Predicting XR Services QoE with ML: Insights from In-band Encrypted QoS Features in 360-VR

Md Tariqul Islam, Christian Esteve Rothenberg
University of Campinas (UNICAMP)
{tariqsaj, chesteve}@dca.fee.unicamp.br

Pedro Henrique Gomes
Ericsson Research
pedro.henrique.gomes@ericsson.com

***Abstract*—The growing popularity of eXtended Reality (XR) is being driven by technological advancements and the demand for advanced immersive digital experiences, including the vision around the metaverse. Within the XR realm, 360-degree immersive video streaming is essential for Virtual Reality (VR) adventures and experiences. The use of E2E encryption for content delivery in 360-VR streaming poses challenges for network operators, making it difficult to manage their networks and assess potential Quality of Experience (QoE) impairments, specifically in 5G and beyond networks. Therefore, we propose a Machine Learning (ML) approach for inferring 360-VR video QoE metrics from network-level encrypted traffic. Our solution uses packet-level information for feature engineering, which serves as input for the ML model to predict target QoE estimators. We evaluate our solution using real 4G and 5G drive test traces with encrypted VR traffic using HTTPS and QUIC protocols. The experimental results show that the trained ML model yields reasonable accuracy with minimal residual error in predicting target VR QoE for both HTTPS and QUIC. Network operators can use such a model to passively monitor the real-time QoE of encrypted VR video sessions and optimize network performance.**

I. Introduction

Video streaming is currently a dominant force in Internet traffic growth. Recent advancements in networking and multimedia technologies are bringing increased attention to Virtual Reality (VR), specifically 360-degree immersive video streaming. As an integral part of eXtended Reality (XR) services, VR is considered a key technology in this field. Reports[1] predicted that VR-associated traffic would increase 12-fold by 2022, and the number of VR headsets (a.k.a., Head-mounted Displays or HMDs) in use is expected to reach 34 million by 2024[2]. The potential economic impact of VR is significant, with projections of a $1.5 trillion boost to the global economy by 2030[3].

To deliver satisfactory immersive Quality of Experience (QoE) for 360-VR video, high bandwidth (up to 500 Mbps) and low latency (lower than 9 ms) are required [1], which are different Key Performance Indicators (KPIs) from traditional streaming over mobile networks. This is due to the nature of VR streaming, which requires features such as viewport awareness, tile-based encoding, and 4K quality. 5G and Beyond (5GB) networks aim to meet these demands for a smooth user experience, making QoE-driven network management for XR/VR services in 5GB networks a relevant yet challenging topic.

To optimize the network for VR services, network operators need a deep understanding of the factors that affect the QoE of users. QoE is influenced by various factors such as human factors (e.g., immersion and cybersickness), system factors (e.g., network and media content), and context factors (e.g., social and task). However, operators cannot access all of these factors and are therefore relegated to investigating the perceived QoE by analyzing the traffic passing over the network and deciding on proactive capacity planning or reactive resource allocation.

Network operators traditionally relied on Deep Packet Inspection (DPI) to infer QoE information directly from network traffic. However, with the rise of E2E encryption, such as video streaming over HTTPS/TCP or QUIC/UDP protocols, inferring QoE through DPI has become challenging. Moreover, VR video encryption limits the use of Full Reference (FR) based objective quality assessment approaches, such as peak signal-to-noise ratio (PSNR) [2] and structural similarity index (SSIM) [3], whose metrics calculations are hindered by encryption blocking access to raw video data. In contrast, recent significant works [4], [5] have shown the potential of Machine Learning (ML) based correlation models, generally classified as No Reference (NR) approaches, for inferring QoE metrics from encrypted traffic for adaptive video streaming. Nevertheless, these approaches have limited applicability in inferring VR service quality, as 360-VR video is spatially divided into multiple parts known as tiles.

In this context, this work proposes an approach to infer QoE metrics of VR videos from E2E encrypted network traffic. To the best of our knowledge, this is the first study demonstrating an ML-based correlation model for mapping VR QoE metrics and in-band encrypted Quality of Service (QoS) features. The main contributions of this paper are as follows:

- We develop an emulated testbed that can generate and collect traces of 360-VR streaming in a mixture of HTTPS/TCP and QUIC/UDP traffic under various network conditions, such as 4G and 5G networks.
- We design a non-invasive and lightweight feature extraction heuristic that generates a comprehensive set of QoS features, relying only on bi-directional network layer packet-level features.
- We experimentally evaluate an AutoML-based supervised learning approach that combines shallow and deep learn-

[1]Cisco VNI - forecast and trends, 2017–2022: Cisco, 2018
[2]VR headset unit sales worldwide from 2019 to 2024: Thomas Alsop, 2022
[3]Seeing is believing: PWC tech report, 2019

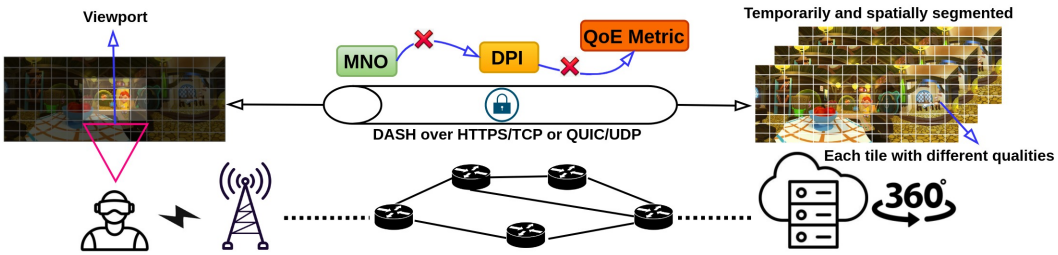


Fig. 1: Tile-Based 360-VR Streaming

ing algorithms to predict objective QoE metrics and QoE value scores of VR streaming sessions using network-level QoS indicators. Results show that for predicting VR QoE, the ensemble model provides a satisfactory fit model (up to 99%) to the data. Additionally, we made both the code and datasets utilized in our work available online[4] to enhance reproducibility.

This paper is structured as follows. Section II presents an overview of the background study, challenges, and a brief related work. Section III outlines the proposed approach and experimental setup. Section IV thoroughly analyzes the experimental results and evaluation of the model's performance. Finally, Section V concludes the paper with future directions.

II. Background, Challenges and Related Work

360-VR Streaming. 360-degree VR videos, whether traditional or stereoscopic, have gained significant research attention, differ from traditional 2D video streaming in terms of content capturing, encoding-decoding, and bandwidth requirements, and allow a user to explore the scene in all directions. However, due to the challenges associated with transmitting high-bandwidth 360-VR video over cellular networks, tile-based [6] streaming is considered the most common implementation among several approaches for 360-VR streaming (Figure 1). In a tile-based solution, a 360-degree device captures the content, which is mapped, encoded, segmented, and placed on an HTTP server. A headset (e.g., Oculus/GVR HMD) tracks the client's head movement and viewport, while a dynamic bitrate algorithm selects the video quality for each tile based on bandwidth and viewport coordinates. The content is then requested, buffered, and displayed on the headset. We further explain some key aspects of VR streaming [7], [6].

Content Acquisition and Stitching: Capturing an entire 360-degree scene demands an omnidirectional camera (e.g., Gear 360, Ricoh theta). A stitching process (e.g., planar representation) later requires aligning the different camera views.

Projection: After content capturing and stitching, a 360-degree sphere is projected to a 2D plane format before encoding and transmission. Different projection techniques are used, such as equirectangular, cubic, and pyramid projection, with equirectangular being the most prevalent and involving flattening a sphere on a 2D surface.

Tile-based Encoding: The resulting content is then encoded in different representations by encoders (e.g., HEVC/H.265) and segmented temporarily and spatially. Spatial segmentation refers to dividing a frame into several areas, known as tiles, with tile-based streaming prioritizing viewport quality while reducing other areas to address bandwidth concerns.

Transmission: To transmit all the encoded tiles in tile-based streaming, the most prominent MPEG-DASH framework is generally used to manage the viewport's quality based on available network conditions. Such a DASH framework can be over HTTPS/TCP or QUIC/UDP.

Display: On the client side, 360 video display on a headset with a viewport prediction algorithm. In practice, such a heuristic solution predicts near-future user head position (e.g., viewport) and takes advantage of a tile-based structure to download only the tiles that belong to the viewport proactively. Later, taking input from viewport prediction information, a dynamic bitrate algorithm is responsible for downloading each of the tiles of the following video segment by prioritizing the viewport's tiles in high quality.

Challenges for Network Operator. In video streaming, the objective QoE metrics (e.g., initial delay, stall, quality variation) significantly impact user engagement [4]. Unlike traditional streaming, tile-based 360-VR streaming added new factors which affect the user's QoE. Specifically, frequent quality changes on the temporal and spatial levels can be an unpleasant experience and create some physical problems (e.g., cybersickness and confusion). Poor QoE can happen for various reasons, such as overloaded servers (content provider's end), inadequate resources/network congestion (network operator's end), and weak signal/low bandwidth (user's network).

To mitigate QoE problems due to network issues, operators need to be able to measure and monitor QoE metrics. Unlike content providers, network operators only have control over network traffic information to evaluate QoE metrics. However, subjective measurement and objective PSNR/SSIM-based solutions are complicated and unfeasible for operators. Moreover, the use of E2E encryption protocols such as HTTPS or QUIC makes DPI more challenging to read the quality of video directly from the network traffic payload. Hence, operators can use a parametric packet model based on encrypted header info to predict QoE using input features from bi-directional video stream traffic.

Identifying target 360-video traffic stream and video session detection is challenging and complex for operators. Previous works [4], [5] discussed some heuristic approaches for such challenges. Nevertheless, we assume the operator can identify a video content provider's traffic streams and video sessions.

[4]https://github.com/sajibtariq/360-VR-QoE-In-band-QoS

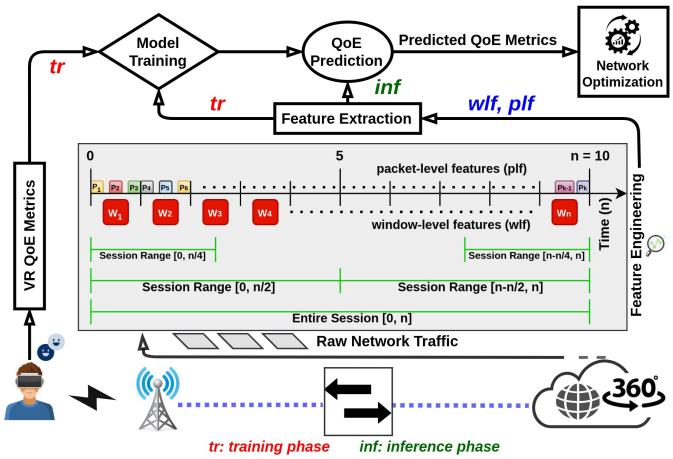


Fig. 2: Overview of the Proposed Approach

Related Work. Recent studies [4], [5], [8] proposed using ML to assess QoE for traditional 2D videos by observing network traffic, notably encrypted traffic. These studies use ML to infer application-level objective QoE metrics, such as quality, stall events, and quality variations, by correlating extracted network-level QoS features. For 360-VR videos, research mainly focused on subjective and FR-based objective QoE assessment. In literature, subjective QoE was examined by observing user characteristics and feedback, considering factors such as cybersickness, presence, motion sickness, perceptual quality, head motion, and content characteristics [9]. Objective QoE assessment of 360-VR has been conducted using metrics such as PSNR [2] and spherical SSIM [3]. Additionally, [10] proposed ML model to predict QoE metrics of tile-based 360-VR videos by incorporating out-of-band network features, such as the enforced network configuration and tiling scheme used during the controlled experiment. However, our proposed method aims to improve upon this approach by utilizing only in-band network features extracted from encrypted traffic. We believe this to be a more robust and accurate method for predicting QoE in 360-VR streaming from the network operators' perspective.

III. Methodology/Proposed Approach

Overview. Motivated by existing practices in supervised ML-based QoS and QoE correlation for traditional video streaming, we employ an ML-based approach to estimate 360-VR QoE from in-band encrypted QoS features. Though operators can store packet information for network traffic, spatial segmentation of 360-VR videos poses challenges for the operator for QoS-QoE correlation as segment requests for each tile create massive traffic for a small fraction of a video, unlike traditional streaming. Figure 2 shows a synopsis of our proposed ML-assisted approach to predict VR QoE metrics for encrypted video traffic features. Our approach for extracting network-level features is non-invasive and lightweight, utilizing IP-level header information from bi-directional network traffic without requiring computationally expensive application-level segment detection and feature extraction methods. We computed a wide range of network QoS features from the packet and

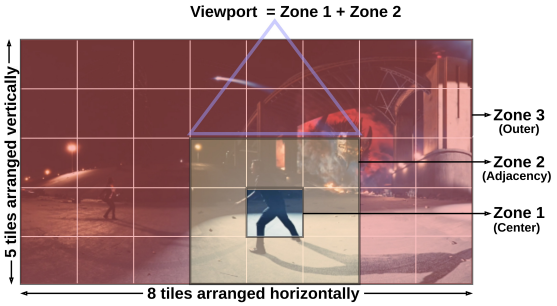


Fig. 3: 8x5 Tiling Scheme Example: 3 Zones with Center Having the Highest Quality, Followed by Adjacency, and Outer

window-based statistics for each independent VR streaming session. Our work focused on a supervised ML model to make the QoS-to-QoE correlation model for assessing VR user QoE metrics which requires ground truth from VR streaming performance for labeling the dataset. Thus the workflow has two operational phases: training and inference.

The training phase involves streaming extensive 360-VR videos under diverse network conditions and collecting traffic traces in the network premises and VR playout performance metrics (e.g., QoE metrics) as ground truth from the VR player. The raw packet traces are fed into proposed feature engineering techniques to yield a comprehensive set of features from IP-level header information. After extracting the features, the dataset, which includes the features and their corresponding ground truth, is used to train supervised ML models for various QoE metrics. In the inference phase, network features from testing data are fed into the trained model to generate predicted QoE metrics. The performance of each model is evaluated with the normalized residual errors between predicted and ground truth values. Operators can utilize predicted QoE to optimize network performance. As proposed in [11], such a QoE prediction engine can fit into the Network Data Analytics Function (NWDAF) in a service-based 5G architecture. NWDAF can gather network-level and application-level KPIs, followed by ML training and inference. Other network functions, including the Policy Control Function (PCF) and User Plane Function (UPF), can leverage NWDAF insights for optimization actions.

Experimental Setup. To implement and evaluate the performance of our work, we conducted a controlled experiment. The testbed includes Mininet-WiFi[5] to emulate network topology comprising one Access Point (AP) and one Open vSwitch (OvS) with an OpenFlow reference controller. A headless VR client [1] was connected to the AP, and a dedicated Caddy HTTP web server, specifically for streaming the encoded 360 videos, was linked to the OvS. The VR client is written in C language using Curl[6] (specifically libcurl) to download content over HTTP protocol. It supports tiling scheme and batches the tile into multiple Zones, as shown in Figure 3. It also has a Viewport Prediction Algorithm (VPA) with a controlled error injection module. VPA takes user head movement coordinates from pre-collected viewport traces dataset

[5]https://github.com/intrig-unicamp/mininet-wifi
[6]https://curl.se/libcurl/

with regular intervals (e.g., 20 ms). Moreover, the VR client supports two dynamic Adaptive Bitrate (ABR) algorithms, Full Delivery (FD) [12] and Full Delivery Basic (FDB) [13] to select appropriate quality representation for each tile by taking the benefits of VPA. The detailed functionality of the VR client can be found in [1].

To support E2E encryption, HTTPS and QUIC require a security certificate setup on both ends (client and server). Thus, for both ends, we made a self-signed certificate with OpenSSL. In the testbed, we modified the VR client to support E2E encryption for both HTTPS and QUIC transport protocols where application protocol HTTP/1.1 was used as default, and HTTP/3 (using nghttp3 library) was built over ngtcp2 (an implementation of QUIC) in Curl. Note that HTTP/3 is a new version of the HTTP protocol that runs over QUIC and aims to improve performance compared to earlier versions like HTTP/1.1. However, it is still in the experimental phase and has yet to be fully adopted. Additionally, the present (as of this writing) Curl (libcurl) based HTTP/3 implementation is experimental and not yet fully optimized for all use cases. It lacks key features such as multiplexing, server push, and other HTTP/3-specific extensions, resulting in potential performance issues in experiments. The modified VR client utilized HTTP/1.1 persistent and non-persistent connections over HTTPS and HTTP/3 persistent connections over QUIC to download tile-based VR content for each Zone. The persistent approach downloaded all tiles for each Zone under the same connection, while the non-persistent approach created a new connection for each tile.

To emulate diverse network conditions dynamically, we used Linux TC reconfiguration on the virtual interfaces between the AP and OvS in intervals of 1 second each, following the downlink bandwidth patterns from 4G and 5G cellular network traces. The traffic traces were generated using Irish telecommunication operators over different mobility [14], [15]. However, we arbitrarily selected ten traces, five for 4G (Mean= 18.59, Std= 11.99 (Mbps)) from bus, car, static, pedestrian, and train mobility, and five for 5G (Mean= 86.49, Std= 78.06 (Mbps)) from only static mobility.

The server offers two VR videos [10], "Google Spotlight-HELP" and "Freestyle Skiing", which are equipped with two different tiling schemes, 8x4 and 12x4, respectively. Each tiling scheme was encoded into three distinct quality representations, including 720p-1.8 Mbps, 1080p-2.7 Mbps, and 4K-6 Mbps. To enhance the streaming experience, each of these quality representations has been further divided into 1-second segments. We capture raw network traffic at the AP interface during a VR streaming session. For the extensive analysis, we run experiments with fixed parameters for each streaming session (up to 60-second duration), including the type of information on network traffic traces, transport and application protocol, VR video, tile scheme, ABR algorithm, viewport traces and error rate, buffer size, and segment number.

Network-level QoS Features Engineering. To collect and analyze encrypted VR streaming session traffic from a vantage point, we employed Tcpdump, a widely-used command-line tool. Tcpdump was chosen due to its efficiency and low resource utilization. It utilizes the libpcap library to capture packets from a network interface. We applied to filter criteria to the captured traffic (identified by the IP/port four-tuple) and offloaded the raw packets to a PCAP file for later analysis. We then utilized the PCAP file as input for our proposed QoS feature engineering approach, as outlined in Algorithm 1.

Algorithm 1 Network-Level QoS Feature Engineering

1: **Input:** captured PCAP file
2: **Output:** stats of four basic features for up and downlink
3: **procedure** FEATURE EXTRACTION
4: bin, $win\_thresh \leftarrow 1$
5: $time\_start$, $time\_start100$, $time\_diff$, $time\_diff100 \leftarrow 0$
6: win_start, vol, pkt_cnt, $vol100$, $pkt\_cnt100 \leftarrow 0$
7: w_{tp}, w_{pc}, p_{iat}, $p_{ps} \leftarrow []$
8: w_{tp100}, w_{pc100}, p_{iat100}, $p_{ps100} \leftarrow []$
9: **for** each packet i in PCAP **do**
10: **if** $win\_start \leq$ *timestamp(i)* $\leq win\_thresh$ **then**
11: vol += (*packetlength(i)* $\times$ 8)
12: pkt_cnt += 1
13: **if** *tcptype(i)* and *packetlength(i)* $\geq$ 100B **then**
14: $vol100$ += (*packetlength(i)* $\times$ 8)
15: pkt_cnt100 += 1
16: **end if**
17: **else**
18: Append $\frac{vol}{bin}$ to w_{tp}
19: Append pkt_cnt to w_{pc}
20: Append $\frac{vol100}{bin}$ to w_{tp100}
21: Append pkt_cnt100 to w_{pc100}
22: vol, pkt_cnt, $vol100$, $pkt\_cnt100 \leftarrow 0$
23: vol += (*packetlength(i)* $\times$ 8)
24: pkt_cnt += 1
25: **if** *tcptype(i)* and *packetlength(i)* $\geq$ 100B **then**
26: $vol100$ += (*packetlength(i)* $\times$ 8)
27: pkt_cnt100 += 1
28: **end if**
29: $win\_thresh \leftarrow win\_thresh$ + 1
30: $win\_start \leftarrow win\_start$ + 1
31: **end if**
32: $time\_diff \leftarrow$ *timestamp(i)* - $time\_start$
33: Append $time\_diff$ to p_{iat}
34: Append *packetlength(i)* to p_{ps}
35: $time\_start \leftarrow$ *timestamp(i)*
36: $time\_diff \leftarrow 0$
37: **if** *tcptype(i)* and *packetlength(i)* $\geq$ 100B **then**
38: $time\_diff100 \leftarrow$ *timestamp(i)* - $time\_start100$
39: Append $time\_diff100$ to p_{iat100}
40: Append *packetlength(i)* to p_{ps100}
41: $time\_start100 \leftarrow$ *timestamp(i)*
42: $time\_diff100 \leftarrow 0$
43: **end if**
44: **end for**
45: $stats \leftarrow$ statistics calculation of all the arrays
46: **return** $stats$

The feature engineering process begins by initializing several variables and an empty array of four basic features: throughput (w_{tp}), packet number (w_{pc}), interarrival time (p_{iat}), and packet size (p_{ps}) (lines 4-7). These network-layer features are defined based on the concept presented in earlier works [4], [5]. The calculation of these features involves various network-level statistics and is performed solely on IP header information, including IP addresses, packet timestamps, and corresponding volume from upstream and downstream encrypted traffic. This makes the feature engineering suitable for various services, platforms, protocols, and other use cases. Note that, to save space, the pseudo-code depicts the feature extraction process in a general manner, but in actual implementation, the algorithm distinguishes upstream and downstream flow based on source IP addresses. The feature extraction process is further segmented into two operations: window-level

TABLE I: Summary of Extracted QoS Feature Statistics

QoS KPI	Applied Statistics	
	Entire Session	F25, L25 F50, L50
`Throughput(TP)`	avg, max, min, medn, std, 10-90p	avg
`Packet Count (PC)`	total, avg, max, min, medn, std, 10-90p	total
`Interarrival Time (IAT)`	avg, max, min, medn, std, 10-90p	avg
`Packet Size (PS)`	avg, max, min, medn, std, 10-90p	avg

Abbreviation	Details
avg, max, min, medn, std	average, maximum, minimum, median, standard deviation
10-90p	the distribution of the 10th to 90th percentile (in steps of 10)
F25, L25, F50, L50	first and last 25% and 50% of streaming session

and packet-level.

Window-level feature extraction refers to calculating features for all packets that arrive within a specific time window. Let `W` be the time window and `P` be the set of all packets within that window. `Features(W) = F(P)`, where `F` is a function that calculates various features or statistics of the packets in set `P`. Note that the window duration, `W`, is an important factor affecting the accuracy and granularity of the features calculated. In this study, a one-second bin is used to collect window-level features. The algorithm iterates through each packet in the PCAP file using a for loop, where it checks if the packet's timestamp is within the current window of time specified by the (win_start) and (win_thresh) variables. The volume (vol) and packet count (pkt_cnt) are updated if the packet is within the window. If the packet is not within the current window, the volume frequency in bits and total packet count are appended to their respective arrays (w_{tp}) and (w_{pc}), and the window is incremented (lines 10-31). On the other hand, packet-level features refer to calculating statistics or characteristics for each individual packet rather than a group of packets within a specific time window. Let `S` be a session of network traffic, and `P` be the set of all packets that belong to that session. `Features(S) = F(P)`, where `F` is a function that calculates various features or statistics of each packet in the set `P`. The algorithm tracks each packet and calculates the time difference between two consecutive packets and the current packet's volume in bytes. The computed values are then appended to their respective arrays (p_{iat}) and (p_{ps}), and reset the variables (lines 32-43).

It is worth noting that TCP streams often contain many small control packets (e.g., SYN, ACK, RST). Therefore, for the TCP stream, the same features are additionally calculated for packets whose length is greater than or equal to 100 bytes (lines 8, 13-16, 20-21, 25-28, 37-43) In this study, the VR session is considered for different phases, including the entire session, the first and last half of the session, and the first and last quarter of the session. In the final stage, the algorithm performs various statistical calculations for all arrays in different session phases (line 45) and returns the statistics as output (line 46). As a result, a total of 292 features were extracted for the HTTPS/TCP connection and 146 features for the QUIC/UDP connection. Each computed QoS feature is given a name format: *kpi__traffic-direction__session-phase__statistic*. All the extracted QoS features for both upstream and downstream are summarized in Table I.

VR QoE Metrics as Ground Truth. Previous study [16] shows that objective QoE metrics are crucial in determining user engagement with video streaming services. In this study, we used a VR emulator that provides performance metrics to characterize a VR streaming session objectively. These metrics include tile transmission quality frequency per Zone, quality switching frequency per Zone, stall time, and startup delay. These metrics are significant because, in tile-based VR streaming, each Zone's combined performance considerably impacts user immersion. A QoE model using these metrics can provide a comprehensive understanding of the user's immersive experience and aid in identifying areas for improvement. However, creating a QoE model for tile-based VR streaming is more complicated than traditional video streaming, as it involves a more complex pipeline. In this study, we adopt a QoE model using VR performance metrics, as outlined below, proposed in [10], [17].

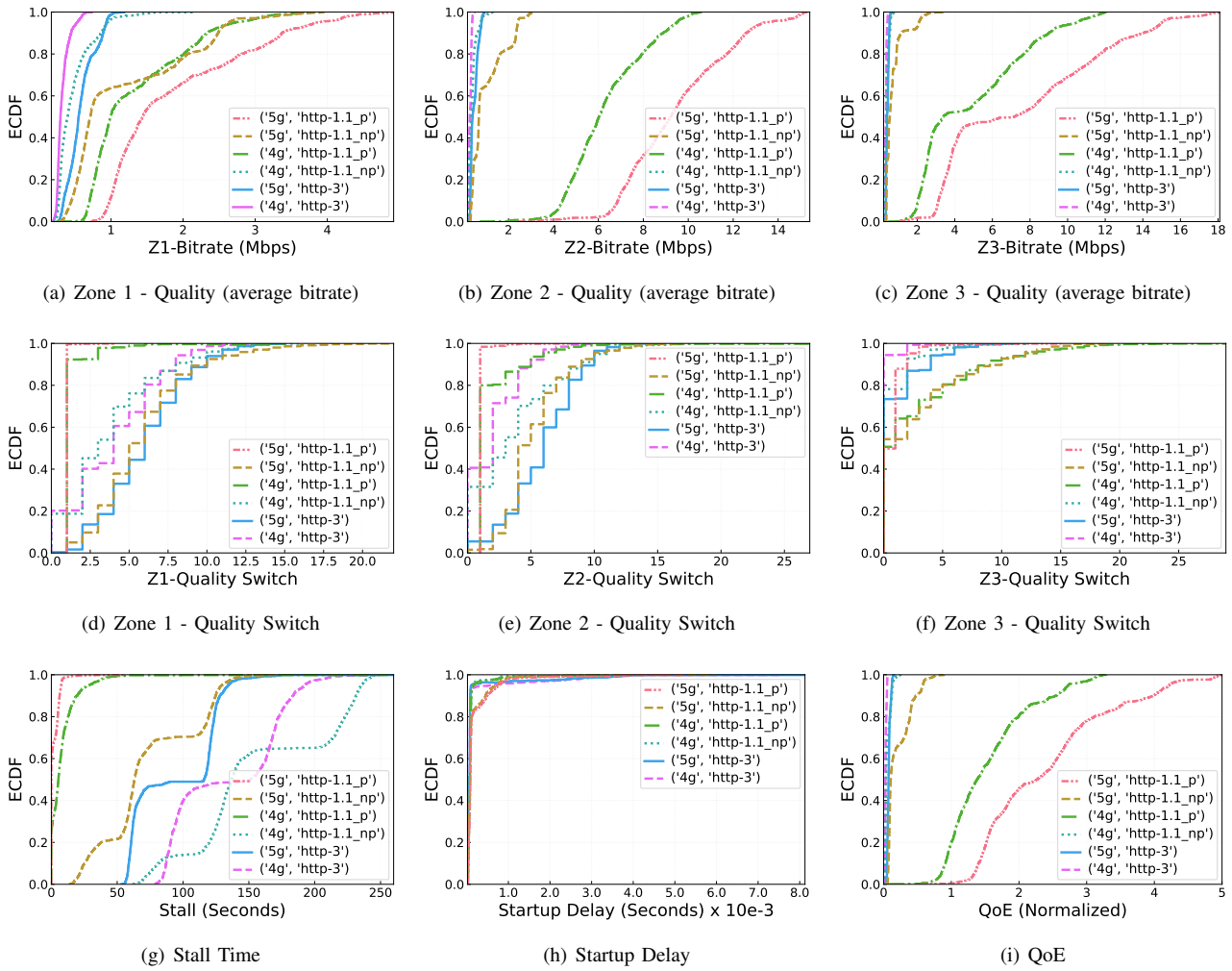
$$QoE_{\text{per Zone}} = q(R) - \mu \cdot t_{\text{stall}} - \lambda \cdot Q_{\text{switch}} - \omega \cdot T_{\text{startup}} \quad (1)$$

$$Q_{\text{switch}} = \sum_{\forall z \in \textit{all Zones}, \forall z \in \textit{all segments}} |q(R^{\text{new}}) - q(R^{\text{old}})| \quad (2)$$

$$QoE_{\text{overall}} = \alpha_1 \cdot QoE_{\text{Zone 1}} + \alpha_2 \cdot QoE_{\text{Zone 2}} + \alpha_3 \cdot QoE_{\text{Zone 3}} \quad (3)$$

In Equation 1, the first term characterizes the streaming quality and is represented by the mapping function $q(R)$, which refers to the bitrate of the streaming. Additionally, the degradation effects of the streaming experience, such as stall time, quality switch, and startup delay, are also taken into account. These factors are represented by the terms t_{stall}, Q_{switch}, and T_{startup}, respectively, and are assigned weighting factors (μ, λ, ω) that reflect their relative importance in determining the QoE of each Zone. A stall event refers to an interruption in the streaming caused by the buffer drain, while a quality switch refers to variations in the quality of consecutive segments (as per Equation 2). The startup delay is the time it takes for the initial buffer threshold to be filled and the video to start playing, and it remains constant. In tile-based streaming, the user's experience is affected differently depending on the viewing region. The central (Zone 1) region has the strongest impact on the user's perception, followed by the adjacent (Zone 2) and the outer (Zone 3) regions. Therefore, the overall QoE is calculated (as per Equation 3) by summing the resulting QoE measurements per Zone from Equation 1, with weighting factors (α_1, α_2, α_3) that reflect the relative importance of each Zone. In this study, we used nearly identical values for the weighting factors (μ = 4.3, λ = 1, ω = 4.3, α_1 = 0.7, α_2 = 0.2, and α_3 = 0.1), except for α_2 and α_3 based on work [10]. Unlike work [10] which assumed no influence of Zone 3 and set α_3 to 0, we considered a minor influence and set α_3 to 0.1, with a reduced value of 0.3 to 0.2 for α_2. Finally, to train the ML model for VR streaming sessions, the following objective QoE metrics were used as ground truth: `Quality` in terms of average bitrate per Zone, `Quality Switch` per Zone, cumulative `Stall Time`, and `Startup Delay`, along with the `QoE` model value.

Supervised ML Implementation. We used AutoGluon-Tabular [18], an advanced AutoML toolkit, to automate feature selection, model selection, and optimization by tuning



(a) Zone 1 - Quality (average bitrate)

(b) Zone 2 - Quality (average bitrate)

(c) Zone 3 - Quality (average bitrate)

(d) Zone 1 - Quality Switch

(e) Zone 2 - Quality Switch

(f) Zone 3 - Quality Switch

(g) Stall Time

(h) Startup Delay

(i) QoE

Fig. 4: The Distribution for Quality, Quality Switch, Stall Time, Startup Delay Metrics and Calculated QoE Value

hyperparameters and model ensembling for a given set of features and labels as it outperforms other AutoML tools. We selected a regression approach for our generated datasets to achieve finer granularity in QoE prediction. The toolkit offers a range of base model options for each QoE metrics prediction problem, including various Tree-based methods (such as Random Forest, Extra Trees, XGBoost, LightGBM, and CatBoost), Deep Neural Networks, and K-nearest Neighbors-based regressors. To select the best model, we split each HTTPS and QUIC dataset into a 75% training and 25% testing ratio. To save computational resources/time, we set `medium_quality_faster_train` as the `presets` parameter during training, which allowed us to train a model with good performance quickly. We then employed bagging with 10 folds to reduce the variance of the model by training multiple versions of the same model on different subsets of the data and averaging the predictions. We set no limit for model training time to achieve the best possible model. Lastly, we used `root_mean_squared_error` (RMSE) as the evaluation metric. AutoGluon uses this metric to evaluate the model's performance during training and selects hyperparameters from AutoGluon-defined sets to optimize the model by minimizing the RMSE on the validation set. In the following section, we evaluate the performance of each trained base model as well as their weighted ensemble model using the testing datasets.

IV. Experimental Results and Model Evaluation

Distribution of QoE Ground Truth. In this section, we discuss the characteristics of the ground truth dataset. Note that for ML model training purposes, we considered two datasets separately entitled HTTPS and QUIC. However, for a better understanding, we are presenting the ground truth distribution focusing on enforced network mode (e.g., 5G, 4G) and application protocol (e.g., HTTP1.1-persistent/non-persistent, HTTP/3) from both datasets. Figure 4 shows the distribution of different target QoE. The x-axis denotes ground truth values, and the y-axis represents Empirical Cumulative Distribution Function (ECDF).

In terms of `Quality` metric for Zone 1 (Figure 4(a)), we observed that using an HTTP/1.1 persistent (http-1.1_p) connection for VR streaming sessions over 5G networks resulted in higher bitrates, with 95% reaching up to 4 Mbps and 97% reaching up to 3 Mbps over 4G. Using an HTTP/1.1 nonpersistent (http-1.1_np) connection resulted in lower bitrates, with

TABLE II: RMSE and r^2 Across Different Objective QoE Metrics and QoE Value Prediction Models

	Dataset	Random Forest		Extra Trees		K Nearest Neighbors		Light GBM		Cat Boost		XG Boost		Neural Networks		Weighted Ensemble	
		RMSE	r^2	RMSE	r^2	RMSE	r^2	RMSE	r^2	RMSE	r^2	RMSE	r^2	RMSE	r^2	RMSE	r^2
Quality-Z1	**H**	-0.25	0.92	-0.26	0.92	-0.34	0.86	-0.22	0.94	-0.22	0.93	-0.22	0.94	-0.28	0.90	-0.22	0.93
	Q	-0.05	0.93	-0.05	0.93	-0.06	0.90	-0.05	0.93	-0.05	0.93	-0.05	0.93	-0.05	0.93	-0.05	0.93
Quality-Z2	**H**	-0.34	0.99	-0.34	0.99	-0.72	0.97	-0.32	0.99	-0.32	0.99	-0.32	0.99	-0.36	0.99	-0.32	0.99
	Q	-0.02	0.99	-0.02	0.99	-0.03	0.97	-0.02	0.99	-0.02	0.99	-0.02	0.99	-0.02	0.99	-0.02	0.99
Quality-Z3	**H**	-0.24	0.99	-0.20	0.99	-0.48	0.98	-0.18	0.99	-0.18	0.99	-0.22	0.99	-0.19	0.99	-0.17	0.99
	Q	-0.01	0.99	-0.01	0.99	-0.01	0.98	-0.01	0.99	-0.01	0.99	-0.01	0.99	-0.01	0.99	-0.01	0.99
Quality Switch-Z1	**H**	-1.39	0.80	-1.38	0.80	-1.57	0.75	-1.36	0.81	-1.38	0.80	-1.42	0.79	-1.43	0.79	-1.37	0.81
	Q	-2.02	0.55	-2.00	0.56	-2.16	0.48	-2.00	0.55	-2.01	0.55	-2.02	0.54	-2.02	0.54	-2.00	0.55
Quality Switch-Z2	**H**	-1.26	0.81	-1.23	0.82	-1.52	0.73	-1.25	0.81	-1.24	0.82	-1.25	0.81	-1.35	0.78	-1.27	0.81
	Q	-1.57	0.76	-1.55	0.76	-1.85	0.66	-1.55	0.76	-1.56	0.76	-1.57	0.76	-1.58	0.75	-1.53	0.77
Quality Switch-Z3	**H**	-1.16	0.87	-1.13	0.87	-1.66	0.74	-1.14	0.87	-1.12	0.88	-1.14	0.87	-1.27	0.84	-1.16	0.87
	Q	-0.86	0.68	-0.84	0.69	-0.96	0.60	-0.84	0.69	-0.84	0.69	-0.86	0.67	-0.87	0.67	-0.84	0.69
Stall Time	**H**	-2.49	0.99	-1.33	0.99	-6.13	0.99	-1.40	0.99	-1.44	0.99	-2.11	0.99	-1.71	0.99	-1.18	0.99
	Q	-2.85	0.99	-2.14	0.99	-8.15	0.96	-1.84	0.99	-2.68	0.99	-1.76	0.99	-1.77	0.99	-1.51	0.99
Startup Delay	**H**	-0.31	-0.26	-0.29	-0.15	-0.31	-0.32	-0.27	0.03	-0.27	0.01	-0.29	-0.13	-0.26	0.05	-0.26	0.05
	Q	-0.60	-0.44	-0.56	-0.27	-0.56	-0.29	-0.50	-0.01	-0.50	-0.01	-0.55	-0.21	-0.50	-0.03	-0.50	-0.02
QoE	**H**	-0.08	0.99	-0.07	0.99	-0.19	0.97	-0.06	0.99	-0.05	0.99	-0.07	0.99	-0.06	0.99	-0.05	0.99
	Q	-0.11	0.98	-0.10	0.98	-0.14	0.97	-0.10	0.99	-0.09	0.99	-0.10	0.98	-0.09	0.99	-0.09	0.99

80% reaching up to 2 Mbps over 5G and 97% reaching up to 1 Mbps over 4G. Using HTTP/3 (http-3) resulted in the lowest bitrates, with 98% reaching up to 1 Mbps over 5G and 99% reaching up to 0.5 Mbps over 4G. In Zones 2 (Figure 4(b)) and 3 (Figure 4(c)), we found http-1.1_p resulted in higher bitrates over both 5G and 4G networks. Specifically, 95% of sessions in Zone 2 and 90% of sessions in Zone 3 achieved a bitrate of up to 14 Mbps over 5G networks, and 99% of sessions in Zone 2 and 95% of sessions in Zone 3 achieved a bitrate of up to 10 Mbps over 4G networks. The performance of http-1.1_np and http-3 connections in these Zones followed a similar pattern and resulted in very low bitrates. It is worth noting that the heuristic used for VR streaming provides a higher representation rate (4K, 1080p, 720p, respectively) for each tile in Zone 1, followed by Zone 2, and finally, Zone 3. However, the overall average bitrates in the distribution for Zone 1 are lower than those of Zone 2 and 3. This is because Zone 1 contains only 1 tile, whereas Zone 2 contains 8 tiles and Zone 3 contains the rest for every single frame of the content.

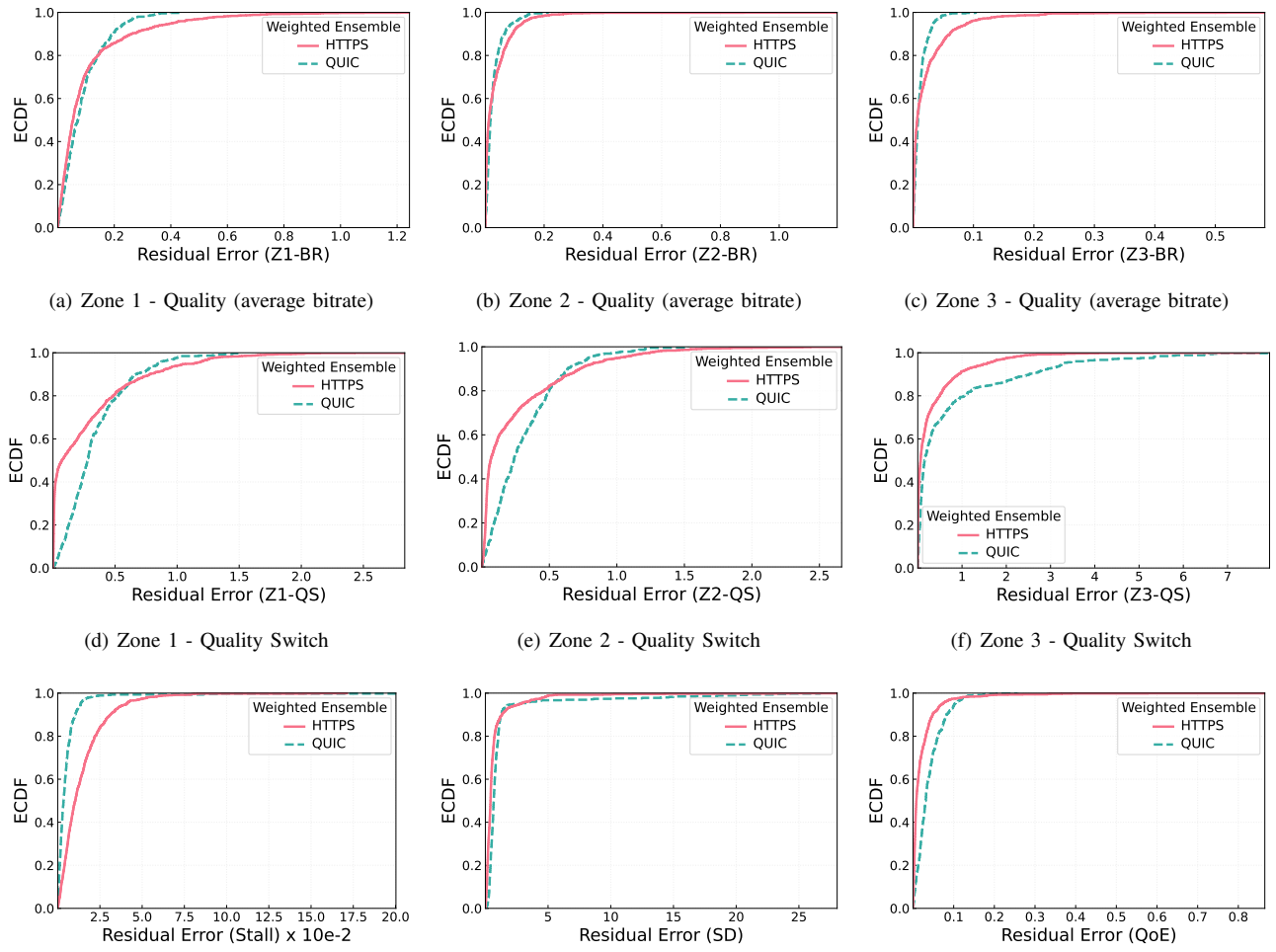
For the `Quality Switch` metric in Zone 1 (Figure 4(d)), most sessions (over 95%) had a single change in quality while using the http-1.1_p over 4G and 5G networks. This is likely because streaming typically begins with a low resolution (720p). The high bandwidth in this Zone allows for a seamless transition to a higher resolution (4K) using a VR streaming heuristic approach. However, http-1.1_np and http-3 over both 5G and 4G had up to 10-time quality changes in 90% of the sessions. In Zone 2 (Figure 4(e)), a high percentage (above 75%) of sessions also experienced a single change in quality while using the http-1.1_p over 4G and 5G networks. Additionally, up to 35-40% of sessions using the http-1.1_np and http-3 over 4G networks showed no quality changes, streaming at a low resolution (720p). In contrast, Zone 3 (Figure 4(f)) had a large percentage of sessions (50-95%) that showed no quality changes, indicating that the video was streamed at the lowest resolution.

The cumulative `Stall Time` results (Figure 4(g)) show that, as expected, http-1.1_p outperforms the others. It was found that 80% of the sessions experienced up to 10 seconds of stalling on both networks. However, over 60% of sessions did not experience any stalling over 5G, and up to 25% did not experience any stalling over 4G. In contrast, both the http-1.1_np and http-3 experienced significant stalling events across all conditions. Later, in terms of `Startup Delay` (Figure 4(h)), a majority of sessions, up to 80%, experienced a very negligible delay during initial playback under all conditions. The remaining sessions showed some variation, but the delay was at most 8 milliseconds. Finally, using the aforementioned QoE model equation, we normalized the calculated values on a scale of 0 to 5 for the `QoE` score. The results (Figure 4(i)) indicate that for above 80% of sessions, http-1.1_p connection achieved QoE score of up to 4 over 5G and 3 over 4G networks, respectively. However, http-1.1_np and http-3 scored between 0 and 1 across all conditions.

The takeaway is that VR performance using HTTP/1.1 persistent outperforms HTTP/1.1 non-persistent connection due to reduced overhead from establishing and tearing down connections for each request. However, despite the general expectation of better performance, we observed that HTTP/3 performs worse in all scenarios under the same conditions, possibly due to its implementation not being fully optimized, as mentioned in Section III.

Model Performance Evaluation. Table II shows the RMSE (rounded to 2 decimals) and r^2 (coefficient of determination) for each base model and their corresponding weighted ensemble model from both HTTPS (H) and QUIC (Q) test datasets. Note that generally, a higher RMSE value is considered worse performance. However, the AutoGluon tool that we used internally assumes that higher values are better, so it flips the sign of the RMSE and reports them as negative values. On the other hand, r^2 measures the accuracy of a regression model's fit. A value nearer to 1 signifies a more precise fit, while a negative value indicates a worse fit than a horizontal line. We found that the weighted ensemble model outperforms in most cases. Thus we selected it and evaluated its performance using residual error calculated as $|$`Predicted - Actual`$|$`/N`, where `N` is the normalizing factor that refers to an average of `Actual` values from test data.

In the prediction model for `Quality` in Zone 1 (Figure 5(a)), the residual error was less than 20% for 82% of the cases in both HTTPS and QUIC datasets. For the rest of the



(a) Zone 1 - Quality (average bitrate)

(b) Zone 2 - Quality (average bitrate)

(c) Zone 3 - Quality (average bitrate)

(d) Zone 1 - Quality Switch

(e) Zone 2 - Quality Switch

(f) Zone 3 - Quality Switch

(g) Stall Time

(h) Startup Delay

(i) QoE

Fig. 5: Residual Error for Quality, Quality Switch, Stall Time, Startup Delay Metrics and Calculated QoE Value

cases, the residual error was up to 85% and 40% in HTTPS and QUIC, respectively. On the other hand, in Zones 2 (Figure 5(b)) and 3 (Figure 5(c)), the residual error was close to zero for 45% of the cases in both datasets. The error was up to 20% in HTTPS and QUIC in Zone 2 and up to 20% and 10% in HTTPS and QUIC, respectively, in Zone 3 for the remaining cases. The overall error rate was minimal in both datasets, indicating high prediction accuracy. However, in some cases, the QUIC dataset performs slightly better.

The prediction model for `Quality Switch` in Zone 1 (Figure 5(d)) had a minimal error in up to 40% of the cases in HTTPS dataset and an error rate of 50% in 40-80% of cases in the same dataset. The error rate was up to 50% in 80% of cases in QUIC dataset. The model in Zone 2 (Figure 5(e)) showed a similar trend to Zone 1. In Zone 3 (Figure 5(f)), the prediction model showed minimal error for up to 45% of the cases in both datasets. The error was up to 50% for 45-80% of the cases in HTTPS dataset and 40-70% of the cases in QUIC dataset. Overall the prediction model in HTTPS dataset performed relatively better compared to the QUIC dataset.

The `Stall Time` prediction model (Figure 5(g)) yielded an error rate of up to 5% and 2.5% for 98% of cases in HTTPS and QUIC datasets, respectively, indicating high prediction accuracy. It performed slightly better in the QUIC dataset compared to the HTTPS dataset. The `Startup Delay` prediction model (Figure 5(h)) showed a high residual error in both datasets. The r^2 values for this prediction model in Table II indicate a poor fit of the model to the data, with negative values indicating that the model's predictions were worse than simply predicting the average of the target variable. Further investigation needs on how to enhance performance in this regard. Finally, the `QoE` value prediction model (Figure 5(i)) exhibited high accuracy with up to 10% error in up to 97% of the cases in both datasets. Overall the regression approach used in this study demonstrated decent accuracy in its predictions of all the target QoE metrics (except for `Startup Delay`) and the `QoE` value based on the extracted in-band network features.

Feature Importance. To gain a deeper understanding of the importance of features in predicting target QoE from our trained predictor using test data, we utilized AutoGluon's built-in method based on permutation shuffling. However, we briefly discuss only `QoE` value prediction feature importance due to space limitations.

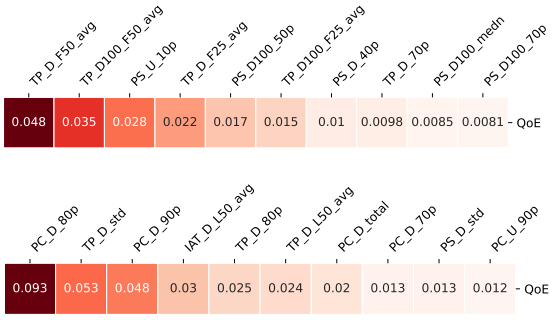


Fig. 6: Most Influential Features for Predicting QoE Value in HTTPS (Top) and QUIC (Bottom) Datasets

Figure 6 illustrates the ten most crucial features that strongly influence QoE value prediction. Across both datasets, features related to the downlink direction play the most prominent role. The most significant features in the HTTPS dataset were the average throughput (with and without TCP control packets), measured explicitly during the first quarter and half of the session, and packet size (up to 70th percentile) measured during the entire session (with and without TCP control packets). In contrast, in the QUIC dataset, packet count (70th to 90th percentile) and throughput (standard deviation mainly) features for the entire session had the most influence. Since HTTPS and QUIC datasets have different patterns of traffic, and thus, the importance of features varies in each scenario.

Such feature importance evaluations for each target QoE can provide in-depth knowledge of key features that significantly influence the prediction. This knowledge can aid in optimizing feature engineering by carefully crafting a minimal set of QoS features, leading to more efficient processing with reduced time and memory consumption.

V. Conclusions and Future Work

The widespread adoption of encryption hinders the assessment of 360-degree VR QoE for XR services. In this work, we propose a machine learning-based approach for predicting various objective QoE metrics, including Quality, Quality Switch, Stall Time, and Startup Delay, as well as the QoE value for 360-VR streaming, using network-level information as indicators. We develop a method for extracting network-level QoS features by observing bi-directional IP-level headers and evaluated it in an emulated environment. Our approach was tested on 360-VR streams over HTTPS/TCP and QUIC/UDP under emulated 4G and 5G network conditions, resulting in datasets that were used to train machine learning models. The results showed that our method achieved satisfactory accuracy in predicting VR QoE over generated datasets. However, the limitations of this work include considering only the emulation environment, unvalidated model QoE value with subjective measurement, no model generalization/reevaluation, inability to determine poor QoE root cause, and lack of integration with radio access and deployment in 5G and beyond networks.

In the future, the proposed approach can be extended by considering more complex network scenarios and incorporating the HTTP/2 protocol to take advantage of multiplexing and server push. It would also be valuable to apply the proposed approach over an actual setup, including using real VR headsets and networks. Another promising area for future research is to explore the potential of using the QoS-to-QoE model for XR services directly in the data plane by investigating the use of P4 or OpenRAN architectures as x-App/r-App.

Acknowledgment

This work was supported by the Innovation Center, Ericsson S.A., and by the Sao Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS. This study was partially funded by CAPES, Brazil - Finance Code 001.

References

[1] R. I. T. D. C. Filho *et al.*, "Dissecting the Performance of VR Video Streaming through the VR-EXP Experimentation Platform," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 4, pp. 1–23, 2019.

[2] E. Upenik *et al.*, "On the Performance of Objective Metrics for Omnidirectional Visual Content," in *2017 ninth international conference on quality of multimedia experience (QoMEX)*, pp. 1–6, IEEE, 2017.

[3] S. Chen *et al.*, "Spherical Structural Similarity Index for Objective Omnidirectional Video Quality Assessment," in *2018 IEEE international conference on multimedia and expo (ICME)*, pp. 1–6, IEEE, 2018.

[4] M. H. Mazhar *et al.*, "Real-time Video Quality of Experience Monitoring for HTTPS and QUIC," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 1331–1339, IEEE, 2018.

[5] F. Bronzino *et al.*, "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, 2019.

[6] J. V. d. Hooft *et al.*, "Tile-based Adaptive Streaming for Virtual Reality Video," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 4, pp. 1–24, 2019.

[7] R. Shafi *et al.*, "360-Degree Video Streaming: A Survey of the State of the Art," *Symmetry*, vol. 12, no. 9, p. 1491, 2020.

[8] N. F. S. de Sousa *et al.*, "Machine Learning-Assisted Closed-Control Loops for Beyond 5G Multi-Domain Zero-Touch Networks," *Journal of Network and Systems Management*, vol. 30, no. 3, p. 46, 2022.

[9] M. S. Anwar *et al.*, "Subjective QoE of 360-Degree Virtual Reality Videos and Machine Learning Predictions," *IEEE Access*, vol. 8, pp. 148084–148099, 2020.

[10] R. I. T. da Costa Filho *et al.*, "Predicting the Performance of Virtual Reality Video Streaming in Mobile Networks," in *Proceedings of the 9th ACM Multimedia Systems Conference*, pp. 270–283, 2018.

[11] S. Schwarzmann *et al.*, "ML-based QoE Estimation in 5G Networks Using Different Regression Techniques," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 3516–3532, 2022.

[12] S. Petrangeli *et al.*, "An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos," in *Proceedings of the 25th ACM international conference on Multimedia*, pp. 306–314, 2017.

[13] F. Qian *et al.*, "Optimizing 360 Video Delivery Over Cellular Networks," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pp. 1–6, 2016.

[14] D. Raca *et al.*, "Beyond Throughput: A 4G LTE Dataset with Channel and Context Metrics," in *Proceedings of the 9th ACM multimedia systems conference*, pp. 460–465, 2018.

[15] D. Raca *et al.*, "Beyond Throughput, The Next Generation: A 5G Dataset with Channel and Context Metrics," in *Proceedings of the 11th ACM multimedia systems conference*, pp. 303–308, 2020.

[16] S. S. Krishnan *et al.*, "Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs," in *Proceedings of the 2012 Internet Measurement Conference*, pp. 211–224, 2012.

[17] J. Ruan *et al.*, "A Survey on QoE-Oriented VR Video Streaming: Some Research Issues and Challenges," *Electronics*, vol. 10, no. 17, p. 2155, 2021.

[18] N. Erickson *et al.*, "AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data," *arXiv preprint arXiv:2003.06505*, 2020.