite link: a satellite gateway, a satellite terminal, and a GSO satellite. The satellite gateway is connected to the Internet. The user is connected to the satellite terminal through a router and has Internet access through the satellite emulated link. We choose not to set up any Performance Enhancing Proxies (PEP) over the satellite link, as the QUIC protocol, used by YouTube servers, does not benefit from PEP optimization. Figure 1 depicts the testbed topology.

The router enables us to add packet loss for specific scenarios, for example, to emulate a Wi-Fi. In addition, for specific scenarios, additional clients and servers are added behind the satellite terminal and the satellite gateway to generate congestion.

We orchestrate and launch the tests, including OpenSAND, Firefox, packet capture, and congestion flows, using OpenBACH [21], an open-source network metrology test bench. An OpenBACH script collects YouTube metrics from the SATboost plugin [22], which obtains information from YouTube interface[2]. We use only the YouTube collection feature of SATboost, not the optimization features. The user computer has an Intel(R) Core(TM) i3-3220 CPU with 4

[1] https://github.com/viveris/satcom-qoe-dataset
[2] Data are collected from the "Stats for Nerds" interface

TABLE 1. Overview of previous work relative to QoE prediction.

Reference	Platform	Algorithms	KQIs Prediction	Real-Time	Video sessions
Mazhar et al. 2018 [12]	YouTube desktop	DT	Initial Delay, Stalling, Resolution	Yes	10,863
Gutterman et al. 2019 [13]	YouTube mobile and desktop	RF	Buffer Warning, Buffer Filling Phase, Resolution	Yes	600
Bronzino & Schmitt et al. 2019 [14]	Netflix, YouTube, Amazon & Twitch desktop	RF, DT & others	Initial Delay, Resolution	No	13,765
Orsolic et al. 2020 [15]	YouTube Mobile	DT, RF	Resolution, Bit Rate	Yes	992
Shen et al. 2020 [17]	Bilibili, YouTube desktop	CNN	Initial Delay, Stalling, Resolution	Yes	10,000+
Wassermann et al. 2020 [16]	YouTube mobile & desktop	mainly RF	Initial Delay, Stalling, Resolution, Bit Rate	Yes	15,000+
Loh et al. 2021 [6]	YouTube mobile	RF, LSTM	Resolution, Initial Delay, Buffer Filling Phase, Stalling	Yes	13,759
Our work	YouTube desktop over SATCOM	RF, LSTM	Resolution, Initial Delay, Stalling	Yes	2,400

core of 3.30GHz and a Intel HD integrated GPU. Tests are performed with Ubuntu 20.04, Firefox version 97.0.1,and Geckodriver 0.30.0 [23], the latter allows us to remotely control Firefox. The uBlock Origin ad blocker plugin [24], has been enabled in Firefox to prevent the collection of advertising data. Traffic data are captured on the satellite terminal, whereas YouTube data are collected on the user.

For the tests, we selected 40 unique videos with durations ranging from 30 s to 17 min 32 s to include a diverse representation of the content available to users. Each video is available in the same set of resolutions: $144p$, $240p$, $360p$, $480p$, $720p$, and $1080p$, and has between 24 and 30 frames per second. To ensure a fair representation of available video content on the YouTube platform, we choose videos from various YouTube channels and of diverse styles.

With a total of $2,400$ monitored video sessions, our dataset comprises a smaller number of sessions compared to recent studies, as shown in Table 1. Nonetheless, to align with users' engagement patterns on YouTube, we deliberately opt for longer videos. For example, the Wasserman et al. dataset, which includes over $15,000$ videos, consists of $4,600,000$ time slots of one second, averaging approximately 5 minutes per video. In contrast, our dataset monitors 1,214,536 time slots of video playback, extending to $1,306,761$ time slots when accounting for initial delays. Although our dataset contains 6.25 times fewer videos, the total time slots are 3.8 times smaller. This deliberate choice allows us to match users' typical engagement on YouTube, selecting video durations based on trending content to mirror user use patterns

B. SCENARIOS

We perform the tests using two bandwidths on the satellite forward link: 1 Mb/s and 12 Mb/s. These bandwidths were chosen on the basis of their representation of realistic public satellite Internet access and in alignment with previous studies [25], [26]. In both cases, the emulated satellite link has a one-way delay of 250 ms.

To diversify our dataset, we include other scenarios in addition to these two baseline scenarios. Two of them are without any degradation from loss on the link or congestion. We add a scenario that adds pressure on the YouTube application to induce some stallings. In this scenario, a regular prioritized UDP flow is added, which uses 90% of the available bandwidth for 1 minute every 2 minutes. The prioritized UDP flow is generated using Iperf3 (v3.10.1) [27]. Two congestion scenarios are included, where one or two additional clients use YouTube simultaneously with the main user. A final scenario is added with losses over the satellite link. To simulate a realistic satellite use case, we select a Gilbert-Elliot model based on data collected from a railway train, which was previously used in a study [28]. This Gilbert-Elliot model, with parameters $p = 0.016$ and $q = 0.938$, is applied to the emulated satellite link.

C. FEATURES

The previous section describes the scenarios used to collect data. In this section, we provide an overview of the computation of features for ML algorithms, based on the data obtained from these scenarios. Features tuning is an essential aspect of machine learning, as identifying the most important features

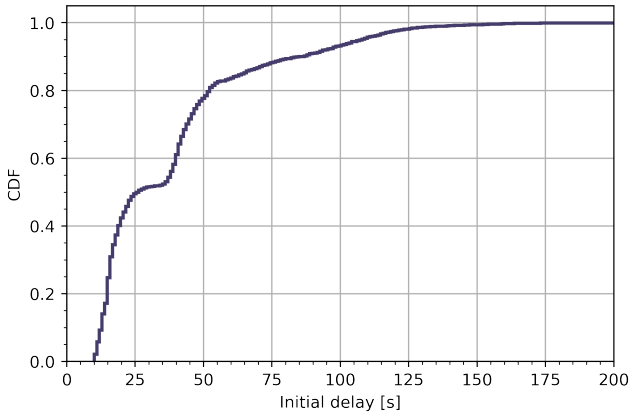


FIGURE 2. CDF of the initial delay

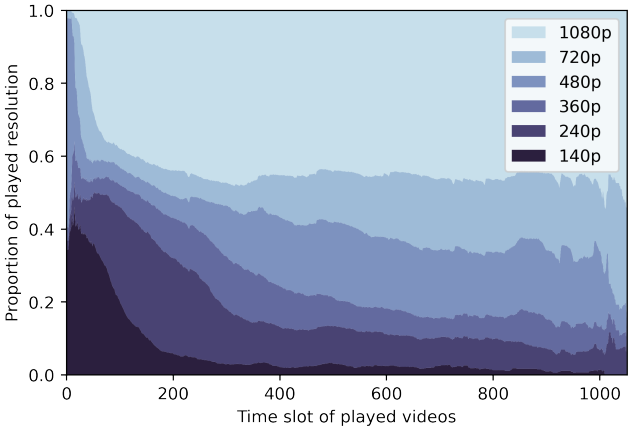


FIGURE 3. Resolution proportions for each time slot.

can improve performance and reduce computational costs. This section provides details about the features used in the ML algorithms.

We use ViCrypt paper [16] methodology to compute features from packet capture, where we collect the size, timestamp, and IP source/destination of packets. Their method considers what has occurred over three different time windows: the last second, the last 3 s, and the entire session. Nevertheless, we made some modifications to their method. For example, we completely removed the TCP and UDP related features because they do not improve the accuracy of our results. This approach produces a total of 199 features, which include 66 features for each time window and an additional feature equal to the number of the current time slot.

D. DATA ANALYSIS

We test 40 unique videos under six different scenarios, with each test performed ten times. Hence, the dataset contains $2,400$ video sessions, for a total of $1,306,761$ time slots of one second that represents more than 15 days. This section provides an analysis of the KQIs in the dataset.

1) Initial Delay

It is noteworthy that we use the timestamp of the first request to YouTube servers as the starting point of the initial delay, and the beginning of the video as the end time. Figure 2 represents a Cumulative Distribution Function (CDF) of the collected initial delays. Consistent with the time slot length used, the granularity for the initial delay is 1 s. As shown in the figure, the shortest initial delays monitored are 10 s. Over half of the videos monitored start within 27 s, which mainly consist of videos sessions with 12 Mb/s bandwidth. The average initial delay is 38.43 s, primarily influenced by the longest 90th percentile of initial delays exceeding 100 seconds. These initial delays are notably lengthy compared with studies on terrestrial networks, which generally have initial delays of approximately $2-3$ s [6], [16].

2) Resolution

The resolution played is information directly provided by the YouTube interface. Nevertheless, when a resolution change occurs, it is not possible to know the resolution played during the last second because data from the YouTube interface are collected every second. Therefore, and as done in previous study [6], we remove the resolution label for time slots with resolution changes. Finally, resolution labels are also ignored when a stall occurs.

Figure 3 provides the ratio of resolutions for each time slot of the video sessions. In Figure 3, as the number of time slots increases, the number of monitored time slots decreases because of variations in the length of the video sessions. What is striking about this figure is the variation in the resolution distributions over time. Within the 100th first time slots, $1080p$ resolution represents a significant portion of the resolutions played. $360p$ and $720p$ resolutions are under-represented in the first 250th time slots, and gain in importance later on.

Table 2 summarizes the distribution of resolutions in the dataset. We monitor video sessions with the "auto" mode enabled, which means that the application determines itself the downloaded and displayed resolution.

The table highlights an imbalance in the resolution dataset, with more than 40% of time slots in $1080p$. This value can be explained by the data shown in Figure 3, where $1080p$ resolution accounts for more than 40% of the collected data after the first 100 s, due to 12 Mb/s scenarios. As the video duration increases, the distribution of the collected resolutions becomes more skewed. For instance, in a video lasting 17 minutes and 30 seconds (1050 seconds), with only 100 seconds not at $1080p$, approximately 90% of the video uses $1080p$. Even in a 10-minute video, where 500 seconds out of the 600 (83.33%) are at $1080p$. Given that half of our collected videos use 12 Mb/s, it is logical to observe that $1080p$ resolution constitutes 40% of the collected resolutions.

TABLE 2. Overview of resolutions in the dataset

Resolution	Number of Time Slots	Percentage
144p	111,611	9.39%
240p	177,710	14.95%
360p	104,710	8.81%
480p	170,296	14.33%
720p	140,080	11.78%
1080p	484,320	40.74%

E. STALLING EVENTS

Unlike the resolution, the information on the stalling events is not directly accessible from the YouTube interface. Nevertheless, we monitor the number of frames read and the frame rate of the videos. We label a time slot as a stalling event when the difference between two consecutive seconds of the number of frames read is less than 90% of the video frame rate, and there are fewer than 20 seconds in the buffer.

As mentioned earlier, stalling events have the most negative impact on KQIs on the QoE of end users. Over the $1,214,536$ time slots of video playback monitored, only $17,022$ (1.40%) are stalling time slots. Figure 4(a) represents the number of stalling event distributions in the dataset. From this figure, we observe that there is no stalling among 65% of video sessions. About 20% of video sessions have only one stalling event, and 95% of video sessions in the dataset have fewer than 3 stalling events. Figure 4(b) provides the duration of the stalling event distribution in the dataset. 25% of the stalling events last one second, and half of them last less than 9 seconds.

IV. SELECTION OF MACHINE LEARNING ALGORITHMS

This section provides a detailed description of the various machine learning algorithms used and compared in this study. Next, we discuss the first results obtained. Finally, we describe our proposals to reduce the computation time, which are used for the rest of the paper.

A. COMPARISON OF THE MACHINE LEARNING ALGORITHMS

To ensure the independence of the test dataset from the training dataset, we select 8 unique videos from the 40 monitored videos and keep them aside as the test group. The duration of these test videos is spread evenly over the duration of the monitored videos. Therefore, we train the machine learning models using a set of 32 distinct videos, and subsequently evaluate their performance on a completely separate set of 8 videos.

1) Random Forest

As mentioned in Section II-C, most of the previous studies [6], [13]–[15] used RF to predict KQIs. More specifically, Wassermann's [16] with ViCrypt also used RF. Since we use their approach for calculating features, we also decide to use RF, with the scikit-learn RF classifier [29], [30]. Table 3 summarized RF hyperparameters set for each KQI. The chosen hyperparameters are selected following a search by 5-fold cross-validation on the training dataset, using the mean F1 score as the target metric. Usually, shuffling and random division of the training dataset are used. Nevertheless, as the video session data are time-related, we choose not to split the training dataset directly, but to split it by video sessions to perform the 5-fold cross-validation.

2) Long Short-Term Memory

As data and prediction are time-dependent, LSTM algorithms can be a potential solution, as performed in previous work [6]. In this study, we use the LSTM from Pytorch implementation [31]. Nevertheless, it is important to remember that LSTM algorithms require a longer training time than RFs. In this study, each LSTM model is trained to predict the three KQIs (initial delay, resolution, and stalling). Thus, during the training process, the prediction errors are aggregated by adding them together, leading to de facto competition among the different predictions for these KQIs. As is conventionally performed for the LSTM algorithm, the features are standardized for the LSTM algorithms.

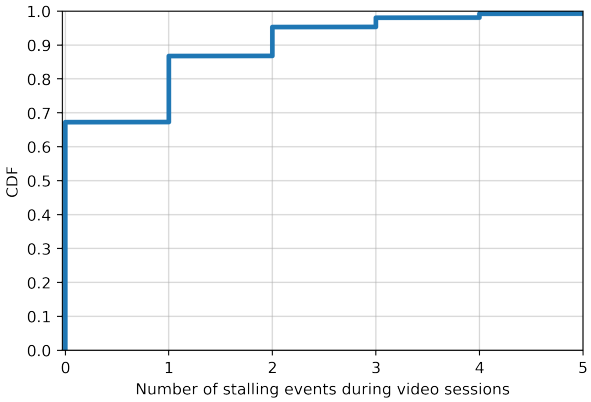
B. COMPARISON OF THE RESULTS

Table 4 presents an overview of the average F1 score for stalling events and resolution prediction, as well as the Mean Absolute Error (MAE) for initial delay, comparing the performance of the LSTM and RF models. In the table, LSTM demonstrates significantly superior performance compared to RF across all predictions. When predicting the initial delay, LSTM achieves a slightly lower MAE than RF, indicating an improvement in accuracy of around 9%. Regarding the resolution prediction, LSTM achieves an F1-score approximately 5% higher than RF, while for stalling prediction, LSTM's F1-score outperforms RF by around 6%. These metrics clearly illustrate the efficiency of LSTM over RF.

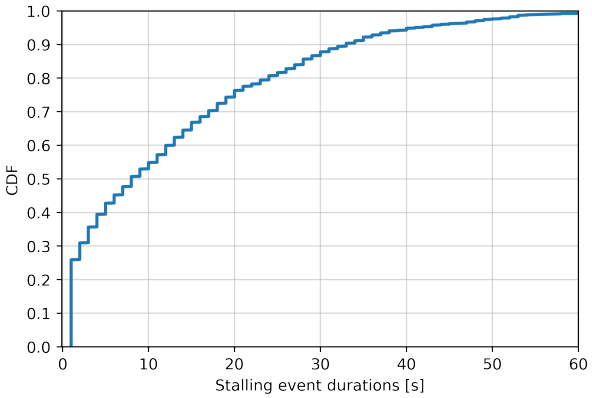
Considering the significant performance gap between LSTM and RF, it is evident that RF falls behind, whereas LSTM demonstrates superior predictive performance in our scenarios. Consequently, for the remainder of this paper, we focus on using LSTM. Nevertheless, LSTM models generally require more time and computational resources than RF models. Therefore, to mitigate these burdens, we intend to reduce the feature set used in the LSTM models.

V. OPTIMIZATION RESULTS

This section describes the various approaches used to reduce the number of features, and presents the performance results of the LSTM models trained on these feature sets. We provide details of the prediction results for the initial delay, resolution, and stalling events in that order.



(a) Distribution of the number of stalling events during video sessions.



(b) Distribution of stalling event length in second.

FIGURE 4. Distributions regarding stalling events.

TABLE 3. Random Forest Hyperparameters.

KQIs	Criterion	Bootstrap	n_estimators	max_depth	max_features
Initial Delay	*entropy*	False	1000	None	*sqrt*
Resolution	*entropy*	True	3000	None	*sqrt*
Stalling	*log_loss*	False	1000	None	*sqrt*

TABLE 4. Performances of LSTM and RF models for each KQI.

	LSTM	RF
MAE of Initial Delay	2.138 s	2.354 s
Average F1-score for Resolution	66.36%	61.08%
Average F1-score for Stalling Events	83.51%	76.88%

A. OPTIMIZATION OF LSTM FEATURES

We seek to reduce the computation cost to make our approach more practical for a potential real-world implementation. Hence, we attempt to identify features that we estimate as less useful and remove them.

We choose four approaches:

- $Fs^{y}_{\mathbf{all}}$: this approach contains the entire feature set;
- $Fs^{y}_{\mathbf{1}}$: for this version, we exclude features related to ratios (i.e., download/upload packets and bytes ratios) and those related to variance when a corresponding feature of standard deviation exists, as they are mathematically related through a quadratic relationship;
- $Fs^{y}_{\mathbf{2}}$: for this feature set, we take Fs^{y}_{1} set and discard the features related to the size of the downloaded packets. We decide to discard these data because the downloaded packet should have low variance in their length; and
- $Fs^{y}_{\mathbf{3}}$: finally, for this approach, we dismissed the same features as Fs^{y}_{2} set and the one related covariance between packet size and arrival time; i.e., the *Temporal Variance*, *Covariance Time-Size*, *Mean Cumulative Size*, *Mean Time*, *Slope*, and *Intercept*, for both download and upload packets, which are jointly calculated. Therefore, by removing them, our objective is to reduce the number of features by 12, hopefully without significantly affecting the results.

In the Appendix , the Table 8 presents a comprehensive list of the features included in our distinct approach.

Lastly, as LSTM has a feedback connection from previous predictions, we want to reduce the number of features by removing the different time windows. Therefore, we introduce three more approaches:

- $Fs^{\mathbf{3}}_{x}$: This approach has features from the three time windows, i.e., the last second, the last 3 seconds and the whole session; and
- $Fs^{\mathbf{2}}_{x}$: This approach has features from two time windows: the last second and the entire session. With this approach, we expect to maximize the use of LSTM and its short-term memory by removing the feature related to the last 3 seconds; and
- $Fs^{\mathbf{1}}_{x}$: This approach has features only from the last second. With this final approach, we aim to rely solely on the LSTM memory and, in so doing, drastically reduce the number of features.

By combining the reductions in the number of features, based on the time windows and their types, we arrive at 12

TABLE 5. Number of features in each feature set. In brackets, the relative percentage compared with the initial feature set (Fs^3_{all}). This ratio allows weighing the decrease in the number of features.

Feature set	Second (Fs^1_x)	Second + Session (Fs^2_x)	Second + Trend + Session (Fs^3_x)
Fs^y_{all}	67 (33.67%)	133 (66.83%)	199 (100.00%)
$Fs^y_{\mathbf{1}}$	47 (23.62%)	93 (46.73%)	139 (69.85%)
$Fs^y_{\mathbf{2}}$	40 (20.10%)	79 (39.70%)	118 (59.30%)
$Fs^y_{\mathbf{3}}$	28 (14.07%)	55 (27.64%)	82 (41.21%)

feature sets. The Table 5 presents an overview of the number of features in each approach.

B. COMPARATIVE ANALYSIS OF THE PERFORMANCE OF FEATURE REDUCTION APPROACHES IN LSTM

This section presents the performance results of the LSTM models trained on various feature sets.

As described in Section IV-A2, each LSTM model is trained to predict the three previously defined KQIs: initial delay, resolution, and stalling events. During the training process, we calculate the prediction errors of each KQI and sum them. As a result, all of these KQIs are aggregated into a unique value without weighting one of them. This leads to a compromise, as the model focuses on reducing the overall error rate, which may lead to varying performance levels for each specific KQI. Therefore, it is important to consider this when comparing KQI-by-KQI performance.

To ensure a fair and unbiased comparison, we conducted 11 training iterations for each feature set. This approach allowed us to consistently evaluate the performance of each model and account for any variations or randomness in the training process.

1) Initial Delay

Figure 5 displays the whisker boxes representing the completed MAE for each feature set. Notably, the poor performance of the Fs^1_x models is evident, and they occasionally yield a MAE greater than 3 seconds. Conversely, the model trained using Fs^2_x demonstrates superior overall performance, particularly for Fs^2_{all} and Fs^2_1.

To further investigate our performance, Figure 6 summarizes the initial delay prediction results of the model achieving the median average F1-score, by showing the percentage distribution of the absolute values of the prediction error. Most of the models can accurately predict the start of videos without any error in over 30% of the cases. Almost all models can predict the start of the video in 90% of cases, with an error of up to 5 seconds. Since about 10% of the cases have an absolute error greater than 5 seconds may appear significant; however, the longest 90% of the initial delays exceed 100 seconds.

2) Resolution

Figure 7 illustrates the whisker boxes presenting the average F1-score achieved for each feature set. In contrast to the initial

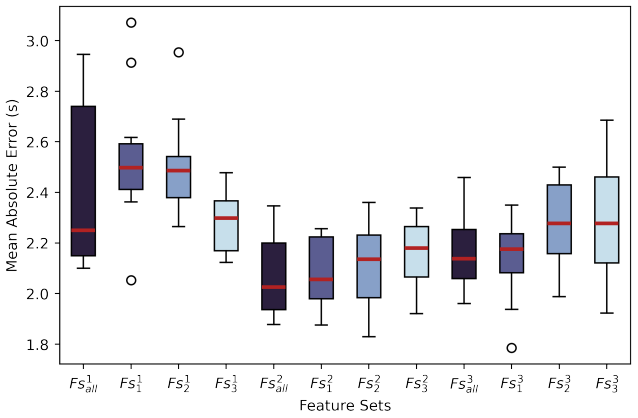


FIGURE 5. Box-and-Whisker Plot of the Mean Absolute Error in Initial Delay Prediction. ***As a reminder, the boxes represent the lower and upper quartiles (25th and 75th percentiles). The red line within the box represents the median value (50th percentile). The whiskers (the two lines outside the box) extend from the quartiles to the last data point within 1.5 times the interquartile range (IQR) of the lower or upper quartile. The circles ∘ represent the outliers, which are the values outside the whiskers and the box.***

delay prediction, the Fs^2_x models exhibit underperformance in predicting resolution, especially Fs^2_{all} which fails to achieve an F1-score of at least 70%. Conversely, the Fs^1_x models demonstrate greater consistency, with F1-scores ranging between 70% and 74%.

For both Fs^2_x and Fs^3_x, the results indicate that reducing the number of features improves the quality of resolution prediction. The peak of this effect is observed in Fs^3_3, which achieves comparable or even superior results compared to Fs^1_3.

3) Stalling events

Figure 8 displays the whisker boxes representing the average F1-score achieved for each feature set in the prediction of stalling events. The difference in F1-scores between models is smaller compared to the F1-score difference observed in the resolution prediction.

The Fs^3_{all} and Fs^2_{all} models exhibit the poorest performance among all the models. Moreover, the feature sets Fs^2_x appear to be less effective in predicting stalling events, whereas the Fs^1_x approach yield better results. In terms of consistency in predicting stalling events, the Fs^y_1 models demonstrate greater stability. In addition, the Fs^1_x models exhibit more consistency compared to the Fs^2_x and Fs^3_x models.

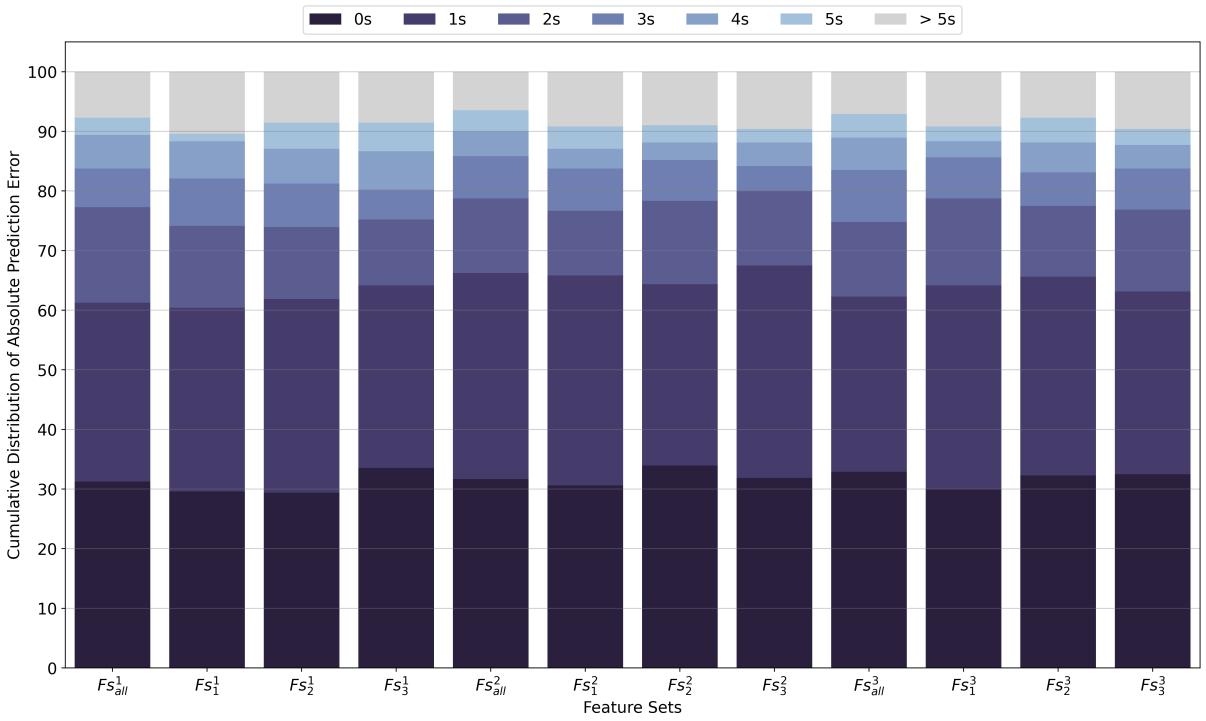


FIGURE 6. Cumulative distribution of absolute prediction error of each feature sets.

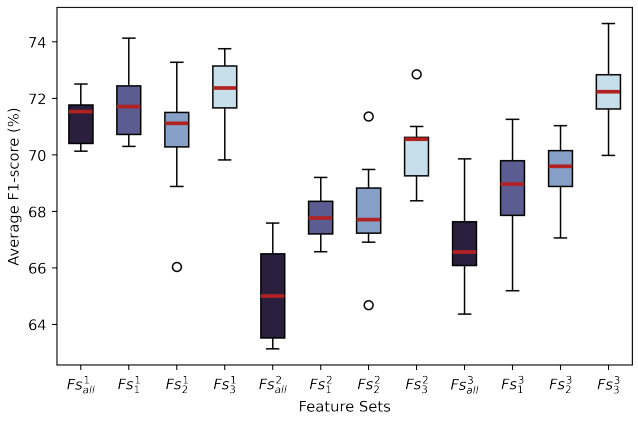


FIGURE 7. Box-and-Whisker Plot of F1-Score in Resolution Prediction.

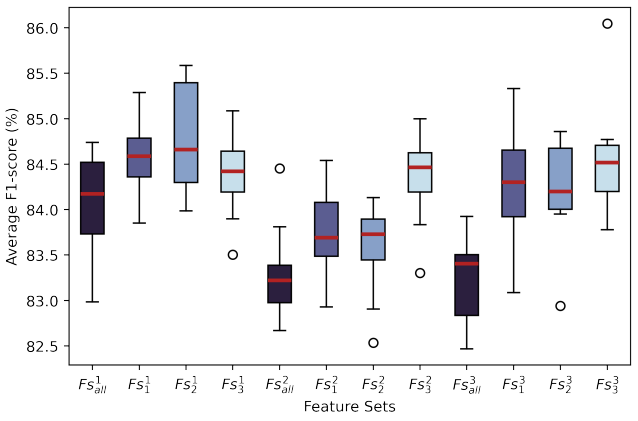


FIGURE 8. Box-and-Whisker Plot of F1-Score in Stalling Event Prediction.

C. INTERPRETATION OF THE RESULTS

This section summarizes the performance results and presents the best feature set approach. Table 6 summarizes the performance metrics obtained for each median-performing model of each Fs_x^y. Noted that the median value shown in the previous graphic displays the medians for each metric individually, rather than the median of all three metrics collectively.

First, let us compare the results based on the number of time windows considered (Fs^y). If we summarize the results, we can observe that Fs_x^1 demonstrates the highest performance for predicting stalling events and resolution, whereas it shows poorer performance for initial delay prediction. On the other hand, Fs_x^2 exhibits the best performance for initial delay prediction, but it performs less effectively in predicting stalling events and resolution. This result is unexpected, as our initial hypothesis was that the feedback connections in the LSTM models should effectively replace the data from the trending window. Fs_x^3 shows good performance for stalling events and initial delay prediction, but it yields relatively weaker results for resolution prediction. Therefore, when considering only the time window used to train the model, the features derived from the last second (Fs_x^1) are the most effective. This is particularly significant when considering that stalling events have the most harmful impact on QoE.

TABLE 6. Prediction performance of the median-performing model for each Fs_x^y.

Time Window Considered	Feature set (nb features)	MAE for Initial Delay	F1-score for Resolution	F1-score for Stalling
Second (Fs_x^1)	Fs_{all}^1 (67)	2.325 s	71.52%	84.77%
	Fs_1^1 (47)	2.913 s	74.13%	85.45%
	Fs_2^1 (40)	2.542 s	71.07%	84.69%
	Fs_3^1 (28)	2.477 s	72.36%	84.98%
Second + Session (Fs_x^2)	Fs_{all}^2 (133)	1.942 s	66.12%	82.57%
	Fs_1^2 (93)	2.229 s	68.86%	83.66%
	Fs_2^2 (79)	2.294 s	67.40%	83.26%
	Fs_3^2 (55)	2.338 s	70.65%	84.51%
Second + Trend + Session (Fs_x^3)	Fs_{all}^3 (199)	2.138 s	66.36%	82.79%
	Fs_1^3 (139)	2.131 s	65.19%	83.04%
	Fs_2^3 (118)	1.988 s	70.01%	84.24%
	Fs_3^3 (82)	2.590 s	72.51%	85.07%

Let us compare the results based on the features considered within each time window (Fs_x). The Fs_{all}^y models show superior performance for initial delay prediction, but they are the worst performing models regarding stalling and resolution predictions. On the other hand, Fs_1^y and Fs_2^y models achieve similar results within the same considered time windows, with no clear advantage or significant weaknesses. As for the Fs_3^y models, they generally yield inferior results for initial delay prediction, except for Fs_3^1, which performs comparably to Fs_2^1 and Fs_1^1; models have better results for resolution prediction and stalling, except for Fs_3^1, which exhibits comparable results to Fs_2^1 and Fs_1^1. Considering these findings, we select Fs_3^y as the preferred model.

Combining these two conclusions, we can determine that the optimal approach for simultaneous prediction of initial delay, resolution, and stalling events is the Fs_3^1 feature set. Furthermore, the Fs_3^1 feature set also has the advantage of having the fewest number of features; this characteristic will facilitate real-time implementation in the future.

Now, let us present the performance of the median-performing Fs_3^1 model. The previous graphic displays the medians for each metric individually, rather than the median of all three metrics collectively. Table 7 provides the prediction performance of the median-performing Fs_3^1 model. In addition, Figures 9 present the confusion matrices for both resolution and stalling event predictions. To summarize the performance of this specific Fs_3^1 model, it achieves a MAE of 2.477,s for predicting the initial delay, an average F1-score of 72.36% for predicting the resolution, and an average F1-score of 84.25% for predicting stalling.

D. COMPARISON WITH PREVIOUS STUDIES

It is important to confirm the relevance of our work compared with the state of the art on other networks. Therefore, this section conducts a comparative analysis, contrasting our performance results with those of the state-of-the-art.

As detailed in one of our previous papers [32], Wassermann et al. [16] employed questionable practices to generate their results. They employed a five-fold cross-validation method to obtain their results. This well-established method involves splitting the dataset into five groups and training a model on four groups while testing the model on the remaining group. This process is repeated for each unique group, resulting in multiple models and their corresponding predictions. By mixing video session data, Wasserman et al. trained their models using time slots from both past and future test datasets. While this approach has improved their model performance on mixed data, it introduces an issue regarding the independence between the training and test datasets. As a result, their models may not generalize well to video sessions that are not mixed with their training dataset. Consequently, we will not make a direct comparison between our findings and theirs in this particular section.

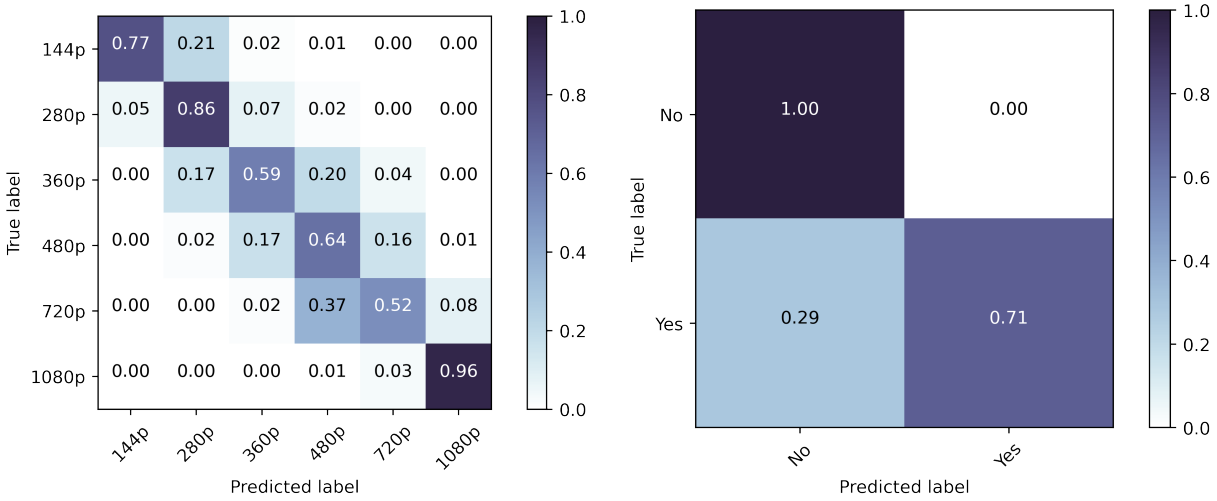
First, let us compare the performance of predicting the initial delay. Nevertheless, it is challenging to make a precise comparison between our performance and that of previous studies, primarily due to the use of different metrics to measure accuracy in predicting initial delay. For instance, Mazhar et al. [12] employed precision and recall as metrics, but we consider these values to be irrelevant as they do not accurately demonstrate the precision of predicting the start of a video. On the other hand, Loh et al. [6] presented MAE as a performance metric.

Loh et al. achieved an MAE of approximately 0.65 s in predicting the initial delay, whereas our models obtain a MAE around 2.48 seconds. The disparity between our conditions may explain this difference. The Loh et al. dataset has a mean initial delay of 2.64 s, whereas our dataset, due to satellite latency, has a mean initial delay more than ten times longer (38.43 seconds). To make a fair comparison of these results, our MAE corresponds to 6% of our average initial delay, whereas theirs corresponds to 24% of their average initial delay.

To compare the predictive capabilities of our model and those of previous studies in terms of resolution, we use the average F1 score obtained. Some studies, such as Malzhar's and Orsolic's [15], predict resolution based on categories such as low or high, making it challenging to directly compare

TABLE 7. Fs^1_3 Prediction Performance

Initial delay MAE		2.477 s	
Resolution	Precision	Recall	F1
144p	89.79%	76.90%	82.84%
240p	78.77%	85.53%	82.01%
360p	54.67%	58.59%	56.56%
480p	53.01%	64.27%	58.10%
720p	65.25%	52.25%	58.04%
1080p	97.09%	96.15%	96.62%
Average	73.10%	72.28%	72.36%
Weighted avg	80.82%	80.14%	80.24%
Stalling Events	Precision	Recall	F1
Yes	99.62%	99.53%	99.57%
No	66.69%	71.34%	68.93%
Average	83.15%	85.43%	84.25%
Weighted avg	99.19%	99.16%	99.17%



(a) Confusion Matrix of Resolution Predictions.

(b) Confusion Matrix of Stalling Event Predictions.

FIGURE 9. Confusion Matrices of Fs^1_3 Model for Resolution and Stalling Event Predictions.

our results with theirs. Nevertheless, most studies employ the same resolution classification as we do. With an average F1 score of approximately 72%, our model's performance falls between that of Gutterman et al. [13] (66.69%, calculated from the data they presented) and Loh et al. (78%). In conclusion, Loh's work demonstrates significantly better results in resolution prediction than our approach.

To compare the performance of the stalling prediction, we also utilize the average F1 score. Our performance is similar to that of other studies, with an average F1 score of 84.25%. This is comparable to Loh et al. study, which achieved 87%. Mazhar's work also demonstrated comparable results for HTTPS flows with an average F1 score of 85.68% (calculated from the data they presented), but had lower results for QUIC flows with an average F1 score of 77.93% (calculated from the data they presented). Shen et al. study, on the other hand, obtained an average F1 score of 59.92% (calculated from the data they presented), which is significantly lower than our results. In conclusion, our model performs relatively well compared to state-of-the-art approaches in predicting stalling events on other networks, although Loh's work shows slightly better performance in this aspect.

To conclude this comparison with the state of the art, although our model does not surpass all other studies, it is important to point out that our results are comparable to other studies that primarily focus on terrestrial networks, which have been extensively studied by the research community. Furthermore, we recall that we are working in the context of a satellite network, which presents unique challenges and characteristics. Therefore, considering the specific context of

satellite networks, these results hold significant value and contribute to the understanding of performance prediction in this particular domain.

VI. REAL-TIME IMPLEMENTATION

Because our goal is to offer a practical solution for ISPs, the implementation of our work in the real world is a crucial part of this work. Ensuring that our models can be applied in real-time is an essential aspect of this endeavor. In addition, it is vital to monitor and address any potential drift in the application process to maintain the accuracy and reliability of our models over time.

One of the major challenges in this real-time implementation is to efficiently process every packet while effectively using the ML model without delaying the overall process.

This section is intentionally designed as a proof of concept and not intended to serve as a benchmark for various implementations. Instead, the focus is to showcase the feasibility and viability of the proposed approach through a basic implementation.

In this section, we employ the median-performing Fs_3^1 model, discussed in Section V-C where its performance is presented. As a reminder, this model only uses 28 features, which represents the lowest number of features considered in our analysis.

First, this section provides an overview of the implementation process and discusses potential bottlenecks that may impede the efficiency of the system. Second, this section presents the monitored information within these bottlenecks and offers a comparison between the predictions and the ground truth.

A. IMPLEMENTATION OVERVIEW

This section discusses the implementation choices, specifically how we processed the previously collected dataset. During the processing of the collected dataset, as described in Section III-C, we intentionally processed each packet separately, treating them as received in real-time. Furthermore, we ensure that the feature set is dynamically updated in real-time for each packet. Therefore, the processing of packets for real-time implementation poses no issue for us, as we have already implemented the necessary procedures during the dataset processing phase.

The implementation of our system is relatively simple, consisting of three parallel threads:

1) *Packet Sniffer*: this thread collects IP packets and places them in the queue for the second thread. By assigning a dedicated thread for packet collection, we can guarantee that all packets are properly gathered;
2) *Packet Process*: the second thread processes the packets, examining their source and destination IP addresses, size, and timestamp. It processes the packets individually and sends the data to the third thread after a one-second interval;
3) *ML Prediction*: this thread normalizes the data received from the second thread and predicts the KQIs.

From this implementation, two potential bottlenecks emerge. The first potential bottleneck arises at the entrance to the second thread, where a queue of packets may form if the packet processing speed is cannot keep up with the incoming packet rate. The second potential bottleneck lies in the ML processing phase, particularly if it exceeds the allocated one-second timeframe.

B. MONITORING AND ANALYSIS OF REAL-TIME IMPLEMENTATION

This section presents and analyzes the monitored results of the implementation.

We maintain the topology as presented in Section III-A and illustrated in Figure 1. Furthermore, the prediction process occurs at the same location where the traffic data are collected, specifically on the router. It is worth noting that the implementation is performed on a virtual machine with two CPU cores of 2.6 GHz, 4 GB of RAM, and no GPU.

First, this section examines the packet queuing aspect at the entrance of the second thread. Second, it assesses the processing time of machine learning predictions in the third thread. Finally, it presents examples of curves that illustrate the relationship between the predictions and the collected truth.

1) Packet Queuing

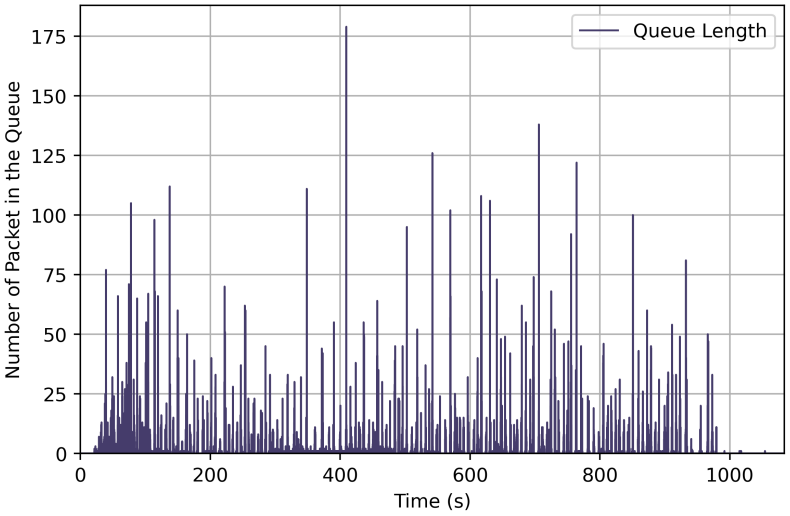
To verify the queue length, we monitor it at regular intervals of 10 ms during a video session. To stress the system, the test uses a capacity of 12 Mbps, ensuring no loss on the link and no congestion. In addition, we select a video with a frame rate of 30 fps, which is long enough to observe different phases.

Figure 10(a) presents the packet queue behavior throughout the entire video session. From the figure, we can observe that at the start of the video session, the traffic is notably high. Around the 1000th second mark, the traffic comes to a halt as the video concludes, causing the packet flow to stop and subsequently filling the buffer. Notably, a peak is observed around the 400th second, with the queue size exceeding 175 packets.

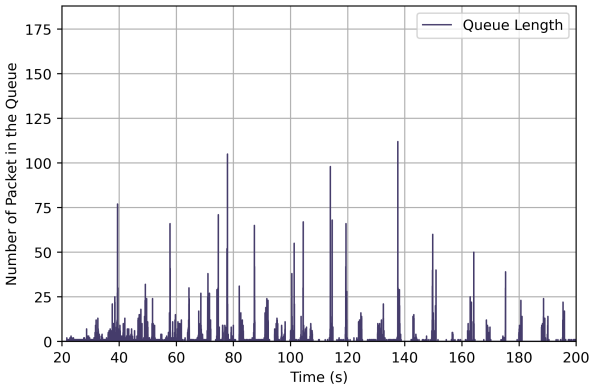
Figure 10(b) provides a zoomed-in view of the beginning of the video, where the traffic density is the highest. Contrary to the initial assumption derived from the unzoomed figure, this new figure reveals that the number of packets in the queue occasionally falls below a 10-packet length. Specifically, we observe that the queue size experiences periodic growth, reaching a range of 75–100 packets within a 10 ms interval; however, it quickly decreases in the subsequent 10 ms interval.

Figure 10(c) provides a zoomed-in view of the highest peak observed during the video session. From this figure, it is evident that the peak, surpassing 175 packets, is rapidly absorbed and does not persist for more than 10 ms.

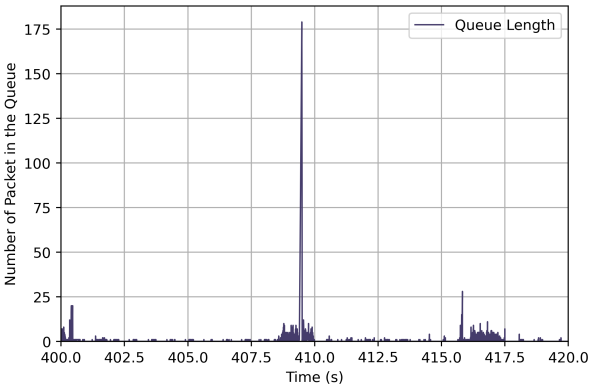
Overall, these figures demonstrate that the packet rate can experience significant increases; however, despite these challenges, the implemented packet processing mechanism exhibits the capability to efficiently manage and absorb any



(a) Packet Queue over Time during a Video Session.



(b) Zoomed-in View of the Beginning of the Video Session.



(c) Zoomed-in View of the Peak of the Packet Queue.

FIGURE 10. Packet Queue Analysis during a Video Session: Overall Dynamics, Session Start, and Peak Behavior.

peaks or dense traffic. This demonstrates the effectiveness of the implemented packet processing approach in maintaining a low real-time drift of the system.

2) Machine Learning Prediction Time

Following the methodology described in Section VI-B1, we monitor the ML prediction time.

Figure 11 presents the CDF of the prediction time for the ML model. The data are collected from four video sessions, consisting of a total of 4111 predictions, with each prediction corresponding to a one-second time slot. To improve data visualization, a logarithmic scale is applied to the x-axis. The prediction time exhibits a notable range of variance, spanning from a minimum of 0.5 ms to a maximum of 342 ms, with a standard deviation of 9.7 ms. From the data in the figure, it is clear that 20% of the predictions are completed within 1 ms, while 50% of the predictions are realized within 1.6 ms. Moreover, in approximately 90% of the cases, the predictions are completed within 10 ms; however, it is worth noting that a small fraction of predictions, less than 1%, exceed 100 ms.

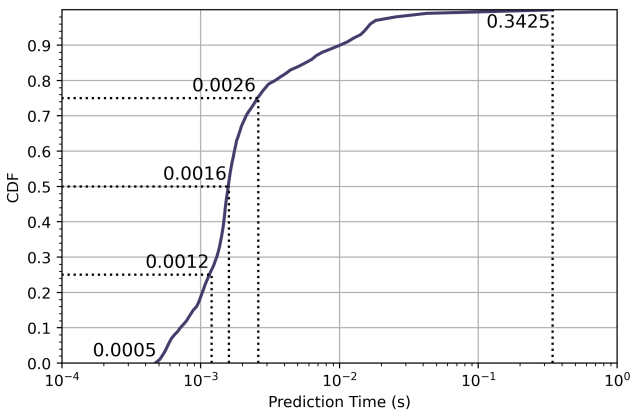


FIGURE 11. CDF of ML Prediction Time during Video Sessions (Logarithmic Scale Y-Axis).

Even in rare instances where the prediction time reaches its longest duration of 342 ms, its impact on meeting the schedule for subsequent predictions remains insignificant. Neverthe-

less, if we want to implement this for multiple clients on the same connection, the process may get out of schedule when there are many concurrent predictions. A potential solution to address potential obstructions to the ML prediction process is to allocate a dedicated CPU specifically for ML predictions. By doing so, we can ensure that no other processes interfere with or hinder the execution of the ML algorithms.

3) Checking the Quality of Prediction

The performance prediction aspects are addressed in Section V; however, in this section, we compare the real-time monitored ground truth with our predictions. As a result, this section presents a comparison between the ground truth and prediction for a video session conducted over a satellite link with a capacity of 12 Mbps. Figures 12 illustrate the comparisons between the monitored ground truth and the prediction during a video session over a satellite link with a capacity of 12 Mbps. No congestion or loss is introduced in this scenario, and as a result, there are no figures representing stalling events because no such events occurred or were predicted.

Figure 12(a) and Figure 12(b) provide a comparison between the start of the video and the corresponding prediction of the start of the video. The second figure is a zoomed-in view of the first 30 s, focusing on the crucial moment of the video playback initiation. Upon analyzing these figures, it becomes evident that the beginning of the video is accurately predicted, with no delay. The predicted start time aligns perfectly with the actual start time of the video playback.

Figure 12(c) displays the video resolution and predicted video resolutions throughout the video session. Meanwhile, Figure 12(d) zooms in on the first 100 s of the data, which encompasses the resolution switches. The predicted resolutions are not considered relevant before the prediction of the start of video playback. These figures indicate that there are no major issues with predicting the $1080p$ resolution. Nevertheless, the challenge lies in accurately predicting the timing of the resolution switches. In these cases, resolutions are overpredicted before actual changes occur.

VII. LIMITATIONS

This section outlines the limitations of the experimental approaches employed in this study to evaluate and contextualize the results effectively.

In this paper, we construct our dataset and the models over emulated satellite link use cases for a specific application, i.e., YouTube. Nevertheless, each stored video streaming application has its own Adaptive Bit Rate (ABR) streaming algorithms that affect the displayed resolution, initial delay, and stalling event occurrences. Furthermore, each application can use different rules for video encoding. For example, YouTube offers encoding recommendations but leaves some liberties to video creators on the matter. Netflix re-encodes available videos, to optimize throughput and QoE for users. Therefore, if we expect that it is possible to apply the same feature processes to other stored video streaming applications, such as Netflix, Amazon Prime, and Disney+, specific datasets tailored to those applications need to be built.

Additionally, the dataset used to create the model consists of 40 videos with a maximum resolution of $1080p$ and a frame rate per second between 24 and 30. Nevertheless, YouTube currently offers videos with resolutions up to $2160p$ (also known as 4K). Moreover, YouTube is experimenting with a "$1080p$ Premium" resolution for paying users, providing an "enhanced" $1080p$ version, which is encoded with a higher bit rate, resulting in a better QoE [33]. Therefore, to adapt the model, we would need to collect data with a wider range of video characteristics.

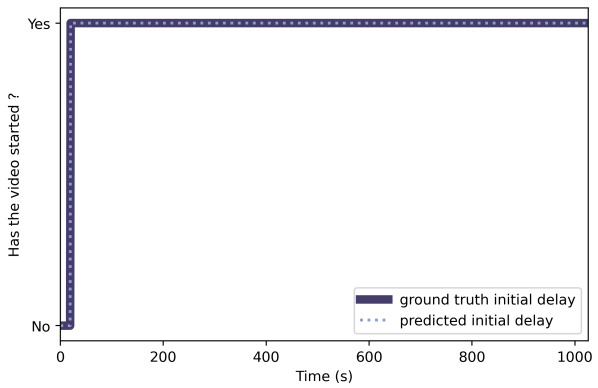
Furthermore, the dataset we collected was based on specific scenarios, which may not cover all possible use cases. For example, a change in the maximum packet size would significantly impact most features, thus affecting the prediction. Similarly, a very different available throughput from the considered scenarios could also impact the prediction quality. Finally, any evolution in the YouTube encoding policy or their ABR algorithm would require collecting another dataset and training new models. Therefore, to enable models to adapt to changes over time, a data collection system must be implemented and run to frequently update the model.

VIII. CONCLUSIONS

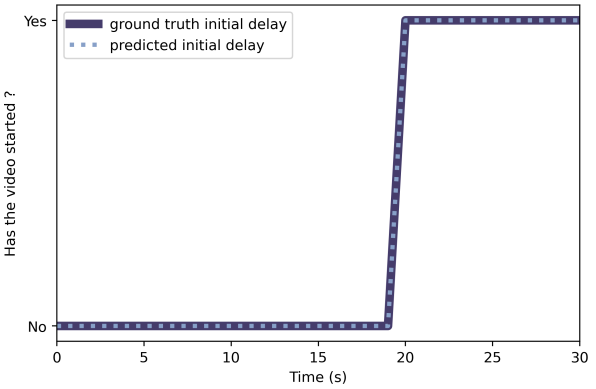
This study aims to predict QoE KQIs, i.e., initial delay, resolution, and stalling event occurrences, of YouTube sessions over a geosynchronous satellite network.

For this purpose, we monitor 2,400 YouTube video sessions and the resulting packet traffic. The collected dataset is available as an open source to the research community. We compare two ML models, RF and LSTM, trained to predict the QoE KQIs based on packet traffic data using an adjusted tried-and-tested approach from Wassermann et al. study [16]. Next, we focus on reducing the required feature set to optimize efficiency, considering both the prediction performance and the computation time.

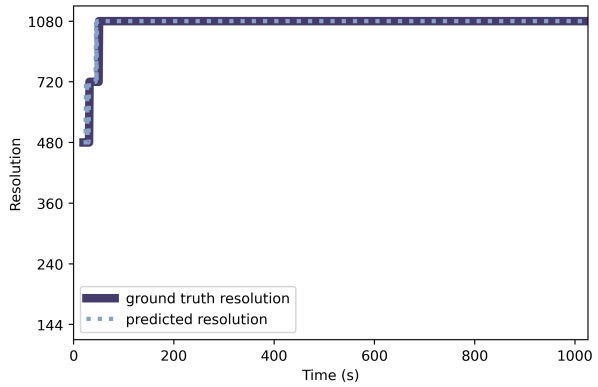
The results show that the LSTM approach significantly outperforms the RF models, specifically in the prediction of resolution and stalling events. Although the RF is a less complex solution with a shorter computation time compared to the LSTM, the difference in performance is too important to be balanced by this gain. Regarding the optimization of the feature sets for the LSTM training, we demonstrate that we can eliminate multiple irrelevant features from Wassermann's approach. First, the results show that models have better performances when using only features from the last second, and not from the three considered time windows. Second, within the time windows, it is possible to significantly reduce the feature considered and achieve relatively the same level of performance. Using the smallest feature set, with a reduction in the number of features from 199 to only 28, we achieved performance comparable to the state of the art on terrestrial networks, demonstrating its applicability and effectiveness in the context of GEO satellite networks.



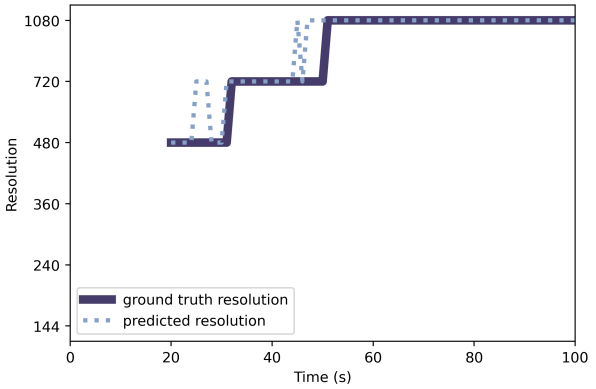
(a) Real-Time Initial Delay Prediction vs. Ground Truth.



(b) Zoomed-in View of Real-Time Initial Delay Prediction vs. Ground Truth.



(c) Real-Time Resolution Prediction vs. Ground Truth.



(d) Zoomed-in View of Real-Time Resolution Prediction vs. Ground Truth.

FIGURE 12. Real-Time Prediction vs. Ground Truth of a video session over a 12 Mbps Satellite Link.

This study also tackles the real-time implementation of our model, and the potential risks associated with ensuring smooth real-time functioning. After identifying two potential bottlenecks, we show that they are not significant issues for real-time functioning.

In future work, we wish to apply this approach to other popular applications, such as Netflix, Twitch, and Teams. Furthermore, another objective is to combine our models with a reinforcement learning agent capable of dynamically applying actions to enhance the QoE for end users.

ACKNOWLEDGMENT

The authors wish to thank Victor Perrier for his help.

APPENDIX. USED FEATURES AND THEIR PRESENCE IN REDUCED FEATURE SETS

REFERENCES

[1] Sandvine, "The global internet phenomena report covid-19 spotlight," 2020. [Online]. Available: https://www.sandvine.com/covid-internet-spotlight-report

[2] K. Bouraqia, E. Sabir, M. Sadik, and L. Ladid, "Quality of experience for streaming services: Measurements, challenges and insights." [Online]. Available: https://ieeexplore.ieee.org/document/8954623/

[3] T. Pecorella, L. S. Ronga, F. Chiti, S. Jayousi, and L. Franck, "Emergency satellite communications: research and standardization activities," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 170–177, 2015.

[4] B. Tauran, E. Lochin, J. Lacan, F. Arnal, M. Gineste, and N. Kuhn, "Scheduling flows over LEO constellations on LMS channels," *International Journal of Satellite Communications and Networking*, 2020. [Online]. Available: https://hal.science/hal-02535708

[5] A. M. Cipriano, P. Gagneur, G. Vivier, and S. Sezginer, "Overview of arq and harq in beyond 3g systems," in *2010 IEEE 21st International Symposium on Personal, Indoor and Mobile Radio Communications Workshops*, 2010, pp. 424–429.

[6] F. Loh, F. Poignée, F. Wamser, F. Leidinger, and T. Hoßfeld, "Uplink vs. downlink: Machine learning-based quality prediction for HTTP adaptive video streaming." [Online]. Available: https://www.mdpi.com/1424-8220/21/12/4172

[7] B. Goehry, H. Yan, Y. Goude, P. Massart, and J.-M. Poggi, "Random Forests for Time Series," Feb. 2021, working paper or preprint. [Online]. Available: https://hal.science/hal-03129751

[8] K. Brunnström, S. A. Beker, K. de Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M.-C. Larabi, B. Lawlor, P. Le Callet, S. Möller, F. Pereira, M. Pereira, A. Perkis, J. Pibernik, A. Pinheiro, A. Raake, P. Reichl, U. Reiter, R. Schatz, P. Schelkens, L. Skorin-Kapov, D. Strohmeier, C. Timmerer, M. Varela, I. Wechsung, J. You, and A. Zgank, "Qualinet White Paper on Definitions of Quality of Experience," Mar. 2013, qualinet White Paper on Definitions of Quality of Experience Output from the fifth Qualinet meeting, Novi Sad, March 12, 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00977812

[9] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," vol. 18, no. 1. [Online]. Available: http://ieeexplore.ieee.org/document/7035000/

[10] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," vol. 17, no. 1. [Online]. Available: https://ieeexplore.ieee.org/document/6913491/

[11] T. e. a. Hossfeld, ''Initial delay vs. interruptions: Between the devil and the deep blue sea.'' [Online]. Available: http://ieeexplore.ieee.org/document/6263849/
[12] M. H. Mazhar and Z. Shafiq, ''Real-time video quality of experience monitoring for https and quic,'' in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 1331–1339.
[13] C. e. a. Gutterman, ''Requet: real-time QoE detection for encrypted YouTube traffic.'' [Online]. Available: https://dl.acm.org/doi/10.1145/3304109.3306226
[14] F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, ''Inferring Streaming Video Quality from Encrypted Traffic: Practical Models and Deployment Experience,'' *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–25, Dec. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3366704
[15] I. Orsolic and L. Skorin-Kapov, ''A framework for in-network qoe monitoring of encrypted video streaming,'' *IEEE Access*, vol. 8, pp. 74 691–74 706, 2020.
[16] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, ''ViCrypt to the rescue: Real-time, machine-learning-driven video-QoE monitoring for encrypted streaming traffic,'' vol. 17, no. 4. [Online]. Available: https://ieeexplore.ieee.org/document/9250645/
[17] M. Shen, J. Zhang, K. Xu, L. Zhu, J. Liu, and X. Du, ''Deepqoe: Real-time measurement of video qoe from encrypted traffic with deep learning,'' in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, 2020, pp. 1–10.
[18] ''Opensand,'' https://www.opensand.org/.
[19] E. Dubois *et al.*, ''Opensand, an open source satcom emulator.'' Kaconf, 2017.
[20] A. Auger, E. Lochin, and N. Kuhn, ''Making Trustable Satellite Experiments: an Application to a VoIP Scenario,'' in *89th IEEE Vehicular Technology Conference*. Kuala Lumpur, Malaysia: IEEE, Apr. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02191751
[21] ''Openbach,'' https://www.openbach.org/.
[22] ''Satboost add-on,'' https://github.com/CNES/satboost/.
[23] ''Geckodriver,'' https://github.com/mozilla/geckodriver/.
[24] ''ublock,'' https://github.com/gorhill/uBlock.
[25] G. Romain, P. David, P. Guillaume, and K. Nicolas, ''Recommendations on using VPN over SATCOM,'' *CoRR*, vol. abs/2111.04586, 2021. [Online]. Available: https://arxiv.org/abs/2111.04586
[26] N. Kuhn, F. Simo, D. Pradas, and E. Stephan, ''Evaluating BDP FRAME extension for QUIC,'' *CoRR*, vol. abs/2112.05450, 2021. [Online]. Available: https://arxiv.org/abs/2112.05450
[27] ''iperf3,'' https://software.es.net/iperf/.
[28] N. Kuhn, F. Michel, L. Thomas, E. Dubois, E. Lochin, F. Simo, and D. Pradas, ''Quic: Opportunities and threats in satcom,'' *International Journal of Satellite Communications and Networking*, vol. 40, no. 6, pp. 379–391, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sat.1432
[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, ''Scikit-learn: Machine learning in Python,'' *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[30] ''Scikit-learn - randomforestclassifier,'' hhttps://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.
[31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, ''Automatic differentiation in pytorch,'' in *NIPS-W*, 2017.
[32] M. Petrou, D. Pradas, M. Royer, and E. Lochin, ''Forecasting youtube qoe over satcom,'' in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, 2023, pp. 1–5.
[33] M. Clark, ''Youtube says it isn't messing with 1080p — '1080p premium' is higher-bitrate,'' [Online; accessed July-2023]. [Online]. Available: https://www.theverge.com/2023/2/23/23612647/youtube-1080p-premium-subscription-bitrate

MATTHIEU PETROU received his MSc degree in embedded system engineering from University of Versailles Saint-Quentin-en-Yvelines in 2019. He is currently a PhD student at ISAE-SUPAERO, under the supervision of Emmanuel Lochin, David Pradas and Mickaël Royer. His current research interests include quality of experience over satellite communication, transport protocol, and machine learning in network.

DAVID PRADAS obtained his MSc degree in telecommunications engineering from UPC in 2006, and he owns a PhD (2011) from ISAE-SUPAERO and from UAB, co-funded by CNES and UAB. During his PhD, he focused on cross-layer designs and methodologies for broadband satellite networks and participated in several international research projects (MOVISAT, Sat-NEx I & II). He is now the R&D manager of network/telecom activities at Viveris Technologies. In the frame of his activity in Viveris (since 2012), he has contributed to OpenSAND and OpenBACH and has led more than 30 R&T (Research & Technology) projects for the CNES and other industrial partners, and is currently involved in the HE COMMECT project. His current research interests are the optimization of higher layers on hybrid satellite/5G systems and new satellite constellations.

MICKAËL ROYER obtained his Masters from ENAC in aeronautical engineering in 2004 (telecommunication specialisation). He received his PhD from the Paul Sabatier University - Toulouse III in May 2016. Since, he held an associate professor position in the ENAC Research Laboratory, more precisely in the Telecommunication team. In parallel, he is at the head of the science and engineering department at ENAC.

EMMANUEL LOCHIN received his PhD from the LIP6 laboratory of Pierre and Marie Curie University - Paris VI in December 2004 and the Habilitation Thesis (Habilitation à Diriger des Recherches) in October 2011 from Institut National Polytechnique de Toulouse (INPT). From July 2005 to August 2007, he held a researcher position in the Networks and Pervasive Computing research program at National ICT Australia, Sydney. He then held a full professor position at ISAE-SUPAERO from September 2007 to March 2020 and has co-founded SPEERYT in July 2018 to stimulate the development and diffusion of an on-the-fly coding scheme named Tetrys. Before SPEERYT, this technology was transferred by TTT to a world leader in Internet content distribution. He is now full professor at ENAC since April 2020 and is also member of TéSA laboratory and computer networking expert in the TeSA scientific committee.

TABLE 8. Overview of the features used in a single time window and their presence in each reduced version of our tests

Feature Category	Feature name	Fs^y_{all}	Fs^y_1	Fs^y_2	Fs^y_3
Time Slot Number	Time Slot Number	✓	✓	✓	✓
Packet Count	Total Number of Packets	✓	✓	✓	✓
	Number of Uploaded Packets	✓	✓	✓	✓
	Number of Downloaded Packets	✓	✓	✓	✓
Byte Count	Total of Bytes	✓	✓	✓	✓
	Total of Uploaded Bytes	✓	✓	✓	✓
	Total of Downloaded Bytes	✓	✓	✓	✓
Ratios	Uploaded Packet Ratio	✓	✗	✗	✗
	Downloaded Packet Ratio	✓	✗	✗	✗
	Uploaded Byte Ratio	✓	✗	✗	✗
	Downloaded Byte Ratio	✓	✗	✗	✗
Arrival Time	Time of First Packet	✓	✗	✗	✗
	Time of Last Packet	✓	✗	✗	✗
	Time of First Uploaded Packet	✓	✗	✗	✗
	Time of Last Uploaded Packet	✓	✗	✗	✗
	Time of First Downloaded Packet	✓	✗	✗	✗
	Time of Last Downloaded Packet	✓	✗	✗	✗
Burst	Throughput of Burst	✓	✗	✗	✗
	Throughput of Uploaded Burst	✓	✗	✗	✗
	Throughput of Downloaded Burst	✓	✗	✗	✗
	Length of Burst	✓	✗	✗	✗
	Length of Uploaded Burst	✓	✗	✗	✗
	Length of Downloaded Burst	✓	✗	✗	✗
Covariance of Cumulative Upload Traffic over Time	Temporal Variance Upload	✓	✓	✓	✗
	Covariance Time Size Upload	✓	✓	✓	✗
	Mean Cumulative Size Upload	✓	✓	✓	✗
	Mean Time Upload	✓	✓	✓	✗
	Slope Upload	✓	✓	✓	✗
	Intercept Upload	✓	✓	✓	✗
Covariance of Cumulative Download Traffic over Time	Temporal Variance Download	✓	✓	✓	✗
	Covariance Time Size Download	✓	✓	✓	✗
	Mean Cumulative Size Download	✓	✓	✓	✗
	Mean Time Download	✓	✓	✓	✗
	Slope Download	✓	✓	✓	✗
	Intercept Download	✓	✓	✓	✗
Uploaded Packet IAT Statistics	Mean Upload IAT	✓	✓	✓	✓
	Min Upload IAT	✓	✓	✓	✓
	Max Upload IAT	✓	✓	✓	✓
	Variance Upload IAT	✓	✗	✗	✗
	Standard Deviation Upload IAT	✓	✓	✓	✓
	Covariance Upload IAT	✓	✓	✓	✓
	Skewness Upload IAT	✓	✓	✓	✓
	Kurtosis Upload IAT	✓	✓	✓	✓
Uploaded Packet Byte Size Statistics	Mean Upload Byte Size	✓	✓	✓	✓
	Min Upload Byte Size	✓	✓	✓	✓
	Max Upload Byte Size	✓	✓	✓	✓
	Variance Upload Byte Size	✓	✗	✗	✗
	Standard Deviation Upload Byte Size	✓	✓	✓	✓
	Covariance Upload Byte Size	✓	✓	✓	✓
	Skewness Upload Byte Size	✓	✓	✓	✓
	Kurtosis Upload Byte Size	✓	✓	✓	✓
Downloaded Packet IAT Statistics	Mean Download IAT	✓	✓	✓	✓
	Min Download IAT	✓	✓	✓	✓
	Max Download IAT	✓	✓	✓	✓
	Variance Download IAT	✓	✗	✗	✗
	Standard Deviation Download IAT	✓	✓	✓	✓
	Covariance Download IAT	✓	✓	✓	✓
	Skewness Download IAT	✓	✓	✓	✓
	Kurtosis Download IAT	✓	✓	✓	✓
Downloaded Packet Byte Size Statistics	Mean Download Byte Size	✓	✓	✗	✗
	Min Download Byte Size	✓	✓	✗	✗
	Max Download Byte Size	✓	✓	✗	✗
	Variance Download Byte Size	✓	✗	✗	✗
	Standard Deviation Download Byte Size	✓	✓	✗	✗
	Covariance Download Byte Size	✓	✓	✗	✗
	Skewness Download Byte Size	✓	✓	✗	✗
	Kurtosis Download Byte Size	✓	✓	✗	✗