Scalable Ground-Truth Annotation for Video QoE Modeling in Enterprise WiFi

Mallesham Dasari[§], Shruti Sanadhya[†‡], Christina Vlachou[†], Kyu-Han Kim[†], Samir R. Das[§]
[§]Stony Brook University, [†]HPE Labs, [‡]Facebook

***Abstract*—Mobile video traffic is dominant in cellular and enterprise wireless networks. With the advent of myriads of applications from video telephony and streaming to virtual reality, network administrators face the challenge to provide high quality of experience (QoE) in the face of diverse wireless conditions and application contents. Yet, state-of-the-art networks lack analytics for QoE, as this requires support from the application or user feedback. While there are existing techniques to map quality of service (QoS) to QoE by training machine learning (ML) models without requiring user feedback, these techniques are limited to only few applications (e.g., Skype), due to insufficient QoE ground-truth annotation for ML. To address these limitations, we focus on video telephony applications and model key artefacts of spatial and temporal video QoE. Our key contribution is designing content- and device-independent metrics and training across diverse WiFi conditions. We show that our metrics achieve a median 90% accuracy by comparing with mean-opinion-score (MOS) from more than 200 users and 800 video samples. Our content-independent metrics significantly reduce the MOS prediction error of previous works and are validated over three popular video telephony applications – Skype, FaceTime and Google Hangouts.**

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 reach 78% by 2021) [19]. This is due to the proliferation of 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. 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 and 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, enterprise network administrators seek to ensure that network resources are well provisioned and managed to provide good experience to multiple users of diverse applications. In this pursuit, network administrators can only rely on passive in-network measurements to *estimate* the exact user experience on the end-device. In this context, several prior works have developed a mapping between network Quality-of-Service (QoS) and end-user QoE for video applications [8], [7], [9], by using machine learning (ML) models and network features from PHY to TCP/UDP layers. ML models for QoE can be deployed at a number of vantage points in the network, such as access points, network controllers, cellular packet core.

In order to train any QoS to QoE model, one needs accurate ground-truth annotation of the QoE. To obtain this, prior work has leveraged application specific APIs such as Skype technical information [18] or YouTube API [7]; or instrumented client-side libraries [8]. While these solutions perform well for specific applications and providers, network administrators have to *deal with a plethora of video applications* and they *cannot control the application logic* for most applications. Nor does every application expose QoE metrics through APIs. Thus, to develop QoS to QoE models in the wild, network administrators need an application-independent approach to measure QoE.

In this work, we propose a generic video telephony QoE model which does not rely on application support. Additionally, it is scalable to diverse content, devices and categories of video application. We take a similar approach as Jana *et al* [11], where we record the screen on the mobile device and estimate video quality. Different from previous works, we exploit video compression methods and identify four new QoE metrics: *perceptual bitrate (PBR)*, *freeze ratio*, *length* and *number of video freezes*. We further demonstrate that our metrics are insensitive to the content of the video call and they only capture the *quality* of the video call. We analyze our model performance by conducting a large scale user-study of 800 video clips across 20 videos and more than 200 users. We conduct an extensive study on different types of users' devices and OS (Android vs. iOS), video content and motion.

We validate our metrics by mapping them to actual users' experience. To this end, we obtain Mean Opinion Score (MOS) from our user-study and apply ML models to map our metrics with users' MOS. We use `Adaboosted` decision trees in predicting the MOS scores, as we describe in Section IV.

In summary, our contributions are the following:

- We uncover limitations of existing work for QoE annotation of video telephony in wireless networks (Section II).
- We introduce new QoE metrics for video telephony that are content, application and device independent (Section III).
- We develop a model to map our QoE metrics to MOS and demonstrate a median 90% accuracy across applications and devices. We analyze each metric importance and reveal its correlation with the application (Section IV).

978-1-5386-2542-2/18/$31.00

II. Motivation for Scalable QoE Annotation

Lack of QoE information from applications: While several works have motivated and addressed the problem of network-based QoE estimation [7], little attention has been paid to the problem of collecting the QoE ground-truth. Most works have relied on application-specific information. This approach is effective for QoE optimizations by content providers, as they only focus on a single application [8] or initial training of QoS to QoE model [7]. Nevertheless, administrators of access networks have to ensure good experience for a wide range of diverse applications being simultaneously used. However, we find that not all applications provide QoE information. For instance, Hangouts and FaceTime does not provide while Skype gives technical information on some versions. To apply QoE estimation models in real networks, we need to remove the dependency of QoE metrics on application 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.

Lack of scalable and reliable QoE measures: Prior work on video quality evaluation leverages *subjective and/or objective* metrics. Subjective metrics are measured with MOS collected through user surveys. They capture absolute QoE but are tedious to conduct and to scale. QoE metrics need to scale to thousands of videos in order to train models that map QoS to QoE. Alternatively, objective metrics can be computed from the video. Objective metrics are further classified in two categories: reference and no-reference based. A reference-based metric uses both sent and received video, and compares the quality of sent vs. received frames. As it is challenging to retrieve and synchronize reference videos for telephony applications, no-reference based quality metrics are preferred. Jana *et al* [11] have proposed a no-reference metric for QoE estimation in Skype and Vtok. They record received videos for each of these mobile telephony applications and compute three no-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* 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 video content.

We conduct similar experiments with Skype to evaluate blur metric of prior works. Our experimental setup is described in Section III. To capture video blur, Jana *et al* [11] employ discrete cosine transform (DCT) coefficients [12] from the compressed data by computing the histogram of DCT coefficients thereby characterizing the image quality. The assumption here is that blurred image has high-frequency coefficients close to zero. Hence, the method studies 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 high luminance components. In

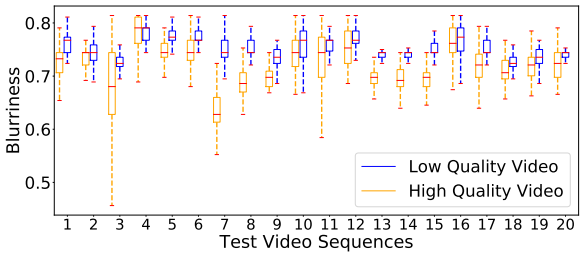

Fig. 1. Blur detection using DCT coefficients [11], [12]. DCT coefficients fail to distinguish between high and low quality video due to content diversity.

Fig. 1, we evaluate blur for 20 video sequences using DCT metric. We have collected these videos in a representative manner to cover diverse content, different types of motion and we have downloaded them in Full HD resolution. The same 20 videos are converted to low quality by compressing and decreasing the resolution, to observe the difference of DCT metric between high and low quality videos.

We notice that the blur 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, videos 2 and 4 contain mostly over-illuminated and dark images and are equally tagged as blurred in both low and high quality scenarios. DCT metric also fails to detect accurate blur levels, even if the image is blurred heavily i.e., it shows low blur differences ($<$0.2 blurriness) between high and low quality videos even though we created extremely low-quality videos (240p resolution and high ($>$200%) compression). Even for a single video, this DCT metric shows a lot of variation, such as for videos 3 and 11, raising concerns about its accuracy. We evaluate the accuracy of this blurriness metric in Section IV-B, showing a MOS error larger than 1.2. We also experiment with three other well-known blur metrics [10], [13], [16], but none of these methods are consistent across diverse videos. This challenges the scalability and versatility of this blur metric in the wild.

The *temporal variation* metric proposed by Jana *et al* [11] aims to capture video stalls and considers the ratio of missed frames to total number of frames in a video, but the metric requires the number of frames sent over the network. Thus, the metric is reference based. A QoE metric needs to be applicable to diverse contents and to not rely on access to reference video. The above limitations motivate us to propose new accurate metrics for blurriness and freezes across diverse video content.

To address the aforementioned requirements, we seek to answer the question: *How to scalably label the quality of a video call, without any support from the application?*

III. Design for Scalable QoE Annotation

A. Measurement Methodology

Set-up: In Fig. 2, we show our measurement set-up and QoE labeling framework. Since video telephony is an interactive application, it requires at least two participating network end-points (clients). In our set-up, we use a mobile device on our local enterprise WiFi network (Client 1) and an Amazon-provided instance as the second end-point (Client 2). Both

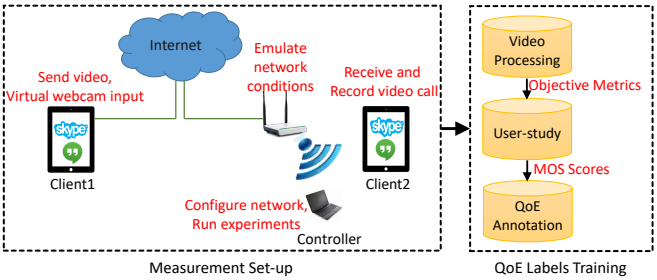

Fig. 2. Measurement set-up and system architecture. *Left:* Video telephony measurements and recording video calls. *Right:* Our framework processing recorded videos offline and extracting objective metrics to predict QoE labels.

clients can run Skype or Hangouts[1]. When the video call is placed from Client 1 to Client 2, Client 2 runs a virtual web-cam that inputs a video file in telephony application instead of camera feed; at Client 1, the video is received during the call. At the Amazon Client 2, we use *manycam* [20], a virtual web-cam tool. This tool can be used with multiple video applications in parallel for automation. In our LAN, a controller sits to emulate diverse network conditions and to run experiments on the connected mobile device through Android debug bridge (ADB) interface (via USB or WiFi connection). To automate the video call process, we use AndroidViewClient (AVC) library [4]. Once the received call is accepted, we start the screen recording of the video call session. The videos are recorded using Az screen recorder [3]. The recorded videos are sent to video processing module to calculate the objective QoE metrics. We use Ffmpeg [2] tool to extract our QoE metrics. Ffmpeg is a video framework with large collection of coding libraries. To validate our metrics and translate them into user experience, the videos are then shown to users to rate their experience. Finally, we retrieve MOS from all users and map our QoE metrics to MOS.

Network conditions: As we experiment with interactive applications (Skype, Hangouts, FaceTime), we need to emulate real user experience under any condition and to obtain samples with MOS values of all levels (1–5). Hence, we conduct tests with both ideal and throttled network. To create average and bad network conditions, we use Linux utility `tc` and we introduce hundreds of ms of delay, packet loss (10–20% ensuring that the call is setup), or low bandwidth (2–10 Mbps).

Test video sequences: We use the same 20 test video sequences as in Section II. We collect these videos from Xiph media [21] and YouTube. We select 20 representative videos that contain different types of motion and content.

B. Our QoE Metrics

PBR: Video bitrate has been considered a standard metric for perceptual quality of video. We compute the bitrate of recorded videos as a QoE metric. Typically, bitrate increases with the video resolution and depends on the level of video compression. Therefore, we capture video blur with `encoded bitrate` by compressing the recorded video. We employ the observation that the blurrier the video is, the higher compression efficiency is. However, we find that the bitrate metric is sensitive to motion in the video, because the block movement for high motion videos is very high and it is low for low motion videos. This results in different encoding bitrates. The low motion videos take advantage of inter-frame prediction, which makes the encoded bitrate motion-sensitive. Therefore, we use intra-coded bitrate by *disabling the inter frame prediction while compressing video.* We experiment with different encoding parameters like quantization parameter (QP) and de-blocking filter techniques and we choose high QP value (30) while coding, to get large bitrate difference when encoding high- and low-quality videos. To achieve robustness to video content, we compute the *relative change between the recorded bitrate and the intra-coded bitrate of the compressed video.* We define this change as the perceptual bitrate (PBR), as it only captures quality of the image. Since we calculate PBR on the recorded video, we do not need a reference video, hence PBR is a no-reference metric.

Freeze Ratio: Freeze ratio is the number of repeated frames over total frames in a given window, i.e., it denotes the amount of time the video is paused because of network disturbance. We use Ffmpeg's *mpdecimate* [2] filter in calculating freeze ratio. The *mpdecimate* algorithm works as follows: The filter divides the current and previous frame into 8x8 pixels blocks and computes sum of absolute differences (SAD) for each block. A set of thresholds (*hi, lo and frac*) are used to determine if the frames are duplicate. The thresholds *hi* and *lo* represent number of 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 yields SAD greater than a threshold of *hi*, and if no more than *frac* blocks have changed by more than *lo* threshold value. We experiment with several other error methods such as MSE and MAD, but we notice similar results among all three, thus we choose SAD for minimal computation overhead. We use threshold values of 64*12 for *hi*, 64*5 for *lo* and 0.1 for *frac* for all our experiments. 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 telephony applications do not generate such video content.

In addition to freeze ratio, we compute length and number freezes in video. We define a freeze if the video is stalled for more than one second. We notice that users do not perceive the stall when the length of freeze is very short or if there are few short freezes in original video. We employ `freeze ratio, length and number of freezes` metrics to capture the temporal artefacts of QoE.

IV. From Video Artefacts to QoE

We validate our QoE metrics presented in Section III with subjective measurements. We seek to ensure that our metrics accurately capture video telephony QoE artefacts. Video artefacts vary across applications and network conditions. Additionally, they can appear together (e.g., when we have both blurriness and stutter) or independently. Intermingled

[1]For FaceTime, we use 2 local clients (iPhone/iPad and MacBook Air) on different sub-networks, as creating virtual iOS and web-cam is challenging.

video artefacts result in diverse QoE levels (from bad to excellent), standardized as MOS by ITU-T [22][2]. We carry out a user study and obtain a MOS for each video.

A. Data Preparation and User-Study Setup

Our dataset is prepared from recorded video calls. We use 20 videos described in Section II, for video calls on Skype, FaceTime and Hangouts. For each video and application, we repeat experiments on multiple devices. The video calls are recorded under different network conditions to produce good, average and bad quality videos. We record video calls for Skype, Hangouts on Samsung Galaxy (SG) S6 edge, Google Pixel Tab, SG S2 Tab and for FaceTime on iPad Pro (model A1674) and iPhone 8 Plus. We post-process these videos into 30 seconds video clips and evaluate QoE for each clip. We host a web server with these video clips and ask the users to rate their experience after watching each sample.

We conduct the user-study in two phases: 1) in two labs, 2) online using a crowd sourced platform. The lab user study is conducted at a university campus and an enterprise lab. The online user study is conducted on Amazon Mechanical Turk platform [1]. Online users are from the United States and are in age range 20–40. We collect results from 60 lab users and more than 150 online users. The standard deviation of MOS between lab and online users is smaller than 1 MOS for 96% of videos. The user-study web page can be run on Chrome, Firefox and Safari. Once the user completes rating all the videos, results are pushed to our server. Amazon MTurk restricts users from taking again the same study, thus our users are unique.

B. Modeling Our Metrics to MOS

We now present our MOS prediction model that corroborates that our metrics accurately capture QoE artefacts and users' MOS. We build a model to map our objective metrics to subjective evaluations (MOS from user-study). Typically network administrators need to estimate a subjective evaluation such as MOS. However, QoE assignment in 5 classes can be cumbersome and may give little information on network quality. Hence, we map MOS scores to 3 classes (i.e., bad, average, good) that can be used by LTE or WiFi deployments to enhance resource allocation. We first explain our modeling methodology from freeze and blur metrics to MOS, and then describe how to translate MOS into 3-class QoE.

Typically, objective QoE metrics are mapped to MOS scores using non-linear regression [5]. Our regression model employs the average MOS from 15 users as ground-truth per video clip and it is based on ensemble methods. Taking the ensemble of models, we combine predictions of base estimators to improve generalizability and robustness over a single model. Moreover, a single model is always vulnerable to over-fitting and is complicated, which can be avoided in the form of ensemble of weak models. We employ boosting method, in particular `AdaBoost`[6], where the base estimators are built sequentially and one tries to reduce the bias of the combined estimator, thereby combining several weaker models and producing an accurate model. Drucker [6] has analyzed details of the Adaboost algorithm.

[2]MOS takes values from 1 to 5, with 5 being excellent.

TABLE I
MODELS PERFORMANCE ON SKYPE/FACETIME/HANGOUTS DATA

Model	Precision (%)	Accuracy (%)	Recall (%)	MSE
SVR	89.33/88.13/91.36	89.33/86.00/90.67	89.33/86.00/90.67	0.36/0.32/0.44
MLP	90.77/90.00/89.34	90.77/89.77/88.48	89.62/89.30/88.48	0.36/0.28/0.45
KNN	82.91/74.98/87.02	81.67/72.00/86.67	81.67/72.30/86.67	0.63/0.48/0.49
RF	89.72/90.37/91.22	89.34/90.00/90.67	89.34/90.00/90.67	0.41/0.28/0.43
ADT	92.21/90.28/93.75	92.00/90.00/93.33	92.00/90.00/90.67	0.29/0.26/0.38

TABLE II
MODEL PERFORMANCE ACROSS DEVICES FOR SKYPE/HANGOUTS

Devices		Model Performance		
Training	Testing	Precision (%)	Accuracy (%)	Recall (%)
SG-S6 Phone	SG-S6 Phone	89.90/90.03	88.57/90.00	88.57/90.00
	SG-S2 Tab	88.41/84.26	88.00/84.77	88.00/84.77
	Pixel Tab	88.52/82.86	87.62/82.00	87.62/82.00
SG-S2 Tab	SG-S6 Phone	83.73/89.61	84.43/89.21	83.00/89.21
	SG-S2 Tab	93.04/91.76	91.43/90.00	91.43/90.00
	Pixel Tab	82.69/85.41	82.86/84.77	82.86/84.77
Pixel Tab	SG-S6 Phone	84.43/86.79	84.00/86.00	84.00/86.00
	SG-S2 Tab	84.40/85.05	86.49/85.00	86.49/85.00
	Pixel Tab	88.20/86.17	94.40/86.67	94.00/86.67

Moving from 5 MOS scores to 3 classes, one has to estimate two thresholds m_1, m_2 in MOS (bad label: MOS $< m_1$, average: $m_1 \leq$ MOS $< m_2$, good: MOS $\geq m_2$). To this end, we first train the regression model and then search the MOS scores space with a sliding window of 0.05 for the two thresholds. We iterate over all such possible thresholds to maximize the accuracy of the trained model. We compute the true labels and prediction labels from the test MOS scores and predicted scores respectively using the corresponding thresholds in each iteration. We then select the thresholds which give highest accuracy from prediction labels. Therefore, our framework is divided into two phases: predicting MOS scores and labeling the scores with optimal thresholds.

We first evaluate our metrics by fitting five most common regressors: Support Vector Regressor (SVR), Random Forests (RF), Multi-Layer Perceptron (MLP), K-Nearest Neighbor (KNN) and Adaboosted Decision Tree Regressors (ADT). Each model is evaluated under 10 fold cross-validation. We perform a fine grid search to tune the hyper-parameters for all the models. We select the best parameters from the grid search and use the best estimator for the rest of the evaluations. The performance of each model is presented in terms of precision, accuracy and recall for three applications after labeling stage. We present micro-average metrics of accuracy, precision and recall, as the macro-average does not yield class importance. Micro-average aggregates the contributions from all the classes to compute the average metric. We also report the mean squared error (MSE) for all models. We observe at least 88% accuracy for all the models in all applications, except KNN regressor. This discrepancy is due to the weakness of KNN with multiple features. We observe that KNN does not generalize our dataset well and predicts most of the samples incorrectly. The mis-prediction is due the fact that each sample in higher dimension is an outlier as the distance metric (euclidean distance in our model) becomes weak with more features, unless

TABLE III
MODEL PERFORMANCE ACROSS DEVICES FOR FACETIME

Devices		Model Performance		
Training	Testing	Precision (%)	Accuracy (%)	Recall (%)
iPad Pro	iPad Pro	93.60	92.00	92.00
	iPhone 8 Plus	90.18	89.93	89.93
iPhone 8 Plus	iPad Pro	88.34	88.34	88.34
	iPhone 8 Plus	92.50	92.00	92.00

the data is well separated in all dimensions. Among all models, ADT has consistent and better accuracy in all applications with a maximum accuracy of 92% in Skype, 90% in FaceTime and 93.33% in Hangouts. We also use boosting with other models, but boosting did not improve the accuracy. Therefore, as ADT performs better than other models, we use this model for all the other evaluations unless otherwise specified. The best hyper-parameters for ADT from grid search are: `n_estimators = 10, learning_rate = 0.1` and `linear` loss.

Fig. 3 shows a scatter plot of user-study MOS vs. predicted MOS for the three applications. The MSE in MOS for the three applications is smaller than 0.4 with ADT for all the applications. We also observe 3 clear clusters approximately divided by the MOS scores $m_1 = 2$ and $m_2 = 4$, that coincide with our intuition for labeling the scores into 3 classes. This also justifies our design choice to finally employ three labels (bad, average, good) out of 5 MOS scores.

Model performance for Skype: Table I shows performance of Skype model across different regressors. The MSE in MOS is 0.29 in case of ADT and it is 0.63 with KNN regressor. Similarly, precision, accuracy and recall for ADT is the highest, while KNN being lowest. ADT model gives a best threshold of $m_1 = 2$ and $m_2 = 3.8$ in separating the MOS scores into labels. While all the other models produce a threshold of $m_1 = 2$ and $m_2 = 3.4$. Here, the best performance in ADT results from (i) its low MSE and (ii) the wide range for average class separation, i.e., it labels all the samples from MOS 2 to 3.8 as average class, while other models yield average class from MOS 2 to 3.4. The performance gap is due to the distribution of our average class samples spread over wider range of bad or good labels. Using $m_1 = 2$ and $m_2 = 3.8$ thresholds, we get 30%, 40% and 30% of our 300 samples in bad, average and good labels.

We further evaluate our model across three devices: SG-S6 phone, SG-S2 Tab and Pixel Tab. Table II shows precision, accuracy and recall for all three devices. We measure performance by training on one device, and testing on other devices. We observe that the performance is always better when trained and tested on the same device compared to training on one device and testing on other device. However, we find a difference of less than 7% in accuracy when trained and tested across different devices, with an accuracy of at least 83%. This corroborates that our model is robust across devices and it can be trained and tested without device constraints i.e., our metrics can collected on a certain device and can be applied to any other device for Skype.

Model performance for FaceTime: Table I shows performance of FaceTime model across different regressors. Similar to Skype, we observe similar performance ($> 89\%$ accuracy) for all models except the KNN regressor. Here, although the RF regressor is performing better (90.37% precision) than ADT, the MSE in MOS is larger than ADT. Interestingly, all models produce same thresholds of $m_1 = 2$ and $m_2 = 3.4$ in labeling the scores. Here, the samples are distributed uniformly across three classes unlike Skype, hence all regressors are performing almost equally. However, KNN regressor still suffers in FaceTime model due to weakness with many features as explained above. Using these thresholds, we get 30%, 36% and 34% of the 200 samples in bad, average and good labels.

To validate that our FaceTime model is device independent, we train and test across iPad and iPhone devices. Table III shows that when training and testing on same device, we observe 92% accuracy. Whereas, training and testing across devices yields at least 88% accuracy. Hence, we observe a difference of 4% accuracy across device training and testing. Our FaceTime model is also device-independent. Note that, we are not comparing the performance of our model training on Android devices and testing on iOS devices and vice-versa, because the recording set-up is different these environments.

Model performance for Hangouts: Table I shows performance of Hangouts model across different regressors. Similar to other applications, ADT outperforms other models with 93.33% accuracy with an average MSE of 0.38. Similar to FaceTime, we observe that all models produce same thresholds of $m_1 = 2$ and $m_2 = 3.5$ in labeling the scores. Using the above thresholds, we get 32%, 34% and 34% of the 300 samples in bad, average and good labels respectively.

We further evaluate the Hangouts model across three devices: SG-S6 phone, SG-S2 Tab and Pixel Tab. Table II shows that an accuracy of at least 86.67% when training and testing on same device, while inter-device train and test gives an accuracy of at least 82%. Overall, we observe less than 8% accuracy difference when trained and tested across different devices. Hence, the model is independent of the device used for training and testing.

C. Comparison with Baseline and Feature Importance

We compare our model prediction error with previous work for Skype application. As we need no-reference metrics for the baseline comparison, we use the DCT blur metric used by Jana *et al* as spatial metric and frame-drop metric in [17] as temporal metric. We fit the ADT model with these two metrics as well as with our metrics using the MOS scores from our user-study. Fig. 4 shows the performance of Skype application across the 20 videos described in Section II. Clearly, for a single video, both baseline and our model yield less than 0.2 MSE in MOS whereas as the number of videos increases, the MSE grows larger than 1.3 MOS for baseline metrics. Whereas, our model has a maximum of 0.4 MSE. The baseline metric's high MOS error with large number of videos is due to DCT metric's inability to scale across diverse video content. In fact, in our experiments we measure feature importance, which is defined as the amount each feature improves performance weighted by the number of samples this feature is responsible for. We observe that feature importance for DCT is as low as

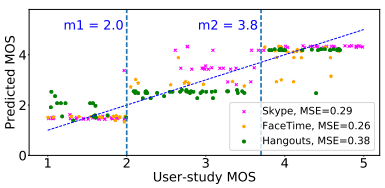

Fig. 3. User-study vs. predicted MOS.

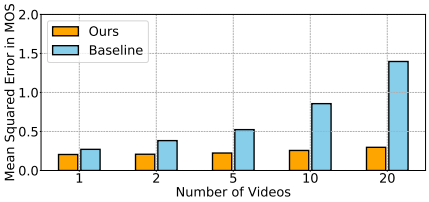

Fig. 4. Comparison of our metrics vs. baseline.

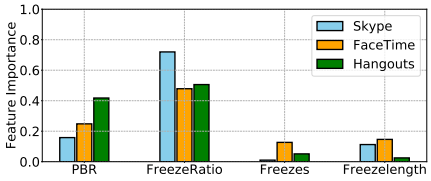

Fig. 5. Feature Importance for the three applications.

0.1, and it is 0.9 for frame-drop metric. Therefore, previous metrics fail to model video telephony QoE across diverse videos. We observe similar results for FaceTime and Hangouts.

We also evaluate the importance of each proposed metric. Fig. 5 shows the importance of features for ADT over all three applications. Out of all features, freeze ratio is dominating with at least 0.5 feature importance on all applications. For Skype, we observe highest (> 0.7) importance for freeze ratio. The rest of the metrics are almost equally important, and removal of any of these features causes an up to 0.2 accuracy degradation. Whereas, for FaceTime and Hangouts, we notice high importance for PBR (0.28 for FaceTime and 0.41 for Hangouts) due to their compromise in quality over freezes.

V. Related Work

Prior works [9] introduce machine learning methods to map QoS to QoE, and assume availability of the QoE ground-truth. Features for QoE estimation can be collected from various vantage points, such as the application server, the end-device (application-client statistics or packet capturing) [15], or the access network (e.g., WiFi AP or LTE base station) [11]. Compared to the above works, we focus on annotating QoE ground-truth, hence easing the extension of QoS to QoE mappings to all video-telephony applications.

Spatial quality assessment: Prior works have proposed multiple metrics to capture video blurriness. However, as we show in Section II, prior blur metrics [11] are highly content dependent. We find similar results for major prior works [10], [13], [16] on no-reference blur detection. In Section V, we present more than 1 MOS lower estimation error for our metrics compared to prior work over 20 diverse videos.

Temporal quality assessment: Several works have investigated the effect of video freezes on user QoE. Wolf and Pinson [23] propose to use the motion energy temporal history of videos, along with framerate of video. This requires additional information from original video hence, making it a reduced reference metric. Similarly, the temporal metric of Jana et al. [11] is a reduced-reference metric. The temporal metric by Pastrana-Vidal and Gicquel [14] is sensitive to resolution and content of the video. Different from all of these works, we present three content-independent, no-reference temporal metrics to assess temporal video artefacts.

VI. Conclusion

Network administrators need QoE information to model QoS to QoE relationship and to efficiently provision their resources. Currently, applications do not provide QoE ground-truth. In this work, we address this problem for video telephony by introducing four scalable, no-reference QoE metrics that capture spatial and temporal artefacts of video quality. We investigate the performance of our metrics over three popular applications – Skype, FaceTime and Hangouts. Finally, we map our metrics with a large-scale MOS user-study and show a median accuracy of 90% in annotating the QoE labels. Our metrics outperform state-of-the-art work while capturing exact user rating. We plan to extend this study to video streaming and VR/AR applications.

Acknowledgement

The Stony Brook authors are partially supported by NSF awards 1443951 and 1718014. The authors also acknowledge Yang Qiu's help with the data analysis.

References

[1] Amazon Mechanical Turk. https://www.mturk.com/.
[2] FFmpeg. ffmpeg.org/.
[3] http://az-screen-recorder.en.uptodown.com/android.
[4] Android View Cient (AVC). https://github.com/dtmilano/androidviewclient.
[5] Li Cui and Alastair R Allen. An image quality metric based on corner, edge and symmetry maps. In *BMVC*, pages 1–10, 2008.
[6] Harris Drucker. Improving regressors using boosting techniques. In *ICML*, 1997.
[7] Aggarwal et.al. Prometheus: toward quality-of-experience estimation for mobile apps from passive network measurements. In *ACM HotMobile*, 2014.
[8] Balachandran et.al. Developing a predictive model of quality of experience for internet video. In *ACM SIGCOMM*, 2013.
[9] Fiedler et.al. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2), 2010.
[10] Golestaneh et.al. No-reference quality assessment of jpeg images via a quality relevance map. *IEEE signal processing letters*, 21(2):155–158, 2014.
[11] Jana et.al. Qoe prediction model for mobile video telephony. *Multimedia Tools and Applications*, 75(13):7957–7980, 2016.
[12] Marichal et.al. Blur determination in the compressed domain using dct information. In *IEEE ICIP*, 1999.
[13] Mittal et.al. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing*, pages 4695–4708, 2012.
[14] Pastrana-Vidal et.al. Automatic quality assessment of video fluidity impairments using a no-reference metric. In *VPQM*, 2006.
[15] Seufert et.al. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials*, pages 469–492, 2015.
[16] Tong et.al. Blur detection for digital images using wavelet transform. In *IEEE ICME*, 2004.
[17] Usman et.al. 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.
[18] Zhang et.al. Profiling skype video calls: Rate control and video quality. In *IEEE INFOCOM*, 2012.
[19] Cisco VNI Forecast. Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper. 2017.
[20] Manycam. https://manycam.com/.
[21] media.xiph.org/video/derf/. Xiph.org Video Test Media.
[22] BT Series. Methodology for the subjective assessment of the quality of television pictures. *Recommendation ITU-R BT*, pages 500–13, 2012.
[23] Stephen Wolf and M Pinson. A no reference (nr) and reduced reference (rr) metric for detecting dropped video frames. In *VPQM*, 2009.