QoE Prediction of Encrypted Video Traffic for Interactive and Impatient Streaming Users

Somiya Kapoor*, David Hasselquist*†, Ethan Witwer*, Mikael Asplund*, Niklas Carlsson*
*Linköping University, Sweden
†Sectra Communications, Sweden

***Abstract*—Encrypted video streaming limits network operators' visibility into user behavior and complicates Quality of Experience (QoE) estimation. While existing QoE prediction models largely assume smooth, linear playback, real-world video-on-demand (VoD) viewing is often interactive: users frequently jump forward or backwards to skip or replay content. Such interactions invalidate buffered data, trigger bursty segment requests, and introduce non-stationary traffic patterns that violate the assumptions underlying prior models. In this paper, we present the first systematic study of QoE prediction of encrypted VoD traffic that accounts for user jumping behavior. Using controlled experiments with time-varying bandwidth and explicit jump events, we show that a state-of-the-art QoE classifier trained on smooth playback degrades sharply once jumps occur, with accuracy collapsing under frequent interaction. We further demonstrate that jumping behavior leaves a clear fingerprint in encrypted traffic: jump and non-jump sessions are perfectly distinguishable, and interaction intensity can be inferred reliably across bandwidth regimes. Leveraging these insights, we introduce PB-QoE, a lightweight and interpretable jump-aware QoE prediction pipeline that combines continuous regression, trained on non-jumping data, with an interaction-dependent penalty. Our evaluation across bandwidth conditions, jump frequencies and directions, and video categories shows that PB-QoE consistently outperforms classification- and regression-only baselines, enabling robust QoE estimation from encrypted traffic even under highly interactive playback.**

I. Introduction

Video streaming, particularly Video on Demand (VoD), continues to dominate Internet traffic across domains such as entertainment, gaming, news, and education. At the same time, viewing behavior is becoming increasingly selective. Rather than watching videos from start to finish, users frequently skip uninteresting portions or jump directly to relevant segments, especially for longer-form content. This trend, driven by time-constrained, goal-oriented viewing habits especially among younger audiences, is now common across VoD platforms.

Such jumping behavior fundamentally alters playback dynamics. When users move the playback position, previously buffered content may become invalid, triggering bursty segment requests, adaptive bitrate (ABR) reconfiguration, and in some cases playback stalls or transient quality degradation. Prior studies have shown that frequent jumps can significantly degrade perceived Quality of Experience (QoE). Importantly, the resulting traffic patterns differ markedly from those generated during smooth, uninterrupted playback.

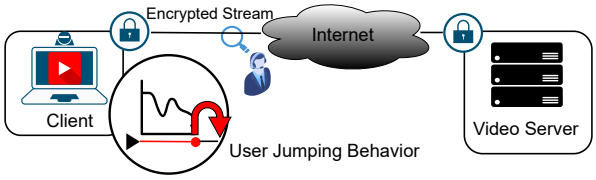


Fig. 1: Encrypted video fingerprinting QoE model.

Accurately estimating QoE under these conditions is increasingly important for network operators and content providers, as QoE estimation underpins performance monitoring, resource allocation, and optimization. However, this task is already challenging due to widespread end-to-end encryption, which hides application-layer signals. While a growing body of work has demonstrated that QoE can be inferred from encrypted video traffic (as illustrated in Figure 1) using traffic fingerprints or learning-based models, these approaches are almost exclusively evaluated under uninterrupted, linear playback. Separately, studies of jumping behavior quantify its impact on QoE but do not address QoE prediction under interactive playback. As a result, it is unclear whether existing QoE prediction models generalize to the non-linear, interaction-driven traffic patterns common in modern VoD services.

In this paper, we address this gap by studying QoE prediction for encrypted VoD traffic under user jumping behavior. We show that state-of-the-art QoE classifiers trained on smooth playback degrade sharply once jumps occur, revealing a clear generalization gap. At the same time, we find that jumping behavior leaves a strong fingerprint in encrypted traffic, enabling reliable detection of both jump events and interaction intensity. Leveraging these insights, we introduce PB-QoE, a lightweight and interpretable jump-aware QoE prediction pipeline, that provides robustness to both network variability and user-driven interaction, and make the following contributions:

- We present the first study of QoE prediction of encrypted video traffic that accounts for jumping behavior.
- We show that state-of-the-art QoE classifiers trained on smooth playback degrade sharply when jumps occur.
- We show that jump events and interaction intensity can be reliably inferred from encrypted video traffic across bandwidth regimes.
- We introduce PB-QoE, a jump-aware QoE prediction pipeline that combines regression with interaction-dependent penalty to achieve robustness under jumping.

- We validate PB-QoE through a comprehensive robustness evaluation across bandwidth conditions, jump frequencies and directions, and video categories.

Overall, PB-QoE enables robust and accurate QoE estimation from encrypted traffic under highly interactive playback.

Outline: After presenting background, related work (Section II), and the dataset collection (Section III), we introduce QoE metrics and analyze the impact of user jumping behavior (Section IV). Next, we present the RF-based QoE classification baseline (Section V) followed by our jump-aware QoE prediction framework (Section VI). Finally, we report the evaluation results (Section VII) and conclude the paper (Section VIII).

II. Background And Related Work

A. Adaptive Bitrate Streaming

Dynamic Adaptive Streaming over HTTP (DASH) [1] is the standard mechanism behind most modern HTTP video services, including large platforms like YouTube [2] and Netflix [3]. In DASH, a video player downloads videos in small segments and can select the quality of each segment based on network conditions, thereby optimizing QoE in terms of average playback quality, quality switch frequency, and rebuffering time. However, user behaviors, such as jumping (seeking), introduce sudden changes in playback position, disrupting buffer management and challenging the adaptive bitrate (ABR) algorithms used by DASH, making it harder to maintain smooth playback. Moreover, end-to-end encryption prevents network operators from seeing application-level data, making it difficult to assess or adapt to such user behavior and accurately estimate QoE.

B. Video QoE Prediction

Several works have investigated the prediction of the QoE of video traffic. Early approaches [4]–[6] typically extract hand-picked network- or transport-layer features and apply machine learning or heuristic rules to estimate either a unified QoE score or individual QoE components such as average quality, quality switching, or stall severity. Some methods operate at the session level, while others support near real-time inference with short windows, often at the cost of reduced accuracy. Certain works use event-driven buffer emulation and a classification framework to estimate client buffer occupancy and predict stall risk from packet-level traces [7].

Recent work has shifted towards deep learning-based QoE estimation, often using LSTMs and Random Forests [8], [9]. CNN-based approaches have also been explored. Shen et al. [10] predict startup delay, rebuffering, and quality from traffic features, while Mazhar and Shafiq [11] infer encrypted video QoE metrics in real time for YouTube streams using network/transport-layer features and decision-tree models. A different line of work builds fingerprint databases to enable very accurate real-time quality identification [12], but these require operators to precompute fingerprints for many videos, which limits practicality.

Our work differs from prior systems in two key ways: existing approaches are typically evaluated under regular, uninterrupted playback and are rarely stress-tested against disruptive user interactions or diverse bandwidth conditions. In contrast, we evaluate QoE prediction on traces collected under time-varying bandwidths at multiple scales, with controlled jump events introduced to explicitly evaluate jumping behavior's impact on QoE prediction and assess generalization beyond models' non-jumping training regimes.

The work most closely related to ours is Kapoor et al. [13], who predict QoE from encrypted video traffic by leveraging the Video-Adapted Robust Fingerprinting model (vRF). Their approach builds on Carlson et al. [14], who first adapted vRF for video identification under fluctuating bandwidth and latency. vRF is in turn a video adaptation of Robust Fingerprinting (RF) [15], originally proposed for website fingerprinting [16]–[18]. This progression shows that vRF features can remain stable across network conditions and are expressive enough to recover video-level properties without application-layer access. However, Carlson et al. [14] focus on live streaming (with small buffers) and non-interactive playback without disruptive user actions, such as jumping. In contrast, our VoD setting includes frequent user interactions that violate these steady-state assumptions and can substantially degrade the accuracy of RF-based QoE prediction.

C. User Jumping Behavior and Its Impact on QoE

Users of VoD and HTTP adaptive streaming services do not always watch videos in a smooth, linear manner. Instead, they often move the playback position forward or backwards to skip uninteresting content or replay specific segments, a behavior commonly referred to as jumping, seeking, or interactive playback [19], [20]. Such actions fundamentally alter the sequence of requested video segments, as the client no longer fetches segments in temporal order.

Several studies have analyzed the prevalence and characteristics of jumping behavior and its impact on playback performance [21]–[23]. These works establish that users frequently jump during playback and that such interactions can degrade perceived quality; e.g., by increasing rebuffering or causing transient quality drops [23]. However, while they quantify the impact of jumping on QoE, they focus on measuring its effects rather than predicting QoE under jumping conditions.

To the best of our knowledge, no prior work has addressed QoE prediction under jumping conditions using encrypted traffic alone. Our work bridges this gap by explicitly modeling user jumping behavior and evaluating how QoE prediction models generalize under interactive playback.

D. QoE Optimizations

Alongside prediction, several network-level schemes aim to improve QoE: server-side assistance [24], [25] and edge-based mechanisms [26]–[28] can help but either add load to the video platform or depend on cooperative/modified clients. Other studies propose temporary bandwidth boosts during startup or after stalls [29] or heuristic/deep-learning policies to balance QoE and fairness [30]. QoE-aware caching has also been suggested to raise performance at scale [14], [31]–[33].

Overall, many of these designs assume visibility into client state or richer metrics than encrypted traffic alone exposes, so they could benefit from more accurate traffic-based QoE estimation like our proposed PB-QoE solution (Section VI).

III. Dataset Collection

This section describes the data collection pipeline used to construct our datasets. The goal of this collection process is to obtain controlled, repeatable, and QoE-labeled network traces that reflect both traditional (linear) playback and non-linear playback under user interactions. To do so, we combine (1) in-browser playback using dash.js, (2) controlled server-side content hosting and encoding, and (3) bandwidth- and interaction-driven session simulations that recreate different bandwidth conditions and user-interaction intensities. Unlike public platform scraping or passive capture, our approach enables fine-grained control over resolution, segment boundaries, bandwidth conditions, and jump timing, ensuring that each trace has a known and reproducible ground truth QoE profile.

Collection Setup: Our data collection framework follows prior work on video fingerprinting [14], [34], employing two interconnected physical machines that act as a client and server, respectively. Using the Selenium [35] browser automation tool, we launch a Chrome session isolated within a network namespace on the client. The client uses the dash.js [36] implementation of MPEG-DASH to request a video stream served by nginx [37] within a corresponding network namespace on the server. Each namespace is capped at 100/100 Mbps throughput and adds a delay of 5 ms per packet, resulting in a round-trip time of slightly over 10 ms. Due to hardware constraints, we run 30 data collection instances in parallel. Both machines run Ubuntu 24.04 with an AMD Ryzen 9 7950X CPU and 128 GB of RAM, and each physical network interface supports 1 Gbps full-duplex communication.

Video Selection and Formatting: Using the YouTube Data API [38], we gather over 100,000 video metadata entries from all assignable categories: Film & Animation, Autos & Vehicles, Music, Pets & Animals, Sports, Travel & Events, Gaming, People & Blogs, Comedy, Entertainment, News & Politics, Howto & Style, Education, Science & Technology. We then randomly sample 100 videos per category, yielding 1,400 videos between 5–7 minutes in duration and have at least 720p resolution. For the experiments, each video is encoded into four DASH representations—three video bitrates (1000, 2000, 4000 kbps) and an audio track—and segment them into 2-second chunks. Finally, we generate a static Media Presentation Description (MPD) manifest for all representations and host the content on an nginx server.

User Session Simulations: To collect data, we stream each video for three minutes, allowing the stream to reach steady-state streaming in the non-jumping case. To capture real-world network effects, we introduce controlled bandwidth variability under five bandwidth regimes. Here, we first create unique bandwidth patterns based on a real-world LTE trace [39] and then scale the traces by one of five factors: $\times 0.5$, $\times 1$, $\times 2$, $\times 4$, and $\times 8$. In the text, we refer to each of these scenarios as Var-X, where X $= 0.5$, 1, 2, 4, and 8.

To evaluate sensitivity to user-driven jumping actions, we repeat each stream with 0, 1, 2, or 4 timeline jumps. Jumps are scheduled relative to playback progress rather than fixed timestamps, and each jump is constrained so that at least 60 s of content remains at all times (accounting for the client buffer being capped at 60 s). This prevents edge cases where the streaming traffic is interrupted due to no more available data on the server and ensures that each interaction reflects a meaningful user action.

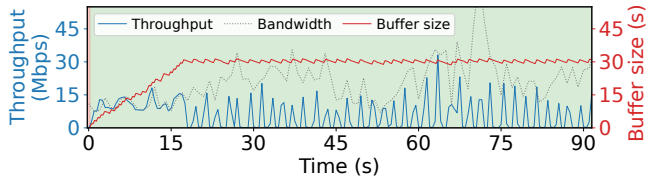
Data Collection Instrumentation: After collecting PCAP files for each session, we construct network traces containing packet timing, directions, and sizes as observed over the network. Instrumenting both clients and servers, we record all user and system events, including when the client issues jump requests, the state of the client at that time, and to which playback point they jumped. Finally, to allow easy calculation of QoE metrics, we use dash.js [36] to continuously collect QoE-related information such as the encoding and playback timing of every played segment and the timing and duration of rebuffering events.

Dataset Summary: In total, we collect encrypted traffic and QoE labels for 1,400 unique videos (14 categories $\times$ 100 videos/category), each streamed under five bandwidth conditions (Var-0.5, Var-1, Var-2, Var-4, Var-8) and four interaction settings (0, 1, 2, or 4 jumps). To establish a non-jump baseline, we repeat each zero-jump condition ten times using randomized LTE bandwidth traces. Each sample contains approximately three minutes of encrypted playback, resulting in 4,550 hours of recorded traffic. Finally, each trace contains packet timing, direction, payload size, encoded representation, and dash.js-exported QoE metrics, providing paired network-QoE observations aligned with the inference goals of Sections V–VII.

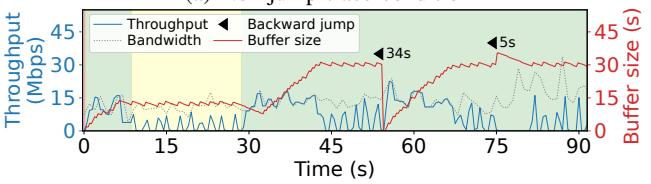
IV. QoE Metrics and Impact of Jumping

We next define the QoE metrics used throughout the paper and analyze how user jumping behavior changes QoE outcomes under different bandwidth conditions. These metrics provide the ground truth signal that the baseline RF classifier (Section V) fails to generalize to and that the extended regression and penalty-based models (Section VI) are designed to capture. We evaluate QoE over the first three minutes of playback, which typically contains buffer fill, the ABR ramp-up period (during which the player typically increases video quality from an initial conservative bitrate towards a steady-state operating point), and any early instability that shapes user-perceived quality.

Example Trace Comparison: Before analyzing the impact of jumping on QoE and QoE prediction, we first illustrate how jumping alters the observable traffic and client buffer dynamics. Figure 2 compares throughput (blue) and client buffer level (red) for a non-jump session and a jump session under time-varying bottleneck bandwidth (grey).



(a) Non-jump trace condition



(b) Jump trace condition

Fig. 2: Comparison of throughput and buffersize over time. The background coloring shows the video quality where red, yellow, and green represents 1k, 2k, and 4k Mbps, respectively.

In the non-jump case (Figure 2a), throughput exhibits the characteristic DASH pattern: an initial startup phase with back-to-back segment downloads, during which throughput closely tracks the available bandwidth, followed by steady playback where segments are fetched periodically and the video is played at the highest quality. The buffer fills during startup and then remains near a stable target level of 30 seconds, despite bandwidth fluctuations.

In contrast, the jump session (Figure 2b) shows pronounced disruptions around user interactions. First, around 8 s into the video, the stream is throttled by the available bandwidth and forced to reduce the quality to 2 Mbps. At this quality, the player attempts to maintain a 12 s buffer. As buffer conditions improve around 28 s into the video, the quality switches to 4 Mbps, triggering an aggressive increase of throughput as buffer fills towards its new steady-state target of 30 s. At 54 s, a video jump invalidates buffered content, causing abrupt buffer drops and triggering aggressive refill phases with bursty throughput. While the jump is 34 s backwards in the stream, the video player exhibits behavior similar to forward jumping, as the player has already dropped previously played content and is forced to (re-)download content. Following the first jump, the client enters a refill phase to reach steady-state streaming. When a jump targets content already buffered, as in the 5 s backward jump at 75 s, playback resumes immediately without additional downloads, and the buffer level increases instantaneously as the player reuses previously downloaded content that has not yet been dropped from memory.

These examples illustrate that user jumps introduce traffic and buffer dynamics that deviate fundamentally from steady playback, motivating the need for jump-aware QoE prediction.

A. QoE Metric Definition

We adopt standard normalized QoE components used in prior work on encrypted QoE estimation [13], [40]: mean utility (video quality), rebuffering ratio (smoothness), and switching rate (stability). All metrics are scaled to $[0, 1]$ to support comparison across bandwidths and content.

Mean Utility: We use a logarithmic utility function to reflect diminishing perceptual returns of increasing bitrate

$$\bar{v} = \frac{1}{N} \sum_{i=1}^{N} \frac{\log(r_i/r_{\min})}{\log(r_{\max}/r_{\min})}, \tag{1}$$

where N is the total number of video segments, r_i is the bitrate of the i^{th} segment, and $r_{\min}$ and $r_{\max}$ are the minimum and maximum available bitrates for that segment, respectively. Higher values correspond to better average playback quality.

Rebuffering Ratio: To measure the stall severity relative to the session duration, we calculate the rebuffering ratio:

$$\rho_{\text{rebuf}} = T_{\text{rebuf}}/T_{\text{session}}, \tag{2}$$

where T_{rebuf} is the total buffering time and T_{session} is the total playback duration. Here, large values are bad for the QoE.

Switching Rate: To measure the stability of the ABR algorithm, we calculate the switching rate:

$$p_{\text{switch}} = N_{\text{switch}}/(N-1), \tag{3}$$

where N_{switch} is the number of bitrate changes and N is the total video segments. Here, smaller values are better, as users generally prefer minimal quality switches.

Combined QoE Score: The QoE score is computed as a linear combination of the key factors:

$$\text{QoE} = \bar{v} - \beta \cdot \rho_{\text{rebuf}} - \gamma \cdot p_{\text{switch}}, \tag{4}$$

where β and γ are weighting factors. Following Chen et al. [40], we set $\beta = 10$ and $\gamma = 1$ to heavily penalize rebuffering, reflecting that stalls typically have a much stronger impact on perceived quality than bitrate switching.

B. Impact of Jumping on QoE Behavior

Figure 3 shows how QoE metrics change across jump counts and bandwidth levels. While we see relatively small impact on the mean utility at low bandwidths (network limits dominate the experience) and high bandwidths (most users obtain all chunks at highest quality), and both rebuffering ratios and switching rates are consistently small at high bandwidths (Var-8), jump behavior is a significant performance discriminator when considering the overall QoE. Looking closer, across the bandwidths, we observe consistent patterns:

- *Mean utility drops* with each jump, as the player is forced to re-ramp bitrate following jump events.
- *Rebuffering ratio increases*, reflecting buffer flush and refill cycles triggered by jumps.
- *Switching rate rises monotonically*, revealing a structured relation between interaction count and ABR instability.
- *Combined QoE decreases almost linearly* with jump count when the network is capable of adaptation.

To ease relative comparisons, Figure 3 is annotated with the percent change in the median value (P50Δ) when comparing 0-jump and 4-jump statistics. We note that the rebuffering ratio median increases sharply at low bandwidths (e.g., +124.7% at $\times$0.5 and +413.3% at $\times$1), while the QoE score median decreases at the same bandwidth conditions (e.g., $-$147.2% at

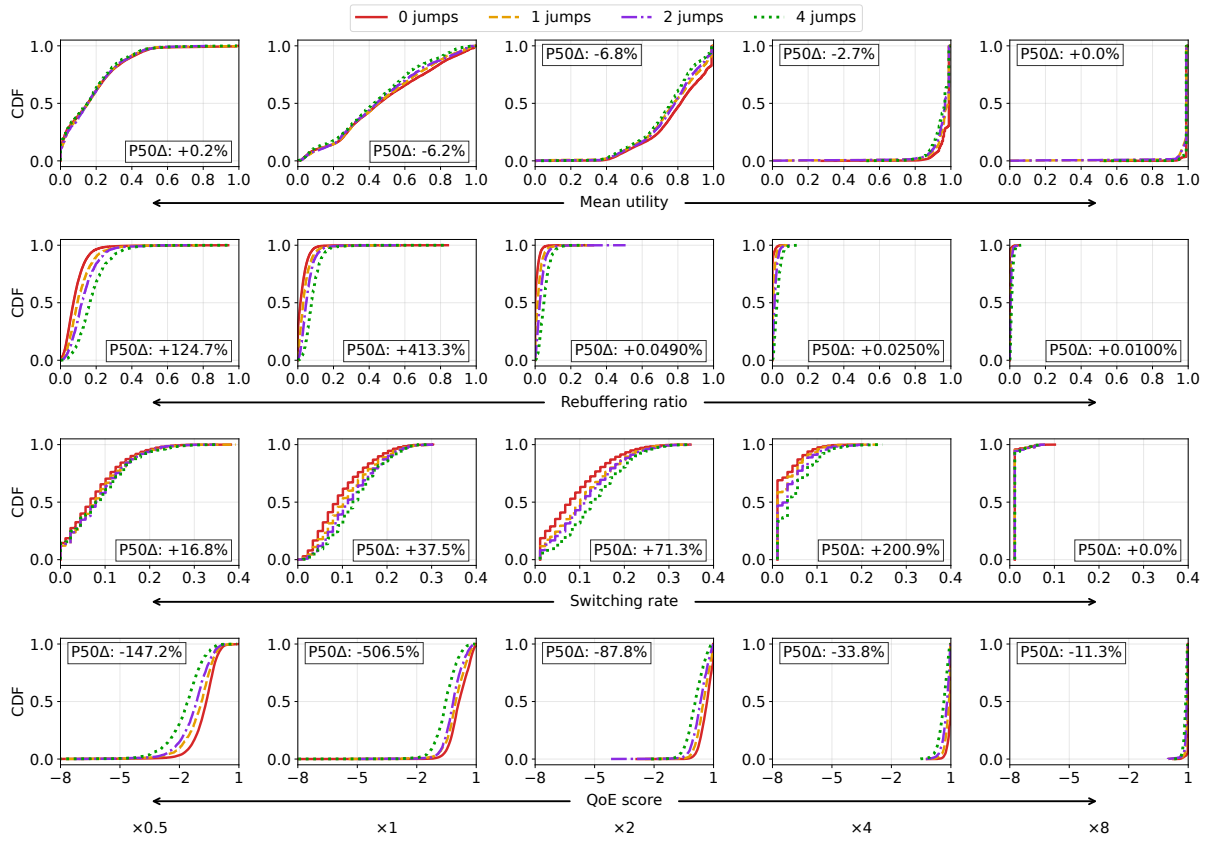


Fig. 3: QoE metric differences across variable bandwidth conditions scaled by different factors ($\times 0.5$ to $\times 8$) and jump counts.

$\times 0.5$ and -506.5% at $\times 1$), with smaller QoE drops as bandwidth increases (down to about -11.3% at $\times 8$); switching rate medians also show large increases (e.g., up to about +200.9% at $\times 4$), whereas mean utility medians change comparatively little across bandwidths.

Together, these trends show that jumping introduces a measurable and structured penalty on QoE that is *not* explained by bandwidth alone. This motivates modeling jump count as an interaction term and applying jump-aware QoE correction in our regression and penalty pipeline in Section VI.

V. RF-Based QoE Classification Baseline

This section describes the baseline model used to establish a reference point for predicting the QoE of encrypted VoD streaming traffic. The design follows the RF-based architecture used by Kapoor et al. [13], originally developed for encrypted live streaming classification. We adapt this QoE classification approach to the VoD domain without modifying the architectural objective: the model performs *QoE classification* over discrete quality levels.

A. Problem Formulation

Let x denote an encrypted traffic trace and y the corresponding QoE class label. Consistent with prior work (e.g., [13]), we define a 5-class prediction task

$$y \in \{A, B, C, D, E\}, \tag{5}$$

where the ordered classes represent decreasing user-perceived quality (A being the highest perceived quality), assigned using quantile thresholds at the 20th, 40th, 60th, and 80th percentiles of the non-jump training QoE distribution.

B. Traffic Aggregation Matrix (TAM)

Each TCP/UDP packet is mapped into a fixed temporal structure. As in RF, we construct a 2×1200 *Traffic Aggregation Matrix* (TAM) covering 180 s of playback, corresponding to approximately 0.15 s per column. Packet timestamps are discretized using $\text{idx} = \left\lfloor \frac{t}{T}(N-1) \right\rfloor$, where t is the packet timestamp (sec), $T = 180$, $N = 1200$, and $\text{idx} \in [0, 1199]$ is the bucket index. Client-to-server packets populate the first row, and server-to-client packets populate the second row.

C. Model Architecture

We leave the RF backbone unchanged: (1) 2D convolutional layers extract local spatio-temporal dependencies from the TAM, (2) a reshaping layer converts the feature map into a 1D temporal sequence, (3) a 1D convolutional stack models long-range throughput patterns, and (4) a fully-connected head with *softmax output* produces class probabilities. Finally, the network is trained using cross-entropy loss and the Adam optimizer, mirroring the methodology of Kapoor et al. [13].

D. Evaluation and Motivation for Extension

To measure robustness under timeline jumps and their resulting burst patterns, we evaluate the baseline RF classifier on both non-jump and jump traces. Only non-jump sessions are used for training, and jumping conditions are introduced

TABLE I: Baseline RF accuracy (%) when trained on non-jump traces and evaluated on jump traces across time-varying bandwidth settings (Var-X) scaled by $X \in \{0.5, 1, 2, 4, 8\}$.

Bandwidth	0 Jump	1 Jump	2 Jumps	4 Jumps
$\times 0.5$, varying	89.3	82.3	82.9	79.0
$\times 1$, varying	76.7	70.1	54.1	33.8
$\times 2$, varying	84.6	61.6	53.0	35.3
$\times 4$, varying	90.6	41.8	24.4	19.6
$\times 8$, varying	98.1	33.3	18.3	48.9

solely at test time. The purpose of this baseline is two-fold: (1) to confirm that the RF classifier achieves competitive performance under smooth, uninterrupted playback, and (2) to determine whether a model trained exclusively on non-jump traffic can generalize to the non-stationary patterns induced by user interactions.

Table I summarizes performance across all bandwidth settings. The results show that when the viewing session remains uninterrupted (0 jumps), the RF classifier performs well, consistently achieving high accuracy and even approaching 98% at higher throughput levels. This confirms that the original architecture is sufficiently expressive for linear playback.

However, performance degrades sharply once jump activity is introduced. Even a single jump causes noticeable errors across most bandwidth settings, and the decline becomes more severe for 2- and 4-jump traces. With higher bandwidth (Var-4 and Var-8), where traffic bursts are strongest, accuracy drops below 20% for 4-jump sessions. These results show that a RF classifier trained solely on non-jump playback fails to generalize to the non-stationary traffic patterns induced by timeline interactions, as its learned decision boundaries do not transfer across the resulting distribution shift.

This outcome highlights a fundamental limitation of the classification-based formulation: QoE shifts caused by user interactions are continuous and state-dependent, whereas the baseline RF treats them as fixed categorical states learned from static distributions. As a consequence, the baseline is not easily adapted to jumping scenarios. These findings motivate the next stage of our methodology.

VI. Jump-Aware QoE Prediction Framework

To overcome the generalization failure observed in Section V, we extend the RF architecture with three new components: (1) a regression head that predicts continuous QoE, (2) a jump-count estimator to detect interaction intensity, and (3) a penalty mechanism that corrects predictions based on inferred jump behavior, as illustrated in Figure 4.

A. From Classification to Regression

The RF backbone remains unchanged up to the final layers: 2D convolutional model short-range bidirectional burst structure in the TAM, and the reshaped sequence passes through 1D convolutions to capture longer-range rate variations and quiet periods. Instead of a softmax classifier, we replace the output head with a single linear unit that outputs a continuous QoE estimate $\widehat{QoE}_{\text{reg}} \in \mathbb{R}$. Training minimizes mean-squared error (MSE) loss:

$$\mathcal{L}_{\text{MSE}} = \left\| \widehat{QoE}_{\text{reg}} - QoE_{\text{true}} \right\|_2^2, \tag{6}$$

effectively encouraging the model to learn smooth degradation trends rather than assigning discrete decision boundaries. We also report relative L_2 error to normalize performance across bandwidth variations where achievable QoE scales differ.

We use Tanh activations in all convolutional blocks; preliminary tests with ReLU, GELU, and Tanh showed that Tanh yielded more stable gradients under bursty arrival patterns. The Adam optimizer is used with a learning rate of 0.01, which provided the best convergence–stability tradeoff across bandwidth conditions.

After training, normalized outputs are projected back to the original QoE scale using the stored mean and standard deviation of the training labels. To compare against classification-style reporting, the continuous predictions can be mapped into classes (A–E) using quantile-based thresholds; however, the model's primary target is continuous QoE.

B. Jump Count Estimation

We infer jump count J from encrypted traces by converting each session into a 1200-bin temporal vector and processed by a lightweight classifier to predict $J \in \{0, 1, 2, 4\}$. We train using an 80:20 split on non-jump traces and evaluate on jump traces for consistency with the stress test setup in the baseline experiment. This module provides the interaction severity signal needed to adjust the regression output.

C. Penalty-Based QoE Adjustment

To incorporate behavioral effects, we apply a jump-aware penalty to the regression output:

$$\widehat{QoE}_{\text{PB}} = \widehat{QoE}_{\text{reg}} - \alpha \cdot J, \tag{7}$$

where J is the estimated jump count and α is a penalty coefficient. We interpret α as the average per-jump reduction in our normalized QoE score under a given bandwidth regime and interaction level. Table II reports empirically estimated α values for each bandwidth regime and jump level. While absolute magnitudes may depend on player configuration and workload, the penalty formulation is generic, and α can easily be recalibrated using a small labeled calibration set.

With this combined approach, the final prediction, $\widehat{QoE}_{\text{PB}}$, accounts for both steady network behavior and user-driven disturbances, directly addressing the failure modes exposed in Section V. The regression formulation allows the model to adapt to continuous QoE variation, and the jump-aware penalty corrects for additional interaction-induced QoE degradations.

VII. Performance Evaluation

A. Regression-Based Model Alone

We first isolate the effect of replacing categorical classification with regression, without introducing any explicit jump awareness. To this end, we evaluate a regression-based QoE

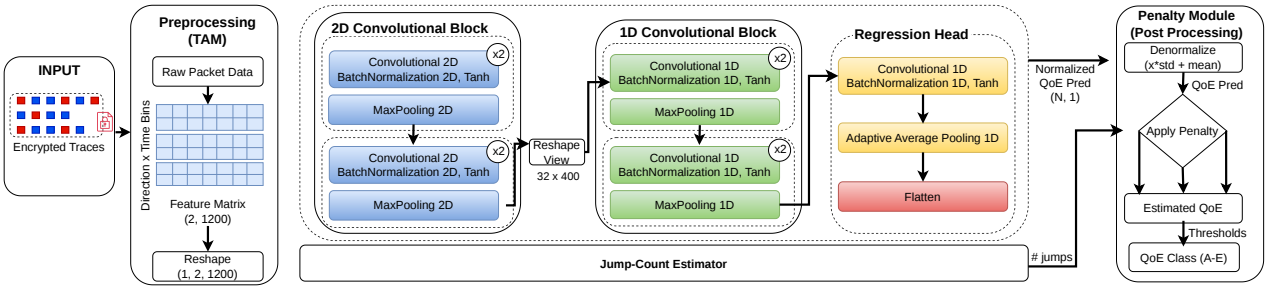


Fig. 4: Extended RF regression model for jump-aware QoE prediction.

TABLE II: Estimated per-jump QoE penalty coefficient α across bandwidth regimes and jump levels.

Bandwidth	1 Jump	2 Jumps	4 Jumps
$\times 0.5$, varying	0.2829	0.2523	0.2393
$\times 1$, varying	0.0250	0.0450	0.1706
$\times 2$, varying	0.0020	0.0200	0.0780
$\times 4$, varying	0.0050	0.0100	0.0803
$\times 8$, varying	0.0331	0.0339	0.0227

TABLE III: Accuracy (%) of regression-based model trained on non-jumping data (no penalty) and tested on jump traces.

Bandwidth	0 Jump	1 Jump	2 Jumps	4 Jumps
$\times 0.5$, varying	85.0	84.2	83.1	81.8
$\times 1$, varying	71.0	67.5	60.4	31.0
$\times 2$, varying	78.6	66.0	63.1	38.9
$\times 4$, varying	89.3	67.1	50.6	26.5
$\times 8$, varying	94.9	87.1	85.7	62.1

model trained exclusively on non-jump traces and test it on jumping sessions. Table III reports the absolute accuracy of this regression model across bandwidths and jump counts, while Table IV directly contrasts its performance against the baseline categorical classifier under identical conditions.

The results show that the regression formulation is consistently more resilient than the categorical classifier when tested on jump traces, especially at moderate ($\times 2$) and high ($\times 8$) bandwidths. This reflects the advantage of modeling QoE as a continuous signal rather than forcing discrete class boundaries learned from non-jump data.

However, despite these gains, accuracy still degrades as jump frequency increases, particularly at intermediate bandwidths ($\times 1$, $\times 2$, $\times 4$), where we see big drops in accuracy when reaching 4-jump traces. This behavior aligns with the design expectations from Section VI: while regression mitigates the brittleness of the baseline classifier, it does not explicitly correct for interaction-induced non-stationarities in the traffic. The regression head therefore generalizes better, but remains insensitive to interaction severity, motivating the jump-count and penalty mechanisms evaluated next.

B. Jump Prediction as an Interaction Signal

Table V summarizes the accuracy of the jump-detection and jump-count components across all bandwidth conditions. The model achieves perfect accuracy (100%) in separating jump and non-jump sessions across all bandwidth scales, showing

TABLE IV: Comparison of classification-based and regression-based models, both trained on non-jumping data without any jump penalty. Testing on jumping dataset.

Bandwidth	Models	Testing dataset		
		1 Jump	2 Jumps	4 Jumps
$\times 0.5$	Baseline	82.3	82.9	79.0
	Reg-based	84.2	83.1	81.8
		+2.30%	+0.24%	+3.54%
$\times 1$	Baseline	70.1	54.1	33.8
	Reg-based	67.5	60.4	31.0
		-3.70%	+11.64%	-8.28%
$\times 2$	Baseline	61.6	53.0	35.3
	Reg-based	66.0	63.1	38.9
		+7.14%	+19.05%	+10.19%
$\times 4$	Baseline	41.8	24.4	19.6
	Reg-based	67.1	50.6	26.5
		+60.52%	+107.37%	+35.20%
$\times 8$	Baseline	33.3	18.3	48.9
	Reg-based	87.1	85.7	62.1
		+161.56%	+368.30%	+26.99%

TABLE V: Accuracy (%) across bandwidth conditions for predicting the jump sessions.

Task	$\times 0.5$	$\times 1$	$\times 2$	$\times 4$	$\times 8$
Jump vs. No-jump	100.0	100.0	100.0	100.0	100.0
Jump count	84.6	84.3	84.9	86.7	88.6

user interaction is fully separable from smooth playback using encrypted traffic alone. This binary signal provides a reliable trigger to activate jump-aware QoE correction.

Beyond detection, the model also estimates the number of jumps $J \in \{0, 1, 2, 4\}$, which we use as a measure of interaction intensity. Jump-count accuracy remains stable in the mid-80% range from Var-0.5 through Var-4 and increases to 88.6% at Var-8, suggesting that the structure of jump-induced traffic patterns is robust to substantial throughput variation.

Importantly, this intermediate prediction supplies information that the regression head cannot infer from rate structure alone. Estimating J therefore provides the missing behavioral variable needed to correct $\widehat{QoE}_{\text{reg}}$ when playback deviates from the non-jump regime seen during training.

C. Penalty-Based QoE Adjustment

Table VI presents the final evaluation of the full penalty-based QoE model (PB-QoE), combining regression with explicit jump-count awareness, and compares it directly against the baseline classifier. This experiment isolates the effect of incorporating interaction severity into QoE prediction: both

TABLE VI: Comparison of full penalty-based QoE adjustment model and the baseline model.

Bandwidth	Models	Testing dataset		
		1 Jump	2 Jumps	4 Jumps
×0.5	Baseline	82.3	82.9	79.0
	PB-QoE	84.5	89.8	96.7
		+2.67%	+8.32%	+22.40%
×1	Baseline	70.1	54.1	33.8
	PB-QoE	69.6	69.2	74.7
		-0.71%	+27.91%	+121.00%
×2	Baseline	61.6	53.0	35.3
	PB-QoE	66.2	65.5	66.5
		+7.46%	+23.58%	+88.38%
×4	Baseline	41.8	24.4	19.6
	PB-QoE	69.0	59.2	66.2
		+65.07%	+142.62%	+237.75%
×8	Baseline	33.3	18.3	48.9
	PB-QoE	88.6	86.8	62.4
		+166.06%	+374.31%	+27.60%

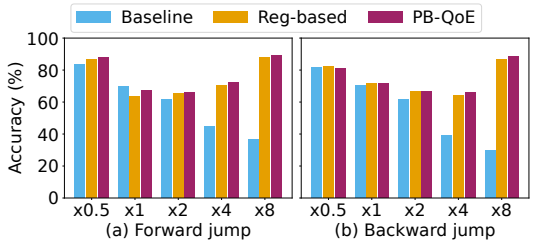


Fig. 5: Accuracy (%) across bandwidth conditions for forward and backward jumps: Baseline, Reg-based, PB-QoE.

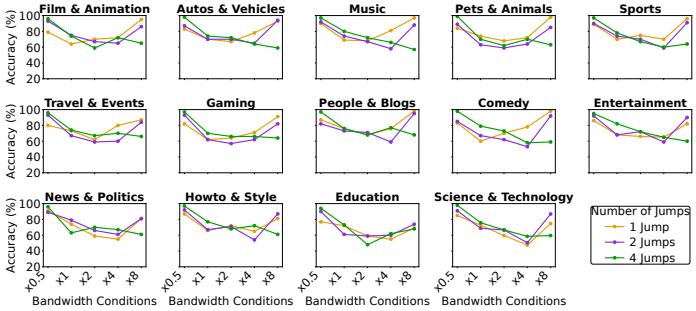


Fig. 6: Category-wise robustness across different bandwidth conditions for 1/2/4 jumps using the Reg-Based method.

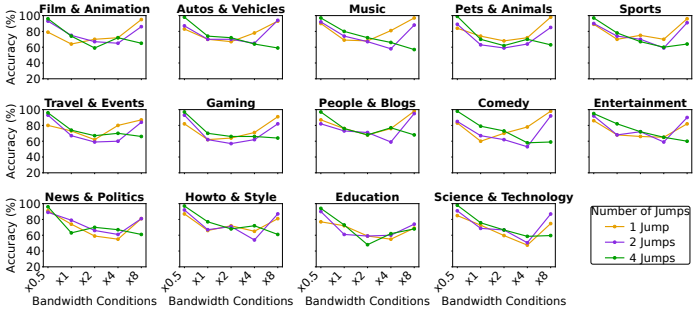


Fig. 7: Category-wise robustness across different bandwidth conditions for 1/2/4 jumps using the PB-QoE method.

models are trained on non-jump data and evaluated on jump traces, but only PB-QoE applies a jump-dependent correction.

Across all bandwidth conditions, PB-QoE consistently outperforms the baseline whenever jumps occur, with the largest gains appearing in the regimes where the regression head alone struggled. The improvement is most pronounced at Var-4 and Var-8, where the gap between the non-jump training distribution and the jump-heavy test distribution is largest; in these settings, PB-QoE more than doubles accuracy and, in several cases, improves performance by over 300%.

Most importantly, these results confirm the model's intended division of responsibility: the regression head captures continuous QoE under smooth playback, while the penalty term compensates for interaction-induced deviations. Together, they overcome the limitations of (1) the categorical model's brittleness to distribution shift and (2) the regression model's inability to infer interaction severity from throughput alone, yielding a practical and interpretable pipeline for jump-aware QoE estimation from encrypted video traffic.

D. Robustness to Jumping Direction

We next stress-test robustness to interaction type by distinguishing between forward and backward jumps, which induce qualitatively different traffic and buffer dynamics. This comparison evaluates whether QoE predictors remain stable across interaction semantics as well as bandwidth conditions.

Figure 5 reports QoE prediction accuracy across bandwidth settings for forward and backward jump traces with one jump per session. In both cases, PB-QoE achieves the best overall accuracy and shows relatively small variation as bandwidth changes, suggesting robustness to both interaction type and network variability. In contrast, the Reg-based model has slightly lower accuracy, while Baseline performs worst overall. These results indicate that although jump direction influences prediction difficulty, PB-QoE provides the most stable performance across jump directions.

E. Category-Based Robustness Comparisons

Finally, while we have found that prediction accuracy varies substantially across video categories when interaction complexity increases, PB-QoE largely mitigates this effect. Figure 6 shows that the Reg-based model is sensitive to both bandwidth variability and jump frequency across the 14 categories. While accuracy for 1- and 2-jump sessions remains moderate to high, many categories experience pronounced degradation at intermediate bandwidths (×1–×4), with performance dropping sharply for 4-jump traces. This widening gap indicates that regression alone is less robust when user behavior becomes highly interactive.

In contrast, Figure 7 shows that PB-QoE exhibits markedly more stable behavior across categories. The separation between the 1-, 2-, and 4-jump curves is substantially reduced, and bandwidth-dependent fluctuations are smoother for most categories. By mitigating the severe 4-jump performance collapse observed with regression alone, PB-QoE provides more resilient QoE prediction across both content types and bandwidth conditions.

VIII. Conclusions

This paper investigates QoE prediction for encrypted Video-on-Demand traffic under user jumping behavior, a common yet overlooked aspect of modern video streaming. Through controlled experiments, we show that user interactions introduce non-stationary traffic patterns that significantly degrade the accuracy of state-of-the-art QoE predictors trained on smooth playback, revealing a clear generalization gap. At the same time, we demonstrate that jumping behavior leaves a strong and structured fingerprint in encrypted traffic: jump events and interaction intensity can be inferred reliably across bandwidth regimes. Leveraging this insight, we introduce PB-QoE, a lightweight and interpretable jump-aware QoE prediction pipeline that combines continuous regression with an explicit, interaction-dependent penalty. While each component alone is insufficient, their combination provides robustness to both network variability and user-driven interaction. Extensive evaluation across bandwidth conditions, jump frequencies and directions, and video categories shows that PB-QoE consistently outperforms classification- and regression-only baselines, with the largest gains appearing under frequent interaction. Overall, our results demonstrate that jump-aware QoE estimation is feasible using only encrypted, network-side signals, enabling more realistic and robust QoE monitoring as user viewing behavior becomes increasingly selective.

Acknowledgement

This work was partially supported by the Swedish Foundation for Strategic Research (SSF), Karlstad Internet Privacy Lab (KIPL), the Swedish National Graduate School in Computer Science (CUGS), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE Multimedia*, 2011.

[2] X. Zhang, G. Xiong, Z. Li *et al.*, "Traffic spills the beans: A robust video identification attack against YouTube," *Computers & Security*, 2024.

[3] R. Seeliger, D. Silhavy, and S. Arbanowski, "Dynamic ad-insertion and content orchestration workflows through manifest manipulation in hls and mpeg-dash," in *IEEE CNS*, 2017.

[4] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video QoE from encrypted traffic," in *Proc. IMC*, 2016.

[5] C. Gutterman, K. Guo, S. Arora *et al.*, "Requet: Real-time QoE detection for encrypted YouTube traffic," in *Proc. ACM MMSys*, 2019.

[6] 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. IEEE Globecom Workshops*, 2016.

[7] 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. ACM MMSys*, 2017.

[8] S. C. Madanapalli, A. Mathai, H. H. Gharakheili, and V. Sivaraman, "ReCLive: Real-time classification and QoE inference of live video streaming services," in *Proc. IEEE/ACM IWQoS*, 2021.

[9] J. Oura, A. Wada, M. Ogawa, K. Yokoo, J. Kakuta, and T. Ninomiya, "QoE estimation method with time-series features extracted from packet flows for video streaming," in *Proc. IEEE CCNC*, 2024.

[10] 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 *Proc. IEEE/ACM IWQoS*, 2020.

[11] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for https and quic," in *IEEE INFOCOM*, 2018.

[12] Y. Zhao, H. Wu, L. Chen, S. Liu, G. Cheng, and X. Hu, "Identifying video resolution from encrypted QUIC streams in segment-combined transmission scenarios," in *Proc. NOSSDAV*, 2024.

[13] S. Kapoor, E. Witwer, D. Hasselquist, M. Asplund, and N. Carlsson, "Predicting video QoE from encrypted traffic: Leveraging video fingerprinting and providing system-level insights," in *IFIP Networking*, 2025.

[14] A. Carlson, D. Hasselquist, E. Witwer, N. Johansson, and N. Carlsson, "Understanding and improving video fingerprinting attack accuracy under challenging conditions," in *Proc. ACM CCS WPES*, 2024.

[15] M. Shen, K. Ji, Z. Gao, Q. Li, L. Zhu, and K. Xu, "Subverting website fingerprinting defenses with robust traffic representation," in *Proc. USENIX Security*, 2023.

[16] D. Hasselquist, M. Lindblom, and N. Carlsson, "Lightweight fingerprint attack and encrypted traffic analysis on news articles," in *Proc. IFIP Networking*, 2022.

[17] D. Hasselquist, C. Vestlund, N. Johansson, and N. Carlsson, "Twitch chat fingerprinting." in *Proc. TMA*, 2022.

[18] T. Pulls and R. Dahlberg, "Website fingerprinting with website oracles," in *Proc. PETS/PoPETS*, 2020.

[19] S. Mongy, "A study on video viewing behavior: application to movie trailer miner," *Int. J. Parallel, Emergent and Distributed Systems*, 2007.

[20] L. Chen, Y. Zhou, and D. M. Chiu, "Video browsing - A study of user behavior in online VoD services," in *Proc. ICCCN*, 2013.

[21] R. Mok, E. Chan, X. Luo, and R. Chang, "Inferring the qoe of http video streaming from user-viewing activities," in *Proc. W-MUST*, 2011.

[22] L. Chen, Y. Zhou, and D. M. Chiu, "A study of user behavior in online VoD services," *Computer Communications*, 2014.

[23] T. Nunome and H. Tani, "The effect of seeking operation on QoE of HTTP adaptive streaming services," *IJCNC*, 2017.

[24] M. Darwich and M. Bayoumi, "Video quality adaptation using CNN and RNN models for cost-effective and scalable video streaming services," *Cluster Computing*, 2024.

[25] S. Altamimi and S. Shirmohammadi, "QoE-Fair DASH video streaming using server-side reinforcement learning," *ACM TOMM*, 2020.

[26] X. Ma, Q. Li, J. Chai, X. Xiao, S.-t. Xia, and Y. Jiang, "Steward: Smart edge based joint QoE optimization for adaptive video streaming," in *Proc. ACM NOSSDAV*, 2019.

[27] X. Ma, Q. Li, L. Zou, J. Peng, J. Zhou, J. Chai, Y. Jiang *et al.*, "Qava: Qoe-aware adaptive video bitrate aggregation for http live streaming based on smart edge computing," *IEEE Trans. on Broadcasting*, 2022.

[28] A. Zhang, Q. Li, Y. Chen, X. Ma, L. Zou, Y. Jiang, Z. Xu, and G.-M. Muntean, "Video super-resolution and caching—An edge-assisted adaptive video streaming solution," *IEEE Trans. on Broadcasting*, 2021.

[29] V. Krishnamoorthi, N. Carlsson, and E. Halepovic, "Slow but steady: Cap-based client-network interaction for improved streaming experience," in *Proc. IEEE/ACM IWQoS*, 2018.

[30] G. Cofano, L. De Cicco, T. Zinner, A. Nguyen-Ngoc, P. Tran-Gia, and S. Mascolo, "Design and experimental evaluation of network-assisted strategies for HTTP adaptive streaming," in *Proc. ACM MMSys*, 2016.

[31] F. Sun, B. Liu, F. Hou, H. Zhou, J. Chen, Y. Rui, and L. Gui, "A QoE centric distributed caching approach for vehicular video streaming in cellular networks," *Wireless Comm. and Mobile Computing*, 2016.

[32] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "QoE-Driven mobile edge caching placement for adaptive video streaming," *IEEE Trans. on Multimedia*, 2017.

[33] S. K. Mehr, P. Juluri, M. Maddumala, and D. Medhi, "An adaptation aware hybrid client-cache approach for video delivery with dynamic adaptive streaming over HTTP," in *Proc. IEEE/IFIP NOMS*, 2018.

[34] D. Hasselquist, E. Witwer, A. Carlson, N. Johansson, and N. Carlsson, "Raising the bar: Improved fingerprinting attacks and defenses for video streaming traffic," in *Proc. PETS/PoPETS*, 2024.

[35] Selenium, 2025, https://www.selenium.dev/.

[36] DASH-Industry-Forum, "dash.js," https://dashjs.org/, 2025.

[37] NGINX, https://www.nginx.com/, 2025.

[38] Google, "YouTube Data API," 2026, https://developers.google.com/youtube/v3.

[39] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: A 4G LTE dataset with channel and context metrics," in *Proc. ACM MMSys*, 2018.

[40] T. Chen, Y. Lin, N. Christianson, Z. Akhtar, S. Dharmaji, M. Hajiesmaili, A. Wierman *et al.*, "SODA: An adaptive bitrate controller for consistent high-quality video streaming," in *Proc. ACM SIGCOMM*, 2024.