SenseQ: Context-aware Video Quality Adaptation for Optimal Mobile Video Streaming in Dynamic Environments

Journal:	*IEEE Internet of Things Journal*
Manuscript ID	IoT-29257-2023
Manuscript Type:	Regular Article
Date Submitted by the Author:	13-Apr-2023
Complete List of Authors:	RYOO, JIHOON; Stony Brook University, Computer Science Baek, Duin; Stony Brook University, Computer Science Kim, Youngchan; Stony Brook University The State University of New York, Computer Science
Keywords:	streaming, mobile adaptive streaming

SenseQ: Context-aware Video Quality Adaptation for Optimal Mobile Video Streaming in Dynamic Environments

Duin Baek, Youngchan Lim and Jihoon Ryoo
Computer Science Department
Stony Brook University
Stony Brook, NY 11794 USA
{duin.back,youngchan.lim,jihoon.ryoo}@stonybrook.edu

The growth of the mobile device and video streaming market has led to a significant increase in mobile video streaming as a primary mode of media consumption over the Internet. Mobile video streaming systems operate in dynamic environments and contexts that can change over time, posing challenges for the HVS(human visual system) and the quality of video streaming. To address these challenges in the context of IoT, we introduce SenseQ, a novel video viewing context-aware mobile video quality adaptation method that leverages various IoT data sources, such as sensor data and image data. By analyzing this data and comprehending the intricate mapping between context and perceived quality, SenseQ is able to minimize network usage while preserving comparable quality perception effectively. In an experiment involving 20 participants, we evaluated the effect of context on HVS quality perception. We found that SenseQ has promising potential in achieving comparable perceptual quality while reducing network usage in IoT-enabled mobile video streaming systems.

***Index Terms*—adaptive streaming, neural networks, machine learning, mobile video QoE**

I. Introduction

Video streaming has grown in various aspects in recent years— from the number of applications to its market size. In particular, mobile video streaming market has experienced an exponential growth, aligned with the advance in mobile devices and network infrastructures. According to the report published by Cisco [1], mobile videos accounted for 59% of mobile data traffic in 2017, and the forecast expects that 79% of mobile data traffic will be mobile video traffic by 2022. In addition, statistics show that more than 75% of world-wide videos and 70% of YouTube videos are played on mobile devices [2], [3]. Aligned with such high demand on video streaming services, accompanied network requirement is exponentially increasing. Although video streaming applications and network technologies are constantly evolving to keep up with the requirement, delivering the best video quality to end users still remains a challenging problem.

To address such a network requirement issue, many works proposed adaptive bitrate (ABR) video streaming methods that reduce the network requirement, while maintaining comparable video quality of experience (QoE) [4], [5], [6], [7], [8], [9], [10]. To achieve the objective, some focused on a single resource dimension– network capacity [4], [5], [10], while others extended their scope to both network capacity and compute resources in mobile devices [6], [7], [9], [8].

Though the existing works achieve their goal to reduce the network requirement and maintain or improve users' QoE by considering various resources present in video streaming systems, they are based on the implicit assumption on users' *static* video viewing environment. In other words, they mostly consider *indoor* conditions where users usually exhibit static movements such as sitting or lying.

In practice, however, mobile users can be situated in *dynamic* viewing conditions where they can exhibit dynamic movements or the environment surrounding them changes. In terms of dynamic movements, we can easily imagine a user who watches a video stream on his/her mobile device, while walking around a park or on a street. Nowadays, it is not uncommon to see such users in real life [11]. Regarding the environment, we can consider various factors such as light intensity that can change over the timing and location.

(a) Outdoor, Day

(b) Outdoor, Night

Figure 1. Effect of Surrounding Light Intensity on Users' Video Quality Perception

Such dynamic viewing environment can open an opportu-

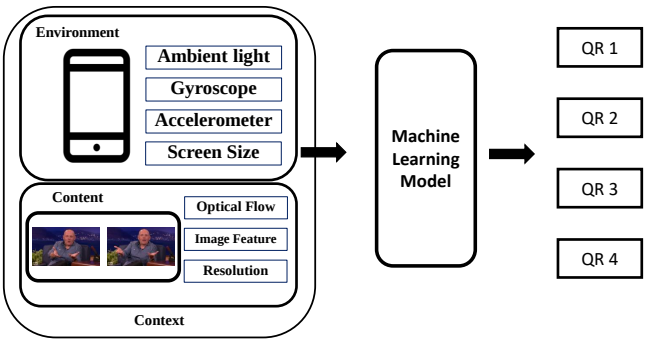


Figure 2. Overview of Quality Rating Prediction in SenseQ

nity for mobile video streaming systems to reduce the bandwidth requirement, while achieving users' comparable quality perception. As investigated in the literature [12], [13], users' video quality perception is not free from the environment where they are situated. For example, when an eye is fully adopted to a brighter environment, it loses accuracy of viewing low luminance stimuli. As a result, users can experience *wash out* effect. Figure 1 shows how light intensity or luminance, one of the most common environmental components affects users' quality perception. Compared to the night case (Figure 1(b)), we can see much more low luminance pixels on a mobile screen being washed out in day-time (Figure 1(a)). Thus, in such a bright day time, users may not discern a low-quality video from a high-quality one as most part of the detailed texture in the video content will not be recognized by the human visual system. In contrast, users may more acutely sense the quality difference in night.

Similarly, when users are moving and watching video streaming, such movement can degrade their visual focus on mobile screens or cause some visual artifacts on their perception [12], [13]. Thus, users find lower quality or resolution of videos more acceptable than they do in the static condition, as they do not notice the imperfectly rendered details [13].

Based on such a characteristic of the human visual system that can be less sensitive to video quality depending on users' dynamic viewing environment, we propose SenseQ, a machine learning (ML) based video quality adaptation method. By analyzing the video viewing context that consists of the *environment data* and the *video content*, SenseQ predicts users' quality perception on a video stream, and decides a video quality accordingly in such a way that minimizes the network usage, while maintaining a comparable quality perception. To this end, SenseQ abstracts this quality adaptation problem into a quality rating (QR) classification problem. Figure 2 shows the overview of QR prediction in SenseQ, including data sources and target label in the classification problem.

To understand how users interact with video viewing context, we first conduct a user study with our own developed mobile application that collects the context data of video streaming event and users' quality perception data needed for training a ML model. Given the dataset from the user study, we conduct an exploratory data analysis to gain an insight on how the video viewing context and users' quality perception correlate.

Utilizing various ML-based feature extraction techniques as well as statistical feature extraction, we represent the context data and video content as feature vectors. With the extracted features, we build a ML model that predicts users' quality rating given the video viewing context data. Evaluation results show that SenseQ has a potential to reduce the bandwidth requirement, while achieving comparable quality perception. In addition, SenseQ can be used to estimate how the quality rating distribution of a video changes over network conditions.

II. Background and Motivation

In this section, we provide a review of the existing works for video quality adaptation, by focusing on the resource or component in video streaming system that they consider.

A. Network-based Video Quality Control

To address the network requirement problem, video streaming systems use ABR algorithms to optimize the quality of experience. Even though it works efficiently by detecting a user's bandwidth and CPU capacity in real time and adjusting the quality of the video stream according, many works proposed improvements over the existing ABR algorithms only in the client side.

To understand the fundamental tradeoff between different approaches (e.g., rate- vs buffer-based) under different environments (e.g., low vs. high throughput variability), mpc[14] formulated a video bitrate adaptation as a stochastic optimal control problem, and identified some shortcoming in the existing approaches that consider only rate- or buffer-based strategies. Building on insights from the control-theoretic formulation, they proposed a model predictive control (MPC) algorithm that can optimally combine throughput and buffer occupancy information, and presented a practical implementation in a reference video player to validate their approach using realistic trace-driven emulations.

Focusing on the tradeoff between reducing the probability that the video freezes (rebuffers) and enhancing the quality of the video in bitrate adaptation, bola[5] formulate bitrate adaptation as a utility maximization problem, and devise an online control algorithm called BOLA that uses Lyapunov optimization to minimize rebuffering and maximize video quality. They prove that BOLA achieves a time-average utility that is within an additive term O(1/V) of the optimal value, for a control parameter V related to the video buffer size. Through the simulated network environment, they show that BOLA achieves near-optimal utility.

To address a key limitation in the preceding ABR algorithms that fixed control rules based on simplified or inaccurate models of other deployment environment, pensive[4] proposed Pensieve, a system that generates ABR algorithms using reinforcement learning (RL). Instead of relying on preprogrammed models or assumptions about the environment, Pensieve learns to make ABR decisions solely through observations of the resulting performance of past decisions. Such observation-based ABR decision is enabled by training a neural network model that selects bitrates for future video

chunks based on observations collected by client video players. Consequently, Pensieve learns ABR algorithms that adapt to various environments and QoE metrics.

B. Compute-based Video Quality Control

Though various advances have improved the QoE, majority of the works consider only a single resource dimension—*network capacity*, resulting in a fundamental limitation, where users with poor network capacity are always delivered a poor quality video.

To overcome the limitation, recent advances introduced a video quality enhancement paradigm that utilizes neural super-resolution (SR) models [9], [8], [6], [7] by leveraging client-side *compute capacity* as an additional resource. The key idea of super-resolution is to *reconstruct* a high-resolution video from its lower-resolution one.

As the pioneer work to utilize SR for video quality enhancement, nas[9] proposed NAS, a neural adaptive content-aware Internet video delivery that considers both network capacity and compute capacity in a client side. Inspired by video compression mechanisms such as H.264 or H.265, nemo[8] proposed an extension of NAS that improves quality enhancement speed significantly by applying SR process only to selective video frames.

To further improve quality enhancement speed, dcsr[7] proposed dcSR, a video quality enhancement method using data-centric super resolution. Based on data-centric AI paradigm, dcSR reduces SR workloads by replacing one big SR model with multiple small SR models, which results in the faster SR process time.

C. Context in Mobile Video Streaming

Unlike computer- or television-based video streaming systems, mobile video streaming systems contain another dimension to consider, *context* of video streaming events. As investigated in the existing works [12], [13], the context of video streaming events affects how users perceive the quality of video streams. Thus, understanding and utilizing the context of video streaming events can play an important role to design a better video quality adaptation method in mobile video streaming systems.

Fortunately, current mobile devices are equipped with various sensors such as accelerometer, gyroscope, and ambient light sensor. Thus, we can detect and estimate users' video viewing context using those sensors. For example, using gyroscope and accelerometer, we can analyze how fast users are moving or walking, which will affect how much visual artifacts human visual system perceives. Similarly, using ambient light sensor, we can measure surrounding luminance, which will affect the level of details in a video that the human visual system can perceive. In addition, recent advance in mobile hardwares allows us to deploy and run ML models on many mobile devices.

III. Experiment

To build a ML model that successfully predicts the desirable resolution of video streams, we need to collect a dataset that contains both video viewing context data and corresponding users' quality perception data. In this section, we present the way we collected those data, and some insights we could extract from them. Figure 3 shows the workflow of data collection. To enable automatic data collection, we set up cloud-based servers, and develop an android application.

A. Server-side Setup

To collect users' environmental context and quality perception data from as well as sending the video streaming to the end client devices, we set up a FTP server and a video streaming server.

1) Video Streaming Server

We set up the video streaming server on an Ubuntu machine, using Node.js [15]. In this experiment, we select in total 10 full high definition (FHD) videos available in YouTube. To enable an adaptive bitrate streaming, we encode each video into 4 different resolutions: 240p, 480p, 720p (HD), and 1080p (FHD) using Shaka Packager [16]. Along with the set of different resolutions of videos, we generate a media presentation description (MPD) file that describes how a video quality can be selected.

When selecting the videos for this experiment, we consider the *content dynamics* on top of the content categories. Out of 10 videos, a half of them contain dynamic scenes that involves frequent transitions and movements. The others, in contrast, contains relatively static scenes. We will discuss how the content dynamic affects users' quality perception in more depth in the later section.

2) FTP Server

We set up the FTP server on an Ubuntu machine, using vsftpd [17]. In the FTP server, both the environmental context data (e.g., accelerometer data sequence) and users' quality rating are stored separately for each video watch event.

B. Client-side Setup

In the client-side, we develop an android application that is responsible for three main functionalities: 1) displaying video streams, 2) collecting data, and 3) delivering the collected dataset to the server.

1) Video Streaming

To display a video stream on a mobile device, we integrate into our android application the **ExoPlayer**, an application level media player for Android [18]. Unlike Android's default MediaPlayer API, the ExoPlayer supports features such as DASH and SmoothStreaming adaptive playbacks.

2) Data Collection

In the client-side application, we collect three types of data in the following order:

1) Demographic information: name, age, gender, and the time when the experiment is executed.
2) Environmental context data: accelerometer (x, y, z axes), gyroscope (x, y, z axes), and ambient light.
3) Quality perception rating: between 1 and 4, with each score mapped to a video resolution (e.g., 1 to 240p and 4 to 1080p).

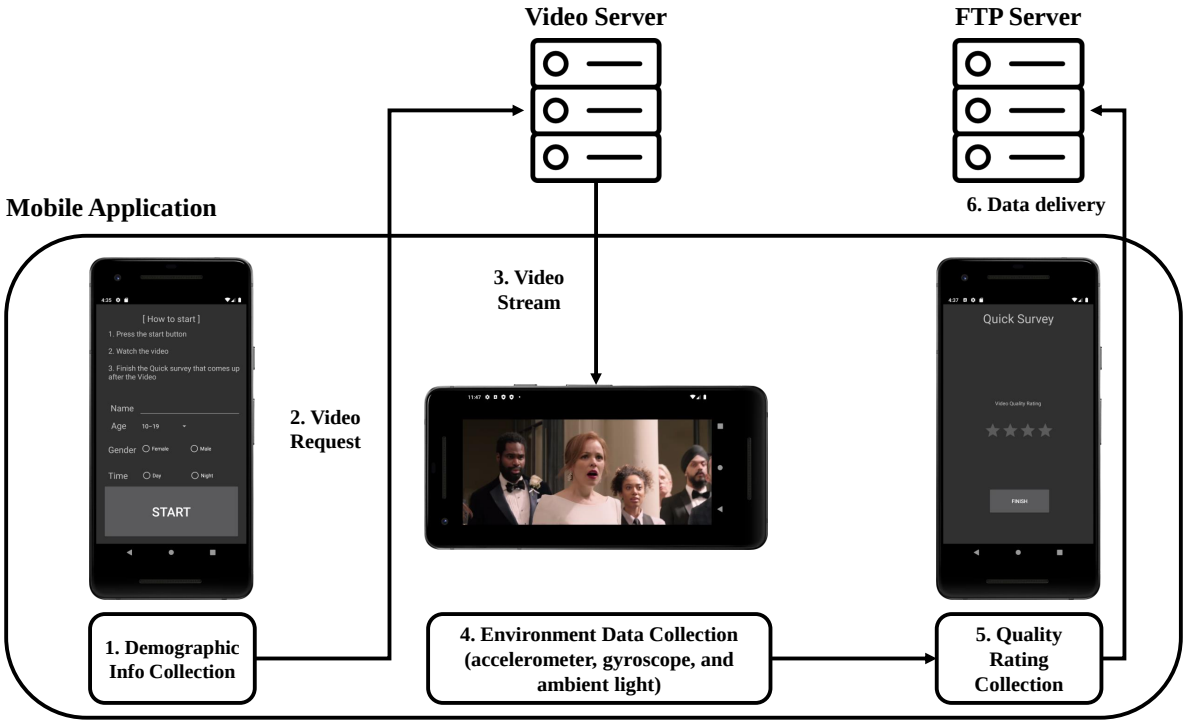
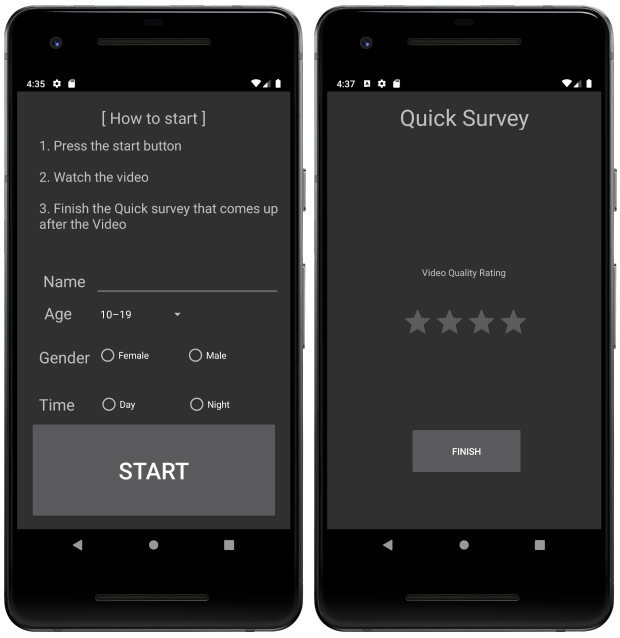


Figure 3. Workflow of Data Collection Experiment



(a) Demographic Info Collection (b) Quality Rating Collection

Figure 4. GUI of Android Mobile Application for Data Collection

When users first launch the application, they are asked to fill in or select their demographic information (name, age, and gender), along with the time when the experiment is executed. Figure 4(a) shows the graphical user interface (GUI) of the demographic information collection screen. This demographic information is collected only once at the beginning of the experiment session. After filling all the information, users can start the experiment by touching ”START” button in the screen.

When users watch a video stream on a mobile device, the environmental context around them is not constant. Thus, we let the application connect to various sensors available in mobile devices, and concurrently write the sensor values while displaying the video stream. Among various sensors available in the current mobile devices, we consider accelerometer, gyroscope, and ambient light sensors that are widely available in most mobile devices. Each type of sensor data is stored in a separate comma-separated values (CSV) file. Figure 5(a) and 5(b) show the structure of CSV files stored by the application, respectively.

Once a user completes watching the video stream, the application automatically transitions to the quality rating as shown in Figure 4(b). Users can rate their quality perception between 1 and 4, and touch ”FINISH” button, which automatically generates a text file that contains users’ demographic and quality rating information as shown in Figure 5(c). Note that we additionally collect the screen size of each mobile device that can affect users’ quality perception.

3) Dataset Delivery to the Server

Once the text file is generated in the quality rating process, the application automatically sends both the sensor CSV files and the text file to the FTP server. Once the data delivery is completed, the application starts a next video stream, and repeats the data collection until reaching the pre-defined stop count.

	time	x	y	z
0	2022-04-29 12:16:59.573 GMT+09:00	1.927332	6.251261	8.614338
1	2022-04-29 12:16:59.731 GMT+09:00	0.531513	5.310340	9.200917
2	2022-04-29 12:16:59.893 GMT+09:00	1.242591	4.836288	7.184999
3	2022-04-29 12:17:00.055 GMT+09:00	0.241814	5.114015	7.678205
4	2022-04-29 12:17:00.210 GMT+09:00	0.581791	5.758055	7.874529

(a) Accelerometer & Gyroscope

	time	data
0	2022-04-29 12:16:59.451 GMT+09:00	346722.0
1	2022-04-29 12:16:59.469 GMT+09:00	346666.0
2	2022-04-29 12:16:59.471 GMT+09:00	346611.0
3	2022-04-29 12:16:59.473 GMT+09:00	346555.0
4	2022-04-29 12:16:59.475 GMT+09:00	346500.0

(b) Light


```
{'Age': ' 20-29',
 'DisplaySize': ' 1080 x 2186',
 'Rating': ' 5.0',
 'Resolution': ' 720',
 'Sex': ' Female',
 'Time': 'day',
 'Video': ' dynamic_0'}
```


(c) Demographic & Quality Rating

Figure 5. Collected Data Format

C. Experiment Procedure

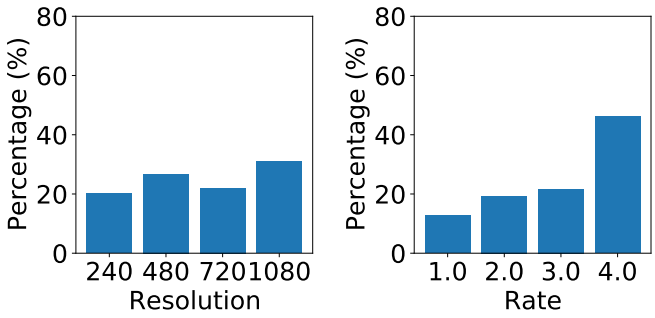
In the experiment, we recruited 20 volunteers, each of whom participated in two sessions of experiments (one in the daytime and the other in the night) to examine how the ambient luminance level affects users' quality perception. In each session, users are asked to watch 10 video streams, each of which is in 1 minute duration. As mentioned earlier, 5 video streams contain dynamic content, while the others do static content. The resolution of each video stream is randomly decided among 240p, 480p, 720p, and 1080p. In each session, we first provided participants with references on evaluating the video quality, and proceeded with the session. Note that the display of video streams is fixed to horizontal format. In this experiment, all the sensor data are collected at 20 Hz as suggested in [19]. However, as mentioned in [19], [20], the actual sampling rate can diverge from the rate we set (between 55% to 220% of the set rate).

IV. Exploratory Data Analysis

In this section, we analyze the dataset collected through the experiment to gain an insight on how the video viewing context and users' quality perception interact.

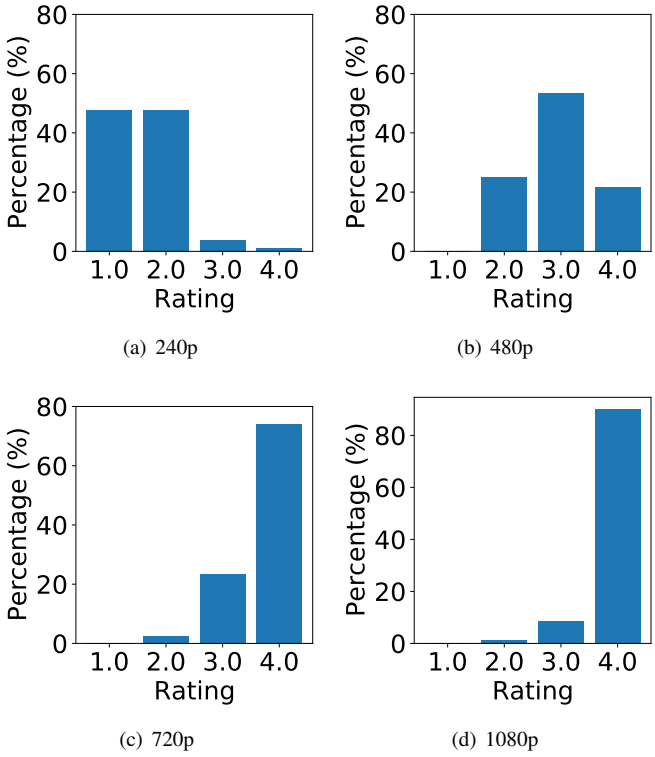
A. Video Resolution Distribution

As mentioned earlier, the android application randomly selects the resolution of a video stream among 240p, 480p, 720p, and 1080p. Figure 6(a) shows the distribution of video resolutions streamed in our experiment. Despite slight difference in the proportion, the video resolutions in our experiment are generally evenly distributed.



(a) Video Resolution (b) Quality Rating

Figure 6. Distribution of Video Resolution and Quality Rating



(a) 240p (b) 480p

(c) 720p (d) 1080p

Figure 7. Resolution and Quality Rating

B. Quality Rating Distribution

As mentioned earlier, we provided participants with references on evaluating video quality. Thus, we initially expected that the distribution of quality rating should be the same or at least similar to that of video resolutions. However, the distribution of quality rating collected through the experiment showed an unexpected distribution. Figure 6(b) shows that the highest rating (4.0) accounts for almost 50%, even though the best resolution occupies only 30% in the actual video resolution distribution. In contrast, the lowest rating (1.0) accounts for less than the proportion of 240p. This indicates that users may tend to over-evaluate the video quality when affected by their viewing context. To investigate where such gaps are from, we divide the distribution of the quality rating, depending on the video resolution. Figure 7 shows the distribution of quality

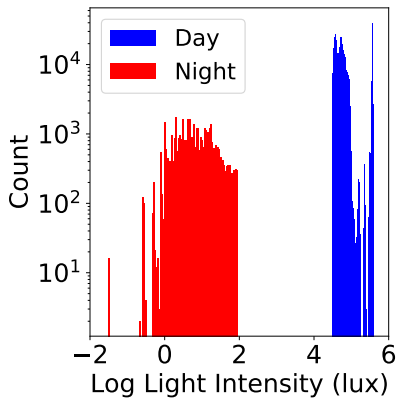


Figure 8. Distinction between Day and Night Light intensity

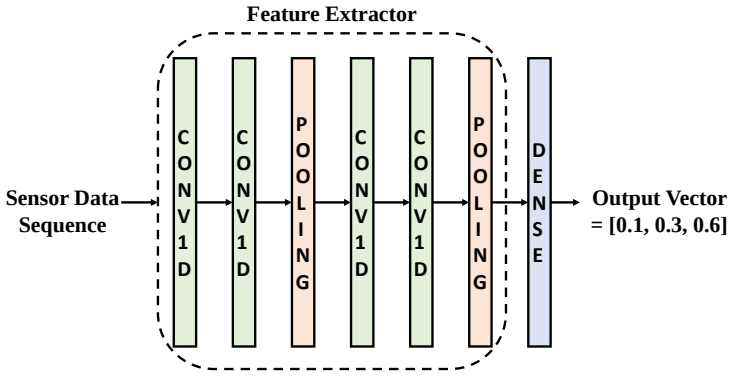


Figure 9. Feature Extractor for Sensor Data Sequence

rating over the resolutions.

Given 240p resolution, participants rate the quality in 1 or 2 at more than 90%. The high percentage of low quality rating (1 and 2) indicates that participants can roughly identify the video quality for 240p videos. However, the high proportion of quality rating 2 shows that many of them fail to evaluate the exact video quality. In this case, more than a half of ratings over-evaluate the video quality. In 480p resolution, the proportion of such over-evaluation increases. More than a half of ratings (75%) over-evaluate the video quality. Similarly, about 75% of quality ratings for 720p over-evaluate the video quality. In 1080p, most participants succeed in evaluate the exact video quality. Interestingly, no participant rate 480p, 720p, and 1080p in the lowest quality. Results show that participants show a tendency to over-evaluate the video quality across all the resolutions. Such over-evaluation occurs more frequently in the nearest better quality rating (e.g., 2.0 for 240p, 3.0 for 480p).

C. Ambient Light Distribution

As mentioned earlier, we consider an outdoor setting in our experiment. Unlike the human body movement that does not change over the time, the level of ambient light is significantly affected by the time. Figure 8 shows the clear distinction between the day light intensity and the night light intensity. Based on 2 log light intensity on the x-axis, the night measurement value and day measurement value are clearly divided on the left and the right side, respectively.

V. Feature Engineering

In mobile video streaming systems, there exist two types of data that can be used to predict users' quality perception: 1) environmental context data, and 2) video content itself. In this section, we describe how each data can be transformed into features that will be fed into an ML model.

A. Environmental Context

1) Body Movement

To extract features from users' body movement that can be represented with accelerometer and gyroscope time-series data, we can do either manual feature extraction or ML based feature extraction. In this work, we choose an ML based feature extraction to extract a complex and better representation that may not be captured by the manual feature extraction.

As the scale of dataset collected in our experiment is limited, we use a *transfer learning*, an ML technique that utilizes knowledge gained from one problem to solve a different but related problem, in order to extract feature from time-series sensor data. To build our feature extractor, we first build a multi-class classifier using the WISDM dataset [19]. The dataset contains accelerometer and gyroscope data collected from 51 participants, each of whom was asked to perform 18 activities, including walking, jogging, standing, and so on. Each participant performed each activity for 3 minutes, wearing a smartwatch on the dominant hand and having a smartphone in the pocket. Since we mainly focus on walk-based activities in this work, we narrow the scope of activities to three activities that align with our target context: standing, walking, and jogging. With the three classes, we build a 1D convolutional neural network model that shows a good performance in human activity recognition [21].

Although we train the neural network model to identify users' activity, our end goal is to extract feature vectors from sensor data. Thus, we use the output of an intermediate layer in the model, similar to the case we use the intermediate layers of the VGG-16 model [22] for image feature extraction. Figure 9 shows the structure of the activity classification model that consists of a feature extractor (convolutional layers and pooling layers) and a classifier (a dense layer). After completing to train the model, we use the output of the last pooling layer before the dense layer in order to extract movement feature vectors.

2) Ambient Light Intensity

As mentioned earlier, ambient light intensity, or luminance can affect how users perceive video quality, especially the level of detail in a video stream that users can visually detect. Although there can exist a certain level of variance, the luminance data collected by mobile devices shows relatively constant values, compared to accelerometer and gyroscope data. This can be inferred from the fact that it is uncommon for a user to be situated in where light sources dynamically

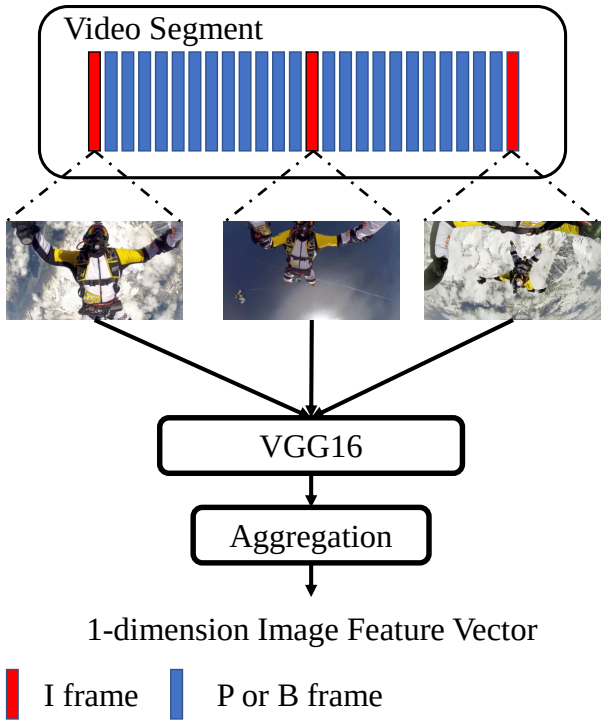


Figure 10. Image Feature Extraction for a Video Segment

change.

To extract features from the light intensity sequence, we calculate the **mean** and **standard** deviation from each light sequence. As mentioned earlier, the luminance data are relatively constant. Thus, we simply extract those two features that represent the general level of light intensity and the level of its consistency, respectively.

3) Display Size

In the existing works, it is mentioned that the display size of mobile devices can affect users' quality perception [12]. Thus, from each demographic and quality rating record, we extract **width** and **height** of the device display. To improve the predictive abilities beyond what width and height can provide individually, we multiply those features to form an additional feature that represents the **display area** of mobile devices.

B. Video Content

1) Image Feature

As mentioned in the literature [23], image features can affect how the human visual system evaluates the quality of images. For example, eyes can detect some minor changes in bright images, while being unable to discern the similar changes in dark images.

To extract image features, we may use traditional feature detection techniques such as Harris corner detection [24] and Scale-Invariant Feature Transform (SIFT) [25]. However, due to significant progress in deep learning based computer vision over the last decades, we can extract more complex features, which enables us to express images in the more semantic numeric representation.

Thus, in this work, we use **VGG-16** [22], one of the most well-known convolutional neural network models for image feature extraction. VGG-16 consists of two main parts: a feature extractor (convolutional layers) and a classifier (dense layers). Due to the limited size of image dataset in our experiment, we use a **transfer learning** with a pre-trained VGG-16 network. The pre-trained VGG-16 network is trained on 1.2 million images to classify them into 1000 object classes, which enables to learn rich feature representations for a wide range of images.

Since we focus on extracting image features, we use the pre-trained feature extractor that consists of 13 convolutional layers and 5 max pooling layers. Without any post-processing on the output of the feature extractor, the last convolutional layer outputs 3D vectors for each image. To transform the output of the feature extractor into a compatible shape with our ML model, we enable the optional pooling mode for the feature extractor. Between the global average pooling and the global max pooling, we use the former in the feature extractor.

As a video is a group of images (e.g., a 2-second-of video segment encoded at 30 FPS contains 60 images), we need to aggregate the group of extracted feature vectors, while maintaining the general feature of those image groups. To mitigate the compute burden in the feature extractor, we utilize one of the key concept in video compression: *temporal redundancy*. When there exists a sequence of similar video frames, we can say that there is temporal redundancy in video data.

Taking advantage of such redundancy, we apply the feature extraction process only to selective video frames. To select the frames for feature extraction, we take into account the way all the video coding formats utilize the temporal redundancy for video compression. In a video segment, I (intra) frames or key frames play as reference points for P (predicted) an B (bi-directional) frames. P and B frames contain small differences from I frames that they are referencing to. This similarity in the image space can be extend to the image feature space.

Thus, we extract the image features only from I frames in a video segment. Figure 10 shows how we extract image feature for a video segment. First, we select I frames in a video segment, and feed them to the pre-trained VGG-16 model, which outputs a 2D vector whose shape is in (number of I frames, feature dimension). Then, we aggregate the 2D vector into an 1D feature vector for the video segment by applying averaging or summation operation. In this work, we use the averaging operation for the aggregation.

2) Optical Flow

Unlike an image that contains the visual information at a particular moment, a video carries a sequence of visual information over time. Due to this sequential visual information, a video can project physical motions of objects onto 2D image plane. However, such motion can keep users from accurately identifying the detail in videos as the human visual system is limited in processing the temporal visual motion [26]. Consequently, motion in video content can affect users' quality perception [27].

Thus, we consider motion in our quality rating prediction problem. Since an ML model decides based on numerical

(a) Static Video (b) Static Video Optical Flow (c) Dynamic Video (d) Dynamic Video Optical Flow

Figure 11. Content Dynamics and Corresponding Optical Flows

values, we need to find a way to represent the motion in videos as numeric values. In this work, we use *optical flow* that can represent motions in videos with magnitude and direction.

In general, optical flow can be defined as the apparent motion of objects in a scene between two consecutive frames. In a broader spectrum, it can be extended to the motion of individual pixels on the image field. In many technologies including robotics and image processing, optical flow serves as a good approximation of the physical motion in the image field as it provides a concise description of both the regions of undergoing motions and their velocity in the image.

To calculate optical flow in video streams, we use the Gunnar Farneback's algorithm [28]. Given two consecutive video frames, the algorithm calculates the optical flow, which can be represented as two 2D vectors, each representing magnitude vector and angle vector, respectively. Note that those vectors have the same size as the input video frames. As a video contains a sequence of image frames, we can extract a sequence of optical flows (magnitude or angle), represented as a 3D vector.

Figure 13 shows clear distinction between the optical flow of a dynamic video and that of a static video. While the optical flows of the static video (Figure 11(b)) are focused on the center, those of the dynamic video (Figure 11(d)) are broadly distributed over the screen. In addition, Figure 11(d) has brighter colors than Figure 11(b), which means the magnitude of the optical flows in the dynamic video is larger than that in the static video.

As feeding all the sequence of optical flows in a video segment to an ML model can incur significant compute overhead, we need to compress the optical flow sequence into well-generalized representation. In this work, we extract a *mean* 1D vector and a *standard deviation (std)* 1D vector from a 3D optical flow sequence vector. By the mean operation, we can represent the average magnitude of movement in each video segment. The std vector describes the consistency of movement in each video segment. After applying the mean and std operations over the width and height axis of each optical flow vector, we can compress the 3D sequence vector into two 1D feature vectors.

VI. SenseQ Design

In this section, we describe the way we select an ML model architecture, and train the selected model. With all the features ready, we need to build a ML pipeline that predicts users' quality ratings. Figure 12 shows the ML pipeline for the quality rating prediction. As described earlier, the pipeline first extracts features from the sensor data and the video content data. Then, it concatenates all the feature matrices into one global feature matrix that will be fed into a ML model. As we consider 4 discrete quality ratings in this work, the probability vector output by the trained ML model contains four probability values, each of which is the probability for each quality rating.

A. Model Architecture Selection

To select the best-performing ML model architecture, we use neural network search (NAS) [29], a method that finds the optimal model architecture. Since it is significantly expensive and time-consuming to run the NAS, we use the AutoML [30], serviced by Google cloud platform [31]. In addition to the best working ML model architecture, the AutoML also searches the hyperparameters for the best performance.

Given our dataset, it took almost two hours for the AutoML to return the model type and hyperparameters. Table I shows the selected model type and its hyperparameters. In our experiment, the best-performing ML model is an extreme gradient boosting based model (XGB) [32] that has features such as parallelization, regularization, and non-linearity. The number of trees indicates the number of gradient boosted trees in the classifier. The max depth represents the maximum tree depth for base trees. the L1 and the L2 represent the L1 regularization term (XGB's alpha) and the L2 regularization term (XGB's lambda) on weights, respectively.

Model Type	# of Trees	Max Tree Depth	L1	L2
XGB	300	6	3	3

Table I
Model Type and Hyperparameters

B. Model Training

As a reminder of the scale of the experiment, we recruited 20 participants, and each participant performed two sessions of quality rating tests. In each session, a participant watched 10 1-minute-videos. After the experiment, we obtained 400 context data and quality rating data pairs. Since we take into account an on-demand video streaming system, we divide the data into a video-segment scale. In this work, we fix the video segment length to 5 seconds, taking into both encoding efficiency and quality [33]. After the division, the data size increases to 4,800 context and quality rating data pairs.

For each segment-scale sensor data and its corresponding video content data, we run the feature extraction process

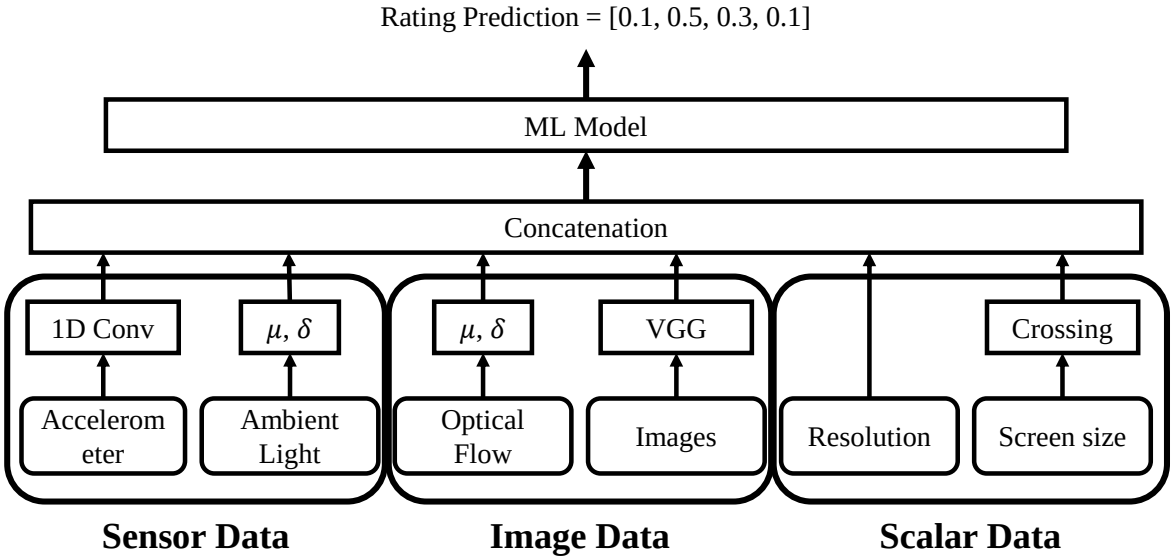


Figure 12. ML Pipeline for Quality Rating Prediction

described earlier, which outputs a feature vector in a shape of (826,). The feature vector consists of a set of vector type features and a set of scalar type features mentioned in Section V. Table II show the data shapes of the vector type features. Note that we exclude the gyroscope data since our empirical test shows its poor correlation to the quality rating. The set of scalar features consists of video resolution, screen width, screen height, screen area, light mean, and light std.

Name	Accelerometer	Image	Optical Flow
Shape	(180,)	(512,)	(128,)

Table II
VECTOR TYPE FEATURES

As a result of the feature extraction process, we obtain the feature matrix in a shape of (4800, 826) and the corresponding quality rating label in a shape of (4800, 1). We randomly split the dataset into a training and a test dataset by 80% to 20% ratio. To keep the same distribution of the target label (quality rating), we use the stratified sampling to split the dataset. As suggested by the AutoML, we build a XGBoost classifier with the hyperparameters in Table I, and train it with the training dataset.

VII. EVALUATION

In this section, we evaluate SenseQ's prediction capability and prediction speed. To investigate the chance of dimensionality reduction in SenseQ, we analyze the feature importance in the ML model. We implemented SenseQ in Python, using XGBoost library [34], an optimized gradient boosting library. Both training and testing process were conducted on the Google Colab's CPU-only virtual machine, equipped with the dual-core Intel Xeon CPU.

A. Prediction Capability

To evaluate the prediction capability, we consider the following performance metrics: 1) precision, 2) recall, 3) F1 score, and 4) ROC AUC. The precision represents the fraction of relevant instances (true positive) among the retrieved instances (positive predictions). As implied in its name itself, the recall represents the fraction of relevant instances (true positive) among the actual relevant instances (positive instances). Considering both precision and recall, F1 score shows a weighted average of the precision and recall, ranging its value from 0 (worst) to 1 (best). An ROC (receiver operating characteristic) curve shows the performance of a classifier at all classification thresholds. Combined with the ROC curve, an AUC (area under the ROC curve) represents an aggregate measure of classification performance across all possible classification thresholds.

Accuracy	Precision	Recall	F1	ROC AUC
86.8%	86.0%	85.5%	0.85	0.98

Table III
PREDICTION PERFORMANCE METRICS

Configured with the hyperparameters in Table I, the XGB classifier achieves the prediction performance metrics shown in Table III. The classifier achieves 86.3% of precision and 83.6% of recall. F1 score reaches 0.85, which can be calculated by the F1 score equation as shown below

$$F1 = 2 * \frac{precision * recall}{precision + recall}. \quad (1)$$

Figure 13(a) shows the ROC curve of the classifier, which achieves the AUC of 0.98. Such a high score indicates that the classifier can correctly predict the quality rating. To investigate how the model performs on each quality rating class, we build a confusion matrix. Figure 13(b) shows the confusion matrix of the XGB classifier. In the worst quality (1) and the best quality (4) rating prediction, the classifier shows excellent prediction capability, 92% and 94% accuracy respectively. Despite slight decrease of accuracy, the classifier still achieves 81% and 70% accuracy for 2 and 3 quality ratings. Such decrease is aligned

with the trend we found in the analysis on video resolutions and quality ratings.

B. Prediction Speed

Since we target the on-demand video streaming system, it is required for the classifier to predict the quality rating in a reasonably short time period so that it can serve video streaming uninterrupted. Table IV shows the statistics of feature processing time and prediction time. Even combining both preprocessing and prediction time together, SenseQcan complete all the computation within 0.05 second. Considering the length of video segments, we can expect to serve the quality rating prediction in real-time.

	Feature Preprocessing	Prediction
Mean	0.012(s)	0.034(s)
STD	0.005(s)	0.008(s)

Table IV
PREPROCESSING AND PREDICTION TIME

C. Feature Importance

In this work, we consider various sources of data to predict the quality rating. To investigate how each data source contributes to the prediction capability, we analyze the feature importance distribution by calculating importance scores for all the input features for our model. Figure 14 shows the distribution of importance values of all the features in the XGB classifier, sorted in the descending order.

Out of all the features, video resolution scores the highest feature importance value (0.44). We can infer that such high score can be attributed from the implicit information about the video content quality that the video resolution value has. Thus, even though the video resolution itself is a simple numeric value, it can be a meaningful feature. Image feature extracted by the VGG-16 model is ranked the second, making 0.25 feature importance value. Body movement feature and screen area show the similar feature importance. This result is consistent with the finding in [12], [13].

In the case of optical flow, we find a low feature importance. While the screen area is ranked the third, screen width and height show low correlation to the quality rating. This demonstrates the case where the feature crossing of features can provide more predictive capability than what those feature can provide individually. In contrast to the finding in [12], [13], the light intensity seems to have low correlation to the quality rating in this experiment.

D. SenseQ based Quality Adaptation

As SenseQ is designed to reduce the network usage in the comparable quality perception, we investigate how SenseQ can serve to save the network usage.

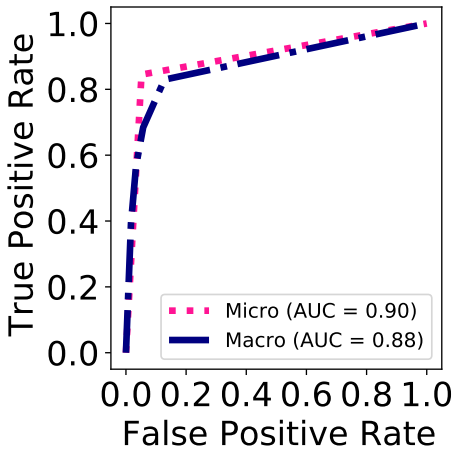
Given the test dataset, we generate synthetic dataset by replacing a subset of feature columns in the test feature matrix. We define a row in the feature matrix as a video viewing instance that has all the feature values such as image feature vector and optical flow feature vector. Since the video resolution is a key element to decide the network usage in the video streaming system, we first replace the video resolution in an instance. Corresponding to the changed video resolution, we also replace the image feature vector and the optical feature vectors in the instance with those of the new resolution video. Note that all the other features remain the same. By replacing those features, we can create a synthetic viewing instance that simulates the circumstance when a user watches the same video stream in the same viewing context as in the original instance, except for the video resolution.

For each instance in the test dataset, we generate synthetic instances in lower resolutions than the original instance. If the instance is in the lowest resolution, we skip the process. Given the synthetic test data, we use the pre-trained XGB classifier to predict their quality ratings. If there exists a prediction to suggest a lower resolution video segment that achieves the same quality rating as the original resolution segment, SenseQ selects the lower resolution video segment. When the classifier predicts multiple resolutions that satisfy the condition, SenseQ chooses the lowest resolution. We name this approach as SenseQ-0, as it does not allow any decrease in quality rating in the optimal case. When we allow to decrease 1 quality rating score, we name our approach as SenseQ-1. When allowing to decrease 2 quality rating score, we name our approach as SenseQ-2.

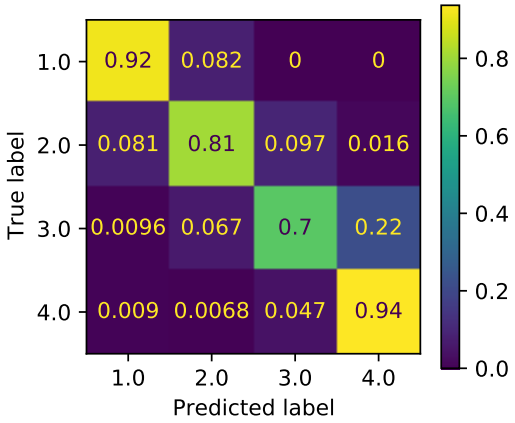
Figure15 compares SenseQ to the *context-agnostic* video streaming method (base). In the optical case when the classification accuracy is 100%, SenseQ-0 can achieve the same quality rating, while reducing the network usage by 25%. Since 100% accuracy is infeasible in practice, we estimate the expected quality rating by multiplying the predicted quality rating with the actual classification accuracy. Thus, we can expect SenseQ-0 to achieve 86% of the quality rating on average. SenseQ-1 and SenseQ-2 are expected to reduce the network usage by 59% and 81%, while achieving 75% and 57% of the original quality rating, respectively. The results show that SenseQ has a potential to reduce the network requirement, maintaining the comparable quality rating.

E. SenseQ based Rating Estimation under Different Network Conditions

Since SenseQ can predict the quality rating given the context data, we can utilize SenseQ to estimate how the quality rating distribution changes over network conditions. As in the synthetic data generation in Section VII-D, we modify the video resolution and corresponding image feature and optical flow features in each video viewing instance. However, in this experiment, we set the upper bound in the resolution that each instance can have. Given 1080p as the upper bound, we do not make any modification in the test dataset. In the case of 720p, we reduce the video resolution of all the 1080p instances down to 720p, and change their image feature and optical flow features accordingly. If an instance has a lower resolution than 720p, we do not make any changes in it. We apply the same principle to 480p and 240p, respectively. By using the pre-trained classifier, we can estimate users' quality rating and network usage over different network conditions.



(a) ROC Curve



(b) Confusion Matrix

Figure 13. ROC Curve and Confusion Matrix

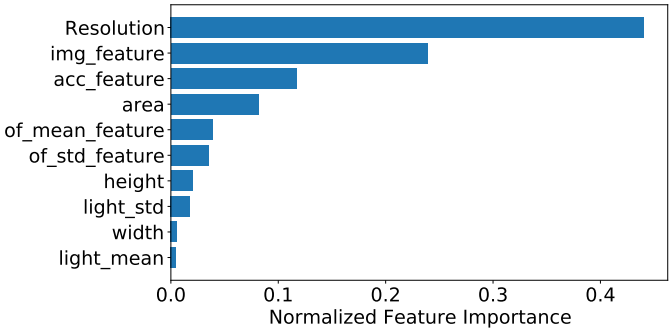


Figure 14. Normalized Feature Importance

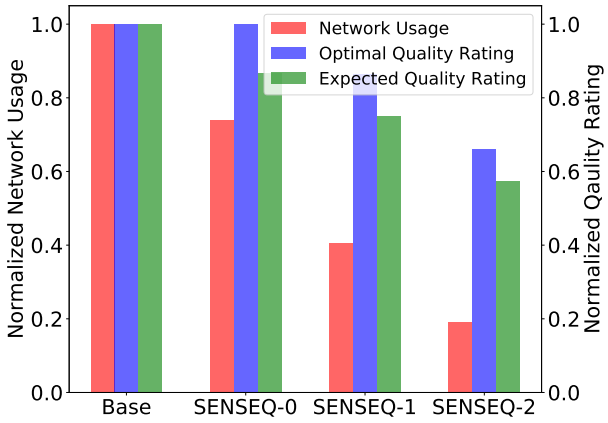


Figure 15. Network Saving and Expected Quality Rating

Figure16 shows the expected quality rating and network usage over different network conditions represented as video resolutions. The video resolutions on the x-axis of Figure 16 represent the best video resolution that can be served in the given network conditions. For example, in the case of 1080p, we assume that current network condition can support the best resolution video stream. In contrast, in the case of 480p, we assume that current network condition is throttled to support maximum 480p video stream. In the case of 720p, the estimation results show that the video streaming service can save the network usage, but it will suffer the loss of quality rating as much as the gain in the network usage. Interestingly, the results show that 480p may save significant portion of network usage, while losing slight quality rating. Though 240p achieves the highest portion of network saving, it may suffer significant quality perception drop.

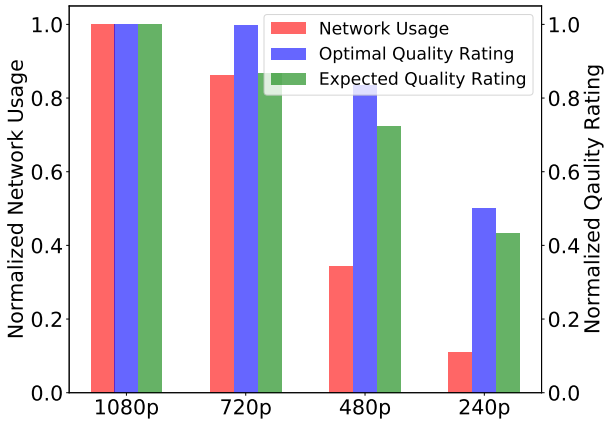


Figure 16. QR Estimation over Different Resolutions

VIII. Conclusion

We proposed SenseQ, a video viewing context-aware mobile video quality adaptation method that adjusts the video quality, by analyzing various data available in the mobile video streaming system. By understanding the complex relationship between the context data and users' quality perception, SenseQ can reduce the network usage in mobile video streaming systems, while maintaining the comparable quality perception. In the evaluation, we have demonstrated that SenseQ shows a potential for the network saving, maintaining the comparable quality rating. In addition, the prediction capability of SenseQ enables us to utilize SenseQ to estimate the network usage and users' expected quality rating over the different network

scenarios. Since 1080p resolution is currently used as the best video quality in on-demand video stream systems, we believe that our scope of video resolutions ranging up to 1080p in the experiment is reasonable enough to reflect the current mobile video streaming systems. However, the scope of SenseQ is not limited in the FHD, and can be seamlessly extended to 4K and 8K resolutions, where we can expect further bandwidth saving. There are still multiple technical challenges that will need a deeper consideration before the context-aware mobile video quality adaptation can be widely used in practice. As an invitation for future collaboration to address such challenges, we are opening our source code to research colleagues.

REFERENCES

[1] Cisco, "Cisco visual networking index:global mobile data traffic forecast update, 2017–2022," http://media.mediapost.com/uploads/CiscoForecast.pdf, 2019.
[2] YouTube, "Youtube for press: Youtube in numbers," https://www.youtube.com/intl/en-GB/about/press, 2020.
[3] eMarketer, "More than 75% of worldwide video viewing is mobile," https://www.emarketer.com/newsroom/index.php/threequarters-video-viewing-mobile/, 2018.
[4] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," ser. SIGCOMM '17. Association for Computing Machinery, 2017, p. 197–210.
[5] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, p. 1698–1711, Aug. 2020.
[6] M. Dasari, A. Bhattacharya, S. Vargas, P. Sahu, A. Balasubramanian, and S. R. Das, "Streaming 360-degree videos using super-resolution," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1977–1986.
[7] D. Baek, M. Dasari, S. R. Das, and J. Ryoo, "Dcsr: Practical video quality enhancement using data-centric super resolution," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 336–343.
[8] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, "Nemo: Enabling neural-enhanced video streaming on commodity mobile devices," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20. Association for Computing Machinery, 2020.
[9] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. Carlsbad, CA: USENIX Association, 2018, pp. 645–661. [Online]. Available: https://www.usenix.org/conference/osdi18/presentation/yeo
[10] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "Elastic: A client-side controller for dynamic adaptive streaming over http (dash)," in *2013 20th International Packet Video Workshop*, 2013, pp. 1–8.
[11] G. Marcus, "People watch netflix while walking, apparently, and the company wants their data," https://mashable.com/article/netflix-while-walking, 2019.
[12] J. Xue and C. W. Chen, "Mobile video perception: New insights and adaptation strategies," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 3, pp. 390–401, 2014.
[13] O. Machidon, T. Fajfar, and V. Pejovic, "Watching the watchers: Resource-efficient mobile video decoding through context-aware resolution adaptation," ser. MobiQuitous '20. Association for Computing Machinery, 2020, p. 168–176.
[14] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 325–338.
[15] R. Dahl, "Node.js," https://github.com/nodejs/node, 2009.
[16] J. Parrish, "Shaka packager," https://github.com/shaka-project/shaka-packager, 2022.
[17] C. Evans, "very secure ftp daemon," https://security.appspot.com/vsftpd.html, 2022.
[18] Google, "ExoPlayer: An extensible media player for Android," https://github.com/google/ExoPlayer, 2019.
[19] G. M. Weiss, K. Yoneda, and T. Hayajneh, "Smartphone and smartwatch-based biometrics using activities of daily living," *IEEE Access*, vol. 7, pp. 133 190–133 202, 2019.
[20] A. D. Document, "Sensordirectchannel," https://developer.android.com/reference/android/hardware/SensorDirectChannel, 2022.
[21] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors." IEEE, 11 2014.
[22] S. Karen and Z. Andrew, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR*, 2015.
[23] J. Korhonen, "Assessing personally perceived image quality via image features and collaborative filtering," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8161–8169.
[24] H. Chris and S. Mike, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.
[25] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, p. 91–110, nov 2004.
[26] B. G. Borghuis, D. Tadin, M. J. Lankheet, J. S. Lappin, and W. A. van de Grind, "Temporal limits of visual motion processing: Psychophysics and neurophysiology," *Vision*, vol. 3, no. 1, 2019.
[27] J. You, T. Ebrahimi, and A. Perkis, "Modeling motion visual perception for video quality assessment," in *Proceedings of the 19th ACM International Conference on Multimedia*, ser. MM '11. Association for Computing Machinery, 2011, p. 1293–1296.
[28] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*. Springer Berlin Heidelberg, 2003, pp. 363–370.
[29] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
[30] G. Cloud, "Automl, train high-quality custom machine learning models with minimal effort and machine learning expertise." https://cloud.google.com/automl, 2022.
[31] ——, "Google cloud: Cloud computing services," https://cloud.google.com, 2022.
[32] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. Association for Computing Machinery, 2016, p. 785–794.
[33] S. Lederer, "Optimal adaptive streaming formats mpeg-dash hls segment length," https://bitmovin.com/mpeg-dash-hls-segment-length/, 2020.
[34] DMLC/XGBoost, "Xgboost python package introduction," https://xgboost.readthedocs.io/en/stable/python/python_intro.html, 2022.

APPENDIX