Video QoE Modeling in Enterprise Wireless Networks

Anonymous Authors

***Abstract*—**

I. Introduction

Over the past decade, mobile video traffic has increased dramatically (from 50% in year 2011 to 60% in 2016 and is predicted to be 78% by 2021) [13]. This is due to the proliferation of a number of interactive as well as non-interactive mobile video applications (such as Skype, FaceTime, Hangouts, YouTube, and Netflix etc). These applications can be categorized into Video Telephony (Skype, Hangouts, FaceTime), Streaming (YouTube, Netflix), and upcoming Virtual Reality and Augmented Reality streaming (SPT [3], theBlu [4]). Users demand high Quality of Experience (QoE) while using these applications on wireless networks, such as WiFi and LTE. This poses a unique challenge for network administrators in enterprise environments, such as offices, university campuses, retail stores. Guaranteeing best possible QoE is non-trivial because of several factors in video delivery path (such as network conditions at the client side and server side, client device, video compression standard, video application). While application content providers focus on improving the server side network performance, video compression and application logic, the enterprise network administrators try to ensure the network is well provisioned and well managed to provide good experience to multiple users of diverse applications. In this pursuit, the network administrator can only rely on passive in-network measurements to *guess* the exact experience that user has on the end-device. In this context, several prior work have looked at developing a mapping between network Quality-of-Service (QoS) and end-user QoE for video applications [7], [5], [12].

In order to train any QoS-QoE model, one needs accurate ground truth label of the user perceived QoE. Prior work has leveraged application specific APIs such as Skype Technical Information [28] or YouTube API [5]; call duration [10]; or instrumented client side library for video delivery platform [7]. While these solutions can work for specific applications and for specific providers, network administrators have to deal with a plethora of video applications. They cannot control the application logic for most of the applications. Nor does every application expose QoE metrics through APIs. Thus, to develop QoS-QoE models in the wild, network administrators need an application independent approach to measure QoE. Note that application independent does not mean ~~application agnostic~~ as a single QoE model cannot apply to all applications. Different applications exhibit different artefacts when quality deteriorates. For instance, streaming quality is impacted largely by buffering and stall ratio, whereas Telephony quality is impacted by bit rate, frames per second, blocking and blurring in the video.

Recently, Jana *et al* [15] proposed an application independent algorithm for QoE of mobile video telephony. They record the screen of the mobile device during the video call and ~~evaluate~~ QoE by ~~computing~~ *blocking*, *blurring* and *temporal variation* in the received video. However, each of these metrics has limitations. Our experiments with Skype, Hangouts and FaceTime over diverse network settings and devices show that these applications do not show any blocking artefacts. Even Jana *et al* [15] observed that blocking did not play a role in user experience. This could be due to its rare occurence. Further, we shown in section II that blur metric is highly sensitive to the level of movement and exact content in a video. This makes it very impractical for use in the wild. Same is true for the temporal variation metric. It is also worth noting that the temporal variation described in [15] is ~~not a non-reference~~ metric. ~~To apply a QoE metric~~ in enterprise networks, ~~it~~ needs to be applicable to diverse content type and not rely on access to reference video.

In this work, we propose a generic video QoE model which does not rely on application support. Additionally, it is applicable to diverse mix of content, devices and categories of video application. We take a similar approach as Jana *et al* [15], where we record the screen on the mobile device and ~~do video quality estimation~~. ~~But~~ we identify two new metrics: *intra coded bit rate* and *stutter ratio* to measure video quality. As we show in Section IV, these metrics are not sensitive to the exact content of the video call, they are only sensitive to the *quality* of the video call. Also, our evaluation is different from the this work with extensive analysis on different types of content, motion, devices.

We conduct an extensive measurement study to show how these two metrics are agnostic to ~~movement~~ in a video and exact content of the video. Further, we show that these metrics capture different artefacts in video. We combine these two complementary metrics into one through XXXX model.

II. Motivation

Lack of QoE information for all applications: While several works have motivated and addressed the problem of network based QoE estimation, little attention has been paid to the problem of collecting the QoE ground truth itself. Most related work has relied on application specific information. This approach is effective for QoE optimizations by content providers, as they only focus on a single application [7] or

intial verification of QoS to QoE mapping [5]. But network administrators in enterprise networks have to ensure good experience for a set of applications being used simultaneously in a network. Table I shows that not all popular applications provide QoE information. Even for applications like Skype, availability of Technical Information is dependent on the version of the application and the mobile OS (no technical information on iOS).

no need to capitalize

To apply QoE estimation models in real networks, we need to remove the dependency of QoE metrics on application features, such as Skype Technical Information. One way to achieve this is to simply record the video as it plays on the mobile device and analyze this video for quality.

Unreliable non-reference quality measures Video quality literature has two ways of measuring QoE: subjective and objective. Subjective metrics are measured in terms of Mean Opinion Score (MOS) collected through user surveys. They capture absolute QoE but are tedious to conduct and scale. Alternatively, objective metrics can be computed from the video itself. Objective metrics are further classified in two ways: reference based and non-reference based. As it is very difficult to get reference videos in the wild, non reference based quality metrics are more desirable. Recently, Jana *et al* [15] have proposed a non-reference metric for QoE estimation of Skype and Vtok application. They record received videos for each of these mobile telephony apps and calculate three non-reference metrics: *blocking, blurring* and *temporal variations*. Then, they combine these three metrics into one QoE metric by using MOS from subjective user study. Their study shows that blocking does not impact MOS of a video clip. While they show that their *blur* metric and *temporal variation* metrics correlate well with MOS, they do not evaluate these metrics over a wide range of clips. For example, one can wonder if *blur* is sensitive to the content in the clip.

in the wild, we do not need reference videos. The real challenge for reference metrics is tight synchronization, right?

We conduct similar experiments as [15] with Skype to evaluate their blur metric. The experimental setup is described in section IV. Jana *et al* [15] used [16] to capture video blur. This method uses Discrete Cosine Transform (DCT) coefficients from the compressed data by computing the histogram of transform coefficients thereby characterizing the image quality. The assumption here is that blurred image will have its high frequency coefficients set to zero. Hence, the method looks at the distribution of zero coefficients instead of absolute pixel values. Although the method estimates out of focus blur accurately, it falls short in estimating realistic blur and sensitivity to noise. The authors also point out that the method is very sensitive to uniform background and images with higher luminance components. We evaluated blur for 20 video sequences using this method as shown in Fig. 1. These videos are grouped into two categories: Low quality and High quality, where the low quality videos are created by compressing high quality videos highly and changing resolution. We collected these videos in a representative manner to cover diverse content, different types of motion and downloaded originally with same resolution (FullHD).

We observed two aberrations in using this blur metric: First,

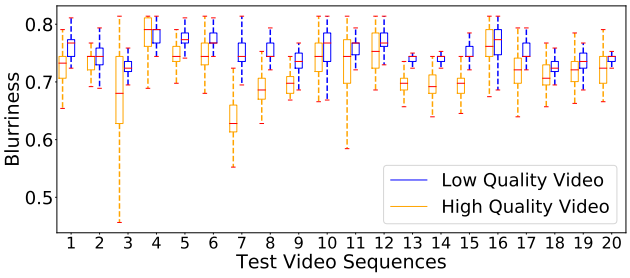


Fig. 1: Blur Detection using DCT Coefficients

the metric is indeed content specific i.e, although it produces high blur values for some low quality videos, it also shows high blur values for some high quality videos. For instance, video 2 and 4 contain mostly over-illuminated and dark images which are equally recognized as blurred in both low and high quality scenarios. Second, in order to identify accurate blur amount, the image has to be blurred heavily i.e, it shows low blur difference between high quality and low quality videos even though we used very low quality videos (240p resolution and high (>200%) compression). Even for a single video, this blur metric shows a lot of variation, such as for video 3 and video 11, raising concerns about its accuracy. We evaluate the accuracy of this metric in section VI. We did not observe any blurriness in any of the high quality videos. Apart from [16], we experimented with four other well known blur metrics [14], [19], [24], [17], but none of these methods were scalable across diverse videos.

In our experiments, we did not see any blocking artefacts. We observed mostly video stutter (temporal feature) and blurring (spatial feature). Hence, we consider metrics that capture these blurring and stutter in our experiments. None of the exisitng blur metrics [16], [14], [19], [24], [17] can be applied as is for our purpose. Further, the *temporal variation* metric in [15] considers the ratio of frames lost to total number of frames in a video, but it does not explain how they get both these values without refering to the original sent video. Thus it is not entirely non-reference. This motivates us to look at new metrics to capture both these artefacts.

this is not a clear argument. because they do not explain, it does not mean that it does not work. Need to come up with more arguments or have a second fig like Fig. 1

Need for a model per application Even while analyzing a recorded video for an application, the QoE metric has to be sensitive to the application category as different applications have to meet different performance guarantees. For instance, streaming quality is impacted largely by buffering and stall ratio, whereas Telephony quality is impacted by bit rate, frames per second, blocking and blurring in the video. Table I list some of the QoE metrics corresponding to different applications. We also observed that within same application category, Telephony for instance, Skype, Hangouts and FaceTime are behaving very differently. We observe more video freezes in Skype while FaceTime and Hangouts ~~are compromising~~ quality over video freezes.

compromise

Application	Ground-truth
Skype	✓*on some versions
Hangouts/Duo	×
FaceTime	×
YouTube/Netflix	✓

A. Availability of QoE Ground-truth

Application Category	Suitable Metric
Adaptive Streaming	Startup delay, Buffering ratio, Stall ratio
Video Telephony	Bitrate, Fps, Blocking, Blurring
Progressive Downloads	PSNR, SSIM
VR/AR, Game Streaming	Latency, buffering ratio, Stall ratio

B. Metrics based on Application Category

TABLE I: Application Specific QoE Information

To address these requirements, we try to answer the question: *How can we estimate the quality of a video call or video streaming session, without any support from the application?*.

III. Background

A. Video Coding Primer

Video coding is performed in three critical steps:

- Frame prediction: The encoder takes images of video and divides each image into macroblocks (typically 16x16, 16x8, 8x16, 8x8, 4x4 pixel blocks). The macroblocks are predicted from previously encoded macroblocks, either from current image (called intra frame prediction) or from previous frames and future frames (called inter frame prediction). Based on the intra and inter macroblocks, the frames are classified as I, P and B frames. The I frame uses intra only prediction i.e, it does not uses any previously coded frames as reference. Whereas P frames are predicted using previously coded frames and B frames are coded from previous as well as future frames. The encoder typically combines both intra and inter prediction techniques to exploit spatial and temporal redundancy respectively. The intra prediction involves different prediction modes [22] while finding candidate macroblock. Inter prediction uses motion estimation by employing different block matching algorithms (such as Hexagon based or full exhaustive search) to identify the candidate macroblock across frames. Finally, a residual is calculated by taking the difference (measured typically using MAD, SSD or SAD) of predicted macroblock and current macroblock. The prediction stage produces 4x4 to 16x16 blocks of absolute pixel values or residual values.
- Transform coding and Quantization: These absolute values are then transformed and quantized for further compression. Typically, two transform coding techniques (Block based and Wavelet based) are used to convert pixel values into transformed coefficients. A subset of these transform coefficients are sufficient to construct actual pixel values, which means reduced data upon quantization. The most popular transform coding is Discrete Cosine Transform (DCT) over 8x8 macroblocks.
- Entropy Coding: Finally, the coefficients are converted to binary data which is further compressed using entropy coding techniques such as CABAC, CAVLC or basic Huffman coding. More details of video compression can be found in [22].

B. QoE Artefacts in Video Telephony

When a user is in a video telephony call, she can experience different aberrations in the video quality, such as:

Stutter: Stutter is a temporal disruption in a video. A user may experience stutter when the incoming video stalls abruptly and the same frame is displayed for a long duration on the screen. Additionally, stutter may also appear as a fast play of video, where the decoder tries to recover from frame losses by playing incontiguous frames in quick succession, creating a perception of ”fast” movement. Both these temporal artefacts are grouped into stutter. This happens mainly because of loss (where some frames or blocks lost) and delay (where the frames are dropped because the frames are decoded too late to display).

Blurriness: Blurriness shows up when encoder uses high quantization parameter (QP) during transform coding. Typically, servers use adaptive encoding based on the network conditions. In adaptive encoding, server tries to adjust QP to minimize bitrate in poor network conditions, which degrades the quality of encoded frame. Another way to describe this is that loss of high frequency information in the image makes it more blurry. High QP reduces the magnitude of high frequency Discrete Cosine Transform (DCT) coefficients almost down to zero, consequently losing the information and making it impossible to extract original DCT coefficients at the time of De-quantization.

Blocking: Most of the current coding standards (such as H.26x and VPx) are block based, where each image is divided into blocks (from 4x4 to 32x32 and recent 64x64 block in H.265). The blocking metric captures the loss of these blocks due to high network packet loss. The decoder will introduce visual impairments at the block boundaries or sometimes places random blocks in place of original blocks if the presentation timestamp elapses, which makes experience bad.

Call Startup Delay: Call startup delay is the time taken from the caller initiating the call to the call ringing at the callee's end. In cases where the caller is not able to setup the call, the call startup delay would be infinite. While call startup delay does measure user experience, it can only provide information at the beginning of the call, not about how the experience is during the call.

Video Sequence	Frames	Fps	bitrate (Kbps)	Motion	RFps	Rbitrate	Stutter Ratio
$TalkShow.y4m$	600	40	1360	low			
$Bunny.y4m$	504	25	3500	average			
$Bus.y4m$	500	12	19037	high			

TABLE II: Motion classification based on Video Coding Parameters and Corresponding QoE Ground-truth

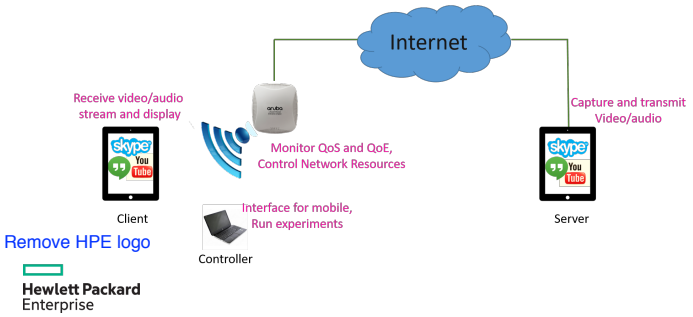
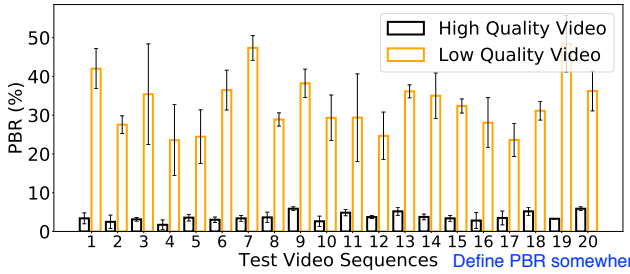


Fig. 2: Measurement Setup



Define PBR somewhere and be consistent with notations. Eg, you have Bitrate change, PBR, intra-coded bitrate for the same video

Fig. 3: Blur Detection using Intra Coded Bitrate

IV. Design

A. Measurement Methodology:

Setup: Our The experimental setup is shown in Figure 2. It includes two clients: one in our LAN and another other residing in the cloud. Here, we used Amazon provided Skype instance as the second client. At the Amazon client, we use *manycam*, a virtual webcam tool that streams a predifined video instead of raw camera capture. This tool can be used with multiple video applications in parallel. A controller sits in the LAN to emulate the network conditions and run experiments on the mobile device connected through ADB interface (via USB or WiFi connection). On our client, we screen record the video call session.

Video Processing: The recorded video is post processed to extract the QoE information. We use metrics such as intra coded bit rate, blurring and blocking by extracting individual images from the recorded video and from video coding. Ffmpeg [2] is a video framework with large collection of coding libraries. We use Ffmpeg to convert, encode and manipulate videos to extract Rbitrate, RFps and stutter ratio etc. We use Matlab and OpenCV to convert video to images and extract blur metric of individual images.

how do we use this? Is it only for the DCT coefficient or our metric?

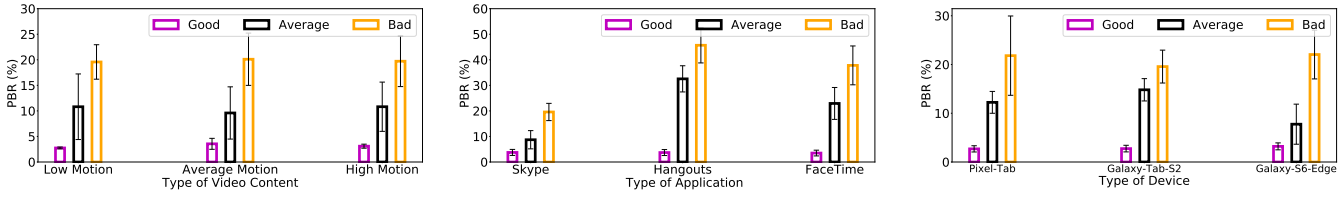
B. Video Sequences

We identify three different types of videos in terms of motion present in the video: low, average and high motion. We collected these test sequences from Xiph media [18] and YouTube. For all the measurements, we use $busMotion.y4m$, $bigBuckBunny.y4m$ and $jimmyTalkShow.y4m$ videos that falls into high, average and low motion respectively. Our automated Telephony client is restricted to few container formats, so we could not host raw videos but had to encode into AVI container format. While converting $y4m$ into AVI, we use $ffmpeg$ to extract fps and $bitrate$ of compression. We exploit these video coding artifacts in classifying the motion present in the video. We observe motion is directly proportional to fps and inversely proportional to output $bitrate$ of video coding. Table II gives statistics corresponding to motion in the video. All the videos are of High Definition (HD) resolution with $16:9$ aspect ratio and $4:4:4$ color space format. The videos are encoded under same encoding parameters such as libx264 video codec, quantization parameter (25), YUV $4:2:0$ output color space format.

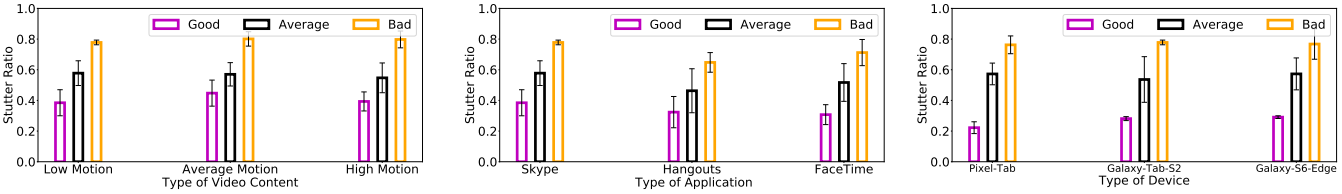
C. Our QoE Metrics

Bitrate Change: We capture video blur with encoding bitrate by compressing the recorded video. The idea here is that the more blurry blurrier the video is, the high higher compression efficiency ~~it~~ is. However, the block movement for high motion videos is very high and it is low for low motion videos which results in different encoding bitrates. The low motion videos takes the advantage of inter frame prediction over high motion videos which makes the encoded bitrate motion sensitive. Hence, we use intra coded bitrate by disabling the inter frame prediction while compressing video. We use Ffmpeg's libx264 encoder for compression. The encoder uses different macroblock prediction modes for luma (brightness component) and chroma (color component) [22]. The matching candidate macroblock is chosen based on the SAD of current and previous coded macroblocks from the same image. The candidate macroblock is then subtracted from the current block to form the residual for next stage of coding (transform and entropy coding). To make the metric more robust, we experimented with different encoding parameters like Quantization Parameter (QP) and deblocking filter techniques. We chose high QP (30) while coding, so that we get ~~more~~ larger difference when coding high and low quality videos. To make the metric more robust to video content, we take the percentage change in bitrate from recorded best quality video to current recorded video. Fig. 3 shows bitrate

can this be considered as sort of reference?



(a) Content and Motion (b) Applications (c) Devices



Polish legend to "good/average/bad network" conditions

(d) Content and Motion (e) Applications (f) Devices

Fig. 4: Bitrate and Stutter Ratio for different videos with respect to content, application and device conditions.

change for 20 videos[1]. These 20 high resolution (720P) videos are reduced to low quality by changing resolution to 240P and compressing (with QP as 30) and recorded them during the Skype call under best network conditions. This experiment is just to validate our blur capturing metric ~~so~~ hence we exclude stutter by sending low quality videos over best network conditions. Unlike blur detection using previous methods, we see a clean gap of median bitrate change between the low quality (>20%) videos and high quality videos (<5%).

Stutter Ratio: We use Ffmpeg's *mpdecimate* [**?**] filter in calculating stutter ratio. By default, the filter divides the current and previous frame into 8x8 blocks pixels and computes SAD for each block. A set of thresholds (*hi, lo and frac*) used to determine if the frames are duplicate. The thresholds *hi* and *lo* represents number 8x8 pixel differences, so a threshold of 64 means 1 unit of difference for every pixel. A frame is considered to be duplicate frame if none of the 8x8 blocks gives SAD greater than a threshold of *hi*, and if no more than *frac* blocks changed by more than *lo* threshold value. We experiment with several other error methods such as SSD and MAD, but we notice no greater difference among all three, thus ~~so~~ we choose SAD for minimal computation overhead. We use ~~a~~ threshold values of 64*12 for *hi*, 64*5 for *lo* and 0.1 for *frac* for all our experiments. We observe see two challenges complications in using metric for very high granular thresholds of labeling the experience: First, the metric does not work if the entire video is having a still image (for instance a black screen throughout the video). However, we think that it is a reasonable assumption that most of video applications will not generate such video content. Second, the metric is a little sensitive to motion in the video i.e, the SAD can of a low motion video could result in lower values compared to high motion videos.

this needs to be carefully addressed. What is little? in numbers or percentage?

[1]The videos are located at the following webpage anonymously: http://www.anonymous.com/

However, we find this effect is nullified (or dominated) by the impact of video freezes due to network. That is, the stutter ratio varies a bit across different videos under best network conditions, but will become consistent threshold under poor network conditions.

use numbers and percentages instead of saying "a bit"

rarely use future tense. here simply say "becomes consistent". Also, "become consistent threshold" needs rewriting.

D. Measurements

Using Call Setup Delay: Skype call startup delay is measured with respect to different loss conditions. We observe an 11 secs worst case and 7 secs median delay when there is 20% packet loss, whereas ~~it is~~ delay has a median 3sec in case of best network condition. ~~However, we~~ W can use this information in the very beginning to predict the call quality ~~but cannot be used during the call.~~

However, as network conditions vary, startup delay does not give any information about QoE during the call.

Video Motion and Content Diversity: The bitrate change is measured for 20 videos by hosting them under HD (1280x720) and CIF (320x240) resolutions. This experiment ~~is to see~~ focuses on showing how well the metric ~~is capturing~~ captures blur present in the video. We avoid stutter in the video by hosting low resolution videos under good network conditions. Fig. 3 shows bitrate change with respect to 20 videos. The bitrate change for all videos has consistent median of less 5% for high quality while more than 25% for low quality videos. Although the variance in bitrate change for low quality videos is higher, there is clean gap between the bitrate change of low and high quality vidoes. This shows that the metric is robust under different video contents and motion in the video.

Further, we find bitrate change and stutter ratio are highly correlated with video quality under different network conditions. We measure these two metrics with six different videos that cover different video motion and content diversity. These experiments are placed on Skype. All the videos are recorded under FullHD (1920x1280) resolution with 60Fps on Samsung Galaxy Tab S2. Fig. 4 shows bitrate change and

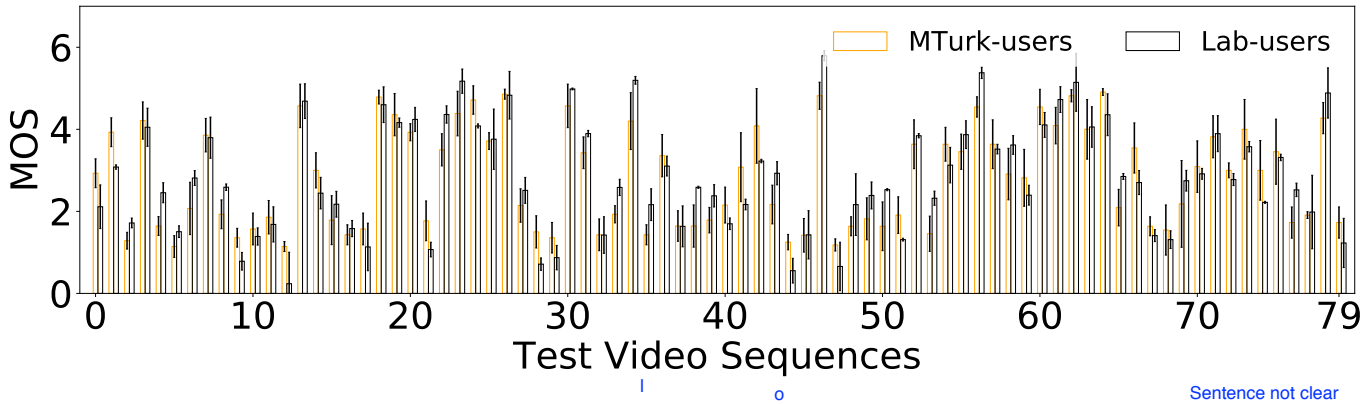


Fig. 5: MOS and standard deviation of QoE scores from 30 Lab and 30 Online users. The variance is within 1 score of standard deviation for 96% of videos. Although the scores vary across the users, the distribution of MOS is similar in two cases.

stutter ratio for Skype under good (no loss), average (1Mbps and 5% packet loss) and bad (500Kbps and 20% packet loss) network conditions. For all the videos with different content and motion, the bitrate change and stutter ratio varies significantly with respect to network conditions. The median bitrate change for best network conditions is always less 5% and more than 20% under bad case. The median stutter ratio for best network conditions is less than 0.2 and more than 0.7 under bad case. Two key observations are: First, we can map these metrics with QoS parameters easily because of clean seperation. Second, the metrics are indeed resilient to differnt content and motion present in the video.

Application Diversity: To make the model robust to multiple applications, we evaluated our metrics with two applications: Skype and Hangouts. Fig. 4 shows the bitrate change and stutter ratio for three applications: Skype, Hangouts and FaceTime. There are three observations: First, the metrics are robust in capturing the QoE of different applications irrespective of content and motion in the video. Second, the median bitrate change for Skype is around 20% whereas for Hangouts it is very high (>30%). Third, the stutter ratio for both Skype and Hangouts falls same in all network conditions. Upon further inspection, we see a lot of video blur in the Hangouts video call compared to Skype video call which reveals the reason behind higher bitrate change in case of Hangouts. We suspect the rate adaptation at the Hangouts application compromizes with the image quality with more quantization whereas Skype retains high quality image. Moreover, the quality and bitrate are highly impacted by underlying video codec the application uses. In our experiments, Skype uses H.264 whereas Hangouts uses VP8. We suspect using different video codecs and different Adaptive Bitrate (ABR) algorithms will produce differnt video quality during call.

Device Diversity: We aim our model to be independent of the device in which the video recordings are placed. That is, the metrics should capture the same video QoE on different devices given the screen recorder is same on all devices. Towards this goal, we evaluated our metrics on three devices: Samsung Galaxy Tab S2, Samsung Galaxy Edge S6 and Google Pixel Tab. As required, both bitrate change and stutter ratio shows same distribution under different network settings on all devices as shown in Fig. 4. *What could be the reason behind a bit higher stutter ratio on samsung devices under good network case?*

V. Video QoE User Study

We map our QoE metrics presented in §IV-C with a baseline metric. Since we do not have a reference objective metric, we carry out a user study and get a mean opinion score for each video.

A. Set-up

We conduct the user-study in two phases: 1) In the Lab, 2) Online using a crowd sourced platform. The lab user study is conducted at a university campus and corporate company labs. The online user study is conducted on the Amazon Mechanical Turk platform [1]. The online users are from within the United States. All the users are in the age range of 20 to 40. We collect results from 60 Lab users and XXX Online users. The user-studies in our paper are approved by the Institutional Review Board (IRB) of our institution.

B. Data Preparation

The video calls are recorded across different network conditions to produce good, average and bad quality videos. We record video calls for Skype, Hangouts on Galaxy S6 edge, Google Pixel Tab and FaceTime on Ipad. We post process these videos into 30 seconds small video clips and evaluate our QoE for each sample. We host a webserver with these video clips and ask the users to rate their experience after watching each sample. The challenges faced while taking in-the wild user-study are: We had to make sure 1) the users watch the whole video without fast forwarding and then rate

Devices		Model Performance		
Training	Testing	Precision	Accuracy	Recall
SG-S6 Phone	SG-S6 Phone	85.61	84.82	84.52
	SG-S2 Tab	84.84	84.22	83.41
	Pixel Tab	82.29	82.16	80.81
SG-S2 Tab	SG-S6 Phone	83.23	83.71	82.67
	SG-S2 Tab	83.34	84.28	84.25
	Pixel Tab	82.88	84.34	81.18
Pixel Tab	SG-S6 Phone	82.42	82.34	81.93
	SG-S2 Tab	83.17	82.14	79.27
	Pixel Tab	84.87	84.67	84.81

TABLE III: Skype Model Performance with Device Diversity

which model is this one?

their experience, 2) the users can rate the video if and only if they watch the video, 3) the users watch on same sized screens to avoid huge variance i.e, small screen users do not perceive poor quality compared to bigger screen users. We address these challenges by carefully designing the user-study website and avoid the noise as much as possible. First, we disable the video controls in the webpage so that users cannot advance the video to finish early. Second, we introduce random attention pop-up buttons while watching the video and ask the users to click on the attention button to confirm that they are watching the video. In case the user not confirm attention, the user cannot rate that video and cannot goto next video until finish the current video and rate. Finally, we restric mobile users in taking the study, to avoid variance due to small screens. The user-study webpage can be run on Chrome, Firefox and Safari on all the Laptops and Desktops. Once the user complete rating all the videos, the results are pushed to our server. Then, we notify a server generated code to user and accept the same in MTurk for processing payments. The MTurk also restricts the users in taking the same study more than once, therefore all the users are unique.

to

on laptops...

C. Results

compare Lab users and online users. Compare the bitrate and stutter with MOS

here, we need to discuss why some models do not perform well based on our dataset

VI. Model

VII. Related Work

There is an extensive notion of prior work on QoE modeling. In this section, we describe our work by comparing literature categorized into following:

QoS based traffic control: In the past user experience is evaluated using network parameters such as delay, loss and throughput [11], [20]. [12]

Server-side QoE Modeling: [7], [6]

Client-side QoE Modeling: [28], [27] [23]

Router/AP level QoE Modeling: [9] [15] [10] **Spatial Quality Assessment:**

Temporal Quality Assessment: Several works [26], [8], [25], [21] investigated the effect of video freezes on user QoE. In []

VIII. Conclusion

References

[1] Amazon Mechanical Turk. https://www.mturk.com/.

[2] FFmpeg. ffmpeg.org/.

[3] Space Pirate Trainer. www.spacepiratetrainer.com/.

[4] theBlu: Encounter. wevr.com/project/theblu-encounter.

[5] Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Shobha Venkataraman, and He Yan. Prometheus: toward quality-of-experience estimation for mobile apps from passive network measurements. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, page 18. ACM, 2014.

[6] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. A quest for an internet video quality-of-experience metric. In *Proceedings of the 11th ACM workshop on hot topics in networks*, pages 97–102. ACM, 2012.

[7] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 339–350. ACM, 2013.

[8] Silvio Borer. A model of jerkiness for temporal impairments in video transmission. In *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*, pages 218–223. IEEE, 2010.

[9] Ayon Chakraborty, Shruti Sanadhya, Samir Ranjan Das, Dongho Kim, and Kyu-Han Kim. Exbox: Experience management middlebox for wireless networks. In *CoNEXT*, pages 145–159, 2016.

[10] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. Quantifying skype user satisfaction. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 399–410. ACM, 2006.

[11] Yanjiao Chen, Kaishun Wu, and Qian Zhang. From qos to qoe: A tutorial on video quality assessment. *IEEE Communications Surveys & Tutorials*, 17(2):1126–1165, 2015.

[12] Markus Fiedler, Tobias Hossfeld, and Phuoc Tran-Gia. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2), 2010.

[13] Cisco VNI Forecast. Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper. *Cisco Public Information*, 2017.

[14] S Alireza Golestaneh and Damon M Chandler. No-reference quality assessment of jpeg images via a quality relevance map. *IEEE signal processing letters*, 21(2):155–158, 2014.

[15] Shraboni Jana, An Chan, Amit Pande, and Prasant Mohapatra. Qoe prediction model for mobile video telephony. *Multimedia Tools and Applications*, 75(13):7957–7980, 2016.

[16] Xavier Marichal, Wei-Ying Ma, and HongJiang Zhang. Blur determination in the compressed domain using dct information. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, volume 2, pages 386–390. IEEE, 1999.

[17] Pina Marziliano, Frederic Dufaux, Stefan Winkler, and Touradj Ebrahimi. A no-reference perceptual blur metric. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages III–III. IEEE, 2002.

[18] media.xiph.org/video/derf/. Xiph.org Video Test Media.

[19] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, 21(12):4695–4708, 2012.

[20] Peter Orosz, Tamas Skopko, Zoltan Nagy, Pál Varga, and Laszlo Gyimothi. A case study on correlating video qos and qoe. In *Network Operations and Management Symposium (NOMS), 2014 IEEE*, pages 1–5. IEEE, 2014.

[21] Ricardo R Pastrana-Vidal and Jean-Charles Gicquel. Automatic quality assessment of video fluidity impairments using a no-reference metric. In *Proc. of int. workshop on video processing and quality metrics for consumer electronics*, 2006.

[22] Iain E Richardson. *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[23] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hobfeld, and Phuoc Tran-Gia. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17(1):469–492, 2015.

[24] Hanghang Tong, Mingjing Li, Hongjiang Zhang, and Changshui Zhang. Blur detection for digital images using wavelet transform. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, volume 1, pages 17–20. IEEE, 2004.

[25] Muhammad Arslan Usman, Soo Young Shin, Muhammad Shahid, and Benny Lövström. A no reference video quality metric based on jerkiness estimation focusing on multiple frame freezing in video streaming. *IETE Technical Review*, 34(3):309–320, 2017.

[26] Stephen Wolf and M Pinson. A no reference (nr) and reduced reference (rr) metric for detecting dropped video frames. In *Fourth International Workshop on Video Processing and Quality Metrics for Consumer Electronics, VPQM*, volume 5, page 3, 2009.

[27] Chenguang Yu, Yang Xu, Bo Liu, and Yong Liu. Can you see me now? a measurement study of mobile video calls. In *INFOCOM, 2014 Proceedings IEEE*, pages 1456–1464. IEEE, 2014.

[28] Xinggong Zhang, Yang Xu, Hao Hu, Yong Liu, Zongming Guo, and Yao Wang. Profiling skype video calls: Rate control and video quality. In *INFOCOM, 2012 Proceedings IEEE*, pages 621–629. IEEE, 2012.