SENSEI: Aligning Video Streaming Quality with Dynamic User Sensitivity

Xu Zhang
University of Chicago

Yiyang Ou
University of Chicago

Siddhartha Sen
Microsoft Research

Junchen Jiang
University of Chicago

Abstract

This paper aims to improve video streaming by leveraging a simple observation—users are more sensitive to low quality in certain parts of a video than in others. For instance, rebuffering during key moments of a sports video (*e.g.,* before a goal is scored) is more annoying than rebuffering during normal gameplay. Such *content-dependent* dynamic quality sensitivity, however, is rarely captured by current approaches, which predict QoE (quality-of-experience) using one-size-fits-all heuristics that are too simplistic to understand the nuances of diverse video content.

The problem is that none of these approaches know the true dynamic quality sensitivity of a video they have never seen before. Therefore, instead of proposing yet another heuristic, we take a different approach: we run a *separate crowdsourcing experiment for each video* to derive users' quality sensitivity at different parts of the video. Of course, the cost of doing this at scale can be prohibitive, but we show that careful experiment design combined with a suite of pruning techniques can make the cost negligible for content providers. For example with a budget of just $31.4 per min video, we can predict QoE 37.1% more accurately than recent QoE models.

Our ability to accurately profile time-varying user sensitivity inspires a new approach to video streaming—*dynamically aligning higher (lower) quality with higher (lower) sensitivity periods*. We present a new video streaming system called SENSEI that profiles and incorporates dynamic quality sensitivity into existing quality adaptation algorithms. We apply SENSEI to two state-of-the-art adaptation algorithms, one rule-based and one based on deep reinforcement learning. SENSEI can take seemingly unusual actions, *e.g.,* lowering quality even when bandwidth is sufficient to prepare for higher quality sensitivity in the near future. Compared to state-of-the-art approaches, SENSEI improves QoE by 15.1% or achieves the same QoE with 26.8% less bandwidth on average.

1 Introduction

An inflection point in Internet video traffic is afoot, driven by more ultra-high resolution videos, more large-screen devices, and ever-lower user patience for low quality [2, 10]. At the same time, the video streaming industry, after its several decades of evolution, is seeing diminishing improvements: recent adaptive bitrate (ABR) algorithms (*e.g.,* [45, 56, 83]) achieve near-optimal balance between bitrate and rebuffering events, and recent video codecs (*e.g.,* [54, 72]) improve encoding efficiency but require an order of magnitude more computing power than their predecessors. The confluence of these trends means that the Internet may soon be overwhelmed by online video traffic,[1] and new ways are needed to attain fundamentally better tradeoffs between *bandwidth usage* and user-perceived *QoE* (quality of experience).

We argue that a key limiting factor is the conventional wisdom that users care about quality in the same way throughout a video, so video quality should be optimized using the same standard *everywhere* in a video. This means that lower quality—due to rebuffering, low visual quality, or quality switches—should be avoided identically from the beginning to the end. We argue that this assumption is not accurate. In sports videos (*e.g.,* the one in Figure 1), a rebuffering event that coincides with scoring tends to inflict more negative impact on user experience than rebuffering during normal gameplay. But there are also sports videos where scoring is not the most quality sensitive part. Thus, user quality sensitivity *varies with the video content dynamically over time*.

Unfortunately, both the literature on ABR algorithms and the literature on QoE modeling adopt the conventional wisdom. Most ABR algorithms completely ignore the content of each video chunk: they focus on balancing high bitrates and low rebuffering times, and thus consider only the size and download speed of the chunks. Traditional ways of modeling QoE are also agnostic to the substance of videos, although recent QoE models—*e.g.,* PSNR [38], SSIM [80], VMAF [11], and deep-learning models [33, 48]—try to find frames that users are more sensitive to by studying the structure of pixels and motions to gauge their saliency. These heuristics seek to

[1] This is vividly illustrated by the recent actions taken by YouTube and Netflix (and many others) to lower video quality in order to save ISPs from collapsing as more people stay at home and binge watch online videos [12].

generalize across *all videos* and thus resort to generic measures (like pixel-level differences), but it is unclear if any heuristic can capture the diverse and dynamic influence a video's content can have on users' sensitivity to quality.

For example, models like LSTM-QoE [33] assume that users are more sensitive to rebuffering events in more "dynamic" scenes. In sports videos, however, non-essential content like ads and quick scans of the players can be highly dynamic, but users may care less about quality during those moments. In the video in Figure 1, LSTM-QoE considers normal gameplay to be the most dynamic part, but the most quality sensitive part of the video according to the user study is the goal. A key insight is that the impact of the substance of a video on user's sensitivity of quality cannot be fully explained by pixel-level patterns or cross-frame motions. Some recent work tries to predict user's dynamic sensitivity, but they either need access to users' viewing history [35] or use off-the-shelf computer-vision saliency models [34] whose predictions have little correlation with quality sensitivity on videos they have never seen before (§2.3 elaborates on this).

The dynamic nature of quality sensitivity suggests new room for improvement. One can achieve *higher QoE with the same bandwidth* by carefully lowering the current quality in order to save bandwidth and allow higher quality when users become more sensitive. Similarly, one can attain *similar QoE with less bandwidth* by judiciously lowering the quality when quality sensitivity is indeed low. In short, we seek to *align higher (lower) quality of video chunks with higher (lower) quality sensitivity of users*.

We present SENSEI, a video streaming system that incorporates dynamic quality sensitivity into its QoE model and video quality adaptation. SENSEI addresses two key challenges.

Challenge 1: How do we profile the unique dynamic quality sensitivity of each video in an accurate and scalable manner?

Crowdsourcing the true quality sensitivity per video: Instead of proposing another heuristic, SENSEI takes a different approach. We run a *separate crowdsourcing experiment for each video* to derive the quality sensitivity of users at different parts of the video. Specifically, we elicit quality ratings directly from real users (obtaining a "ground truth" of their QoE) for multiple renderings of the same video, where each rendering includes a quality degradation in some part of the video. SENSEI automates and scales this process out using a public crowdsourcing platform (Amazon MTurk), which provides a large pool of raters, while using pruning techniques to reduce the number of rendered videos that need to be rated. We then use these ratings to estimate a *weight* for each video chunk that encodes its quality sensitivity, independent of the quality of other chunks. While crowdsourcing has previously been used to model QoE, SENSEI is to our knowledge the first to scale it to *per-video* QoE modeling.

Challenge 2: How do we incorporate dynamic quality sensitivity into a video streaming system to enable new decisions? Today's video players are designed to be "greedy": they pick a bitrate that maximizes the quality of the next chunk while avoiding rebuffering events. But in order to utilize dynamic quality sensitivity, a player must "schedule" bitrate choices over *multiple* future chunks, each having a potentially different quality sensitivity. This means that some well-established behaviors of video players, *e.g.,* only rebuffer when the buffer is empty, may need to be revisited.

Refactoring ABR logic to align with dynamic quality sensitivity: SENSEI works within the popular DASH framework. It integrates the aforementioned per-chunk weights into existing ABR algorithms to leverage the dynamic quality sensitivity of upcoming video chunks when making quality adaptation decisions. The per-chunk weights enable new adaptation actions that "borrow bandwidth" from low-sensitivity chunks and give them to high-sensitivity chunks. For example, SENSEI may lower the bitrate even when bandwidth is sufficient, or initiate a rebuffering event with a non-empty buffer, to afford higher bitrates when quality sensitivity becomes higher. We apply SENSEI to two state-of-the-art ABR algorithms: Fugu [83], a more traditional rule-based algorithm, and Pensieve [56], a deep reinforcement learning-based algorithm.

Using its scalable crowdsourcing approach, SENSEI can predict QoE more accurately than state-of-the-art QoE models. For example, with a budget of just $31.4 per min video, SENSEI achieves 55% less QoE prediction errors than existing models. Compared to state-of-the-art ABR algorithms, SENSEI improves QoE on average by 15.1% or achieves the same QoE with 26.8% less bandwidth on various video genres.

Contributions and roadmap: Our key contributions are:

- A measurement study revealing substantial temporal variability in users' quality sensitivity and its potential for improving video streaming QoE and bandwidth usage (§2).
- The design and implementation of SENSEI, including: 1) a scalable crowdsourcing solution to profiling the true dynamic quality sensitivity of each video (§4,§5),[2] and 2) a new ABR algorithm that incorporates dynamic user sensitivity into existing algorithms and frameworks (§6).

2 Motivation

We begin by showing that existing approaches to modeling video streaming QoE fail to accurately capture the true user-perceived QoE (2.1). We then present user studies that reveal a missing piece in today's QoE modeling: users' quality sensitivity varies dynamically throughout a video (2.2), and this dynamic quality sensitivity is hard to capture using prior heuristics or vision models (2.3). However, by incorporating dynamic quality sensitivity into existing ABR algorithms, we can significantly improve QoE and save bandwidth (2.4).

2.1 Prior QoE modeling and their limitations

QoE models are crucial to modern video streaming systems. A QoE model takes a streamed video as input and returns a

[2] Our study was IRB-approved (IRB18-1851).

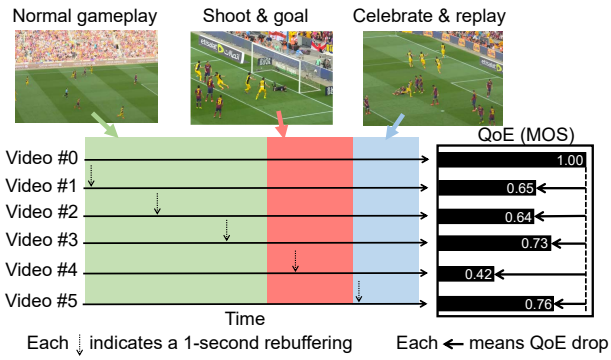


Figure 1: *Example of dynamic quality sensitivity. Users are asked to rate the quality (on a scale of 1 to 5) of different renderings of a source video (`Soccer1`), where a 1-second rebuffering event occurs at a different place in each rendering. We observe substantial differences in the QoE impact (measured by mean opinion score, or MOS) across the renderings.*

predicted QoE as output. When streaming a video, the video player optimizes the QoE by adapting the bitrate of each video chunk to the available bandwidth. QoE is often measured by the mean opinion score (MOS) assigned by a group of users to the quality of a video.[3]

Quality metrics: Today's QoE models consider two aspects.

- *Pixel-based visual quality* tries to capture the impact of visual distortion on QoE. These metrics, such as PSNR and VMAF, are based on pixel/motion-based patterns [11, 38, 48, 63, 68, 79, 80] and recently on visual attention [27, 44, 86].
- *Streaming quality incidents* during the streaming process can negatively impact user experience, such as rebuffering, low bitrate, and bitrate switches. Their impact is modeled by metrics, such as rebuffering ratio, average bitrate, and frequency of bitrate switches during a video (*e.g.,* [18, 28]).

Some work also considers contextual factors (*e.g.,* viewer's emotion, acoustic conditions, etc.), but these are orthogonal to our focus on the video's content.

QoE models: Recent QoE models combine both pixel-based visual quality metrics and quality-incident metrics to for more accurate QoE prediction. We consider three such QoE models: KSQI [29], P.1203 [66], and LSTM-QoE [33], which were proposed within the past two years and have open-source implementations. KSQI combines VMAF, rebuffering ratio, and quality switches in a linear regression model. P.1203 combines QP (quantization parameter) and quality incident metrics in a random-forest model. Most recently, LSTM-QoE takes STRRED [68] and individual quality incidents as input to a long short-term memory (LSTM) neural network designed to capture the "memory effect" of human perception of past quality incidents. (We discuss related work in §8.)

User study methodology: We evaluate these QoE models (KSQI, P.1203, LSTM-QoE) on 16 source videos randomly selected from four public datasets [21, 30, 36, 78] covering

[3] Our methodology extends to other QoE metrics as well.

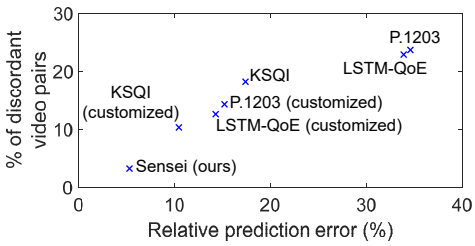


Figure 2: *Existing QoE models exhibit substantial QoE prediction errors (x-axis), which cause them to frequently mispredict the relative QoE ranking between two ABR algorithms on the same video, i.e., a discordant pair (y-axis).*

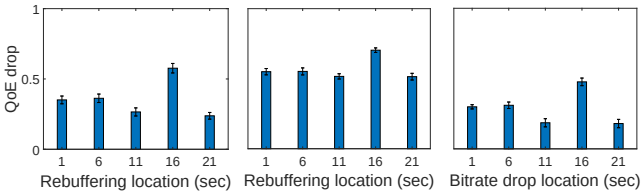
a wide range of content genres (sports, scenic, movies, etc.). These videos are streamed using one of three ABR algorithms: Fugu [83], Pensieve [56], and BBA [45], over 7 throughput traces randomly selected from real-world cellular networks [5, 65], with bandwidths ranging from 200Kbps to 6Mbps. §7.1 and Appendix A provide more details on the videos and network traces. This creates 336 ($16\times7\times3$) rendered videos. To obtain the ground truth QoE of each rendered video, we elicit QoE ratings from crowdsourced workers on Amazon MTurk [1]. We obtain at least 30 ratings from different MTurkers and use the MOS over these ratings as the true QoE of the rendered video. §4 and §5 describe our crowdsourcing methodology in detail.

QoE prediction accuracy: Given the ground-truth QoE, we evaluate the three QoE models both with their pre-trained parameters and after customizing (retraining) them on 315 of the rendered videos selected at random. All models are tested on the remaining 21 videos; we scale their output range and the true QoE to $[0, 1]$. The x-axis of Figure 2 shows the mean relative prediction error of each QoE model on the test set; relative prediction error is defined as $|Q_{predict} - Q_{true}|/Q_{true}$, where $Q_{predict}$ and Q_{true} are the predicted and true QoE of the video. We see that these errors are nontrivial; even the most accurate QoE model has over 10.4% error on average.

We also examine whether these models can correctly *rank* the QoE achieved by two different ABR algorithms. For each pair of source video and throughput trace, we first rank every two of the three ABR algorithms using their true QoE and then again using the predicted QoE. If the rank is different, this pair is called a discordant pair. The y-axis of Figure 2 shows the fraction of discordant pairs among all possible pairs (a common measure used in rank correlation): over 10.2% of pairs are discordant even for the most accurate QoE model. This suggests that using QoE predictions to compare different algorithms (*e.g.,* [45, 56, 83]) may not be reliable.

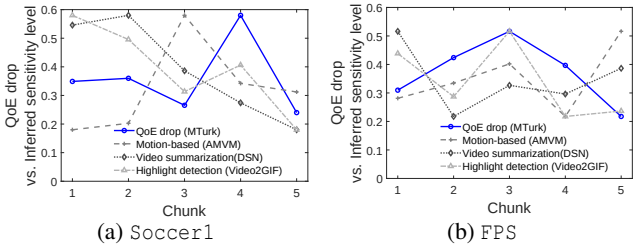
2.2 Temporal variability of quality sensitivity

Figure 2 shows that, unlike prior methods, our QoE model (presented in §4) can predict QoE and rank ABRs significantly more accurately on the same train/test set. We argue that this gap stems from a common assumption shared by all previous QoE models, which is that all factors affecting QoE can be captured by a handful of objective metrics. This



(a) 1-sec rebuffering (b) 4-sec rebuffering (c) Bitrate drop

Figure 3: *Impact of different quality incidents at different points in the video in Figure 1. The pattern of variability remains the same across the different quality incidents. The error bars show standard deviation of means.*



(a) `Soccer1` (b) `FPS`

Figure 4: *QoE rating drop when adding a 1-sec rebuffering at different points in the video, compared to chunk sensitivity levels inferred by saliency models.*

premise ignores the impact of high-level video content (rather than low-level pixels and frames) on users' sensitivity to quality at different parts of the video. We now demonstrate how this quality sensitivity varies as video content changes.

Quantifying dynamic quality sensitivity: Users' sensitivity to quality at a certain part of a video is reflected by the *QoE drop* when a low-quality incident occurs at that part of the video, *i.e.,* $\Delta = Q_{before} - Q_{after}$, where Q_{before} is the MOS of the video without the low-quality incident and Q_{after} is the MOS of the video with the the low-quality incident. To measure the true quality sensitivity at different parts of a source video, we create a *rendered video series* as follows. Rendered videos in a video series have the same source content and highest quality (highest bitrate without rebuffering), except that a low-quality incident (a rebuffering event or a bitrate drop) is deliberately added at different positions, *e.g.,* at the 4th second, 8th second, and so forth. Then, as before, we use Amazon MTurk to crowdsource the true QoE of each rendered video, following our crowdsourcing methodology (§4,§5).

Figure 1 shows an example video series created using a 25-second soccer video as the source video and a one-second rebuffering event as the low-quality incident. We observe significant differences between the QoE drops caused by the rebuffering event at different parts of the video. The highest QoE drop (caused by rebuffering at the 15th second) is 2.1$\times$ higher than the lowest QoE drop (rebuffering at the 10th second). This shows that a low-quality incident can have significantly higher/lower impact on user experience if it occurs a few seconds earlier or later.

Quality sensitivity is inherent to video content: Our user study also suggests that the type of low-quality incident does not affect the ranking of QoE drops within a video series, even though it affects the absolute QoE drops. In other words, quality sensitivity seems to be *inherent* to different parts (contents) of the video. Figure 3 shows the dynamic user sensitivity of three low-quality incidents on the same source video: 1-second rebuffering, 4-second rebuffering, and a bitrate drop from 3Mbps to 0.3Mbps for 4 seconds. Although the absolute values of the QoE drops depend on the particular quality incident, the relative rankings are identical. The strong rank correlation (measured by Spearman's rank coefficient) is persistent across all videos in our dataset: 0.95 rank coefficient between the 1-second and 4-second rebuffering events and 0.94 between the 1-second rebuffering and bitrate drop.

Sources of dynamic quality sensitivity: We speculate that the dynamic quality sensitivity stems from users paying different degrees of attention to different parts of a video. In our dataset, we identify at least three types of moments when users tend to be more (or less) attentive to video quality than usual. The first are key moments in the storyline of a video when tensions have built up; *e.g.,* in `BigBuckBunny` (animation) when the bullies fall into a trap set by the bunny, or in `Soccer1` when a goal is scored. The second are moments when users must pay attention to get important information; *e.g.,* showing the scoreboard in sports videos (`Soccer2`), or acquiring supplies after killing an enemy (`FPS2`). The third are transitional moments with scenic backgrounds, when users tend to be less attentive to quality; *e.g.,* the universe background in `Space`. One can expect many more cases.

2.3 Modeling quality sensitivity?

Can it be captured by QoE models? Traditional QoE models predict the same QoE for all rendered videos in a video series. Even models that do predict different QoE assume that the impact of video content can be captured by pixels and motions; *e.g.,* VMAF [11] (the visual quality metric used by KSQI) gives lower QoE estimates if a bitrate drop occurs when the frame pixels are more "complex". Unfortunately, the impact of content on user sensitivity (as explained above) cannot be fully captured by pixel-level patterns. In Figure 1, the true highest QoE drop occurs when the low-quality incident occurs during the goal, but both VMAF and LSTM-QoE predict it during normal gameplay.

Can it be captured by vision saliency? User sensitivity is conceptually similar to temporal saliency in computer vision. Can saliency/highlight detection models capture user sensitivity of quality? We examine three representative approaches.

- *Traditional motion-based models*, such as AMVM (average motion-vector magnitude) [13, 52], use the motion vector magnitudes of pixels in a chunk to indicate user sensitivity (*i.e.,* users are more sensitive to more dynamic scenes).
- *Interestingness score per frame (highlight detection)*, such as Video2GIF [39] and [34], train a regression model (using C3D [75] neural network as the spatio-temporal feature extractor) on videos with human-annotated per-frame inter-

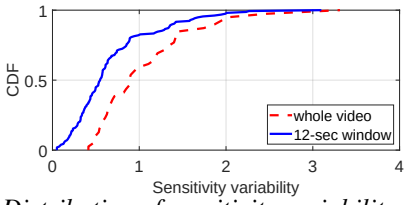


Figure 5: *Distribution of sensitivity variability when a low-quality incident (1-second rebuffering, 4-second rebuffering, or a bitrate drop for 4 seconds) is added at different points in the same video. The trend is similar even if the low-quality incident and QoE gap is localized to a 12-second window.*

estingness scores.[4] The model then produces a per-frame interestingness score which might indicate user sensitivity.

- *Video summarization models*, such as dppLSTM [87] and DSN [90], infer how important each frame is to the whole story of a video, by extracting vision features [74] and using an LSTM to model temporal dependencies. The more important a frame is, the higher user sensitivity might be.

Figure 4 plots the average saliency scores (are normalized $[0,1]$) returned by these models (details in Appendix E) on each chunk of two example videos, and compares them with the QoE drops caused by a 1-sec rebuffering event at different chunks. Unfortunately, the score correlate only weakly with the true user sensitivity: for all videos in our dataset, their absolute (Pearson's) correlations and (Spearman's) rank correlations are less than 0.23 and 0.18, respectively.

The above shows that saliency models may be misaligned with video quality sensitivity. In the soccer video (Figure 1), the part right before the goal is the most quality sensitive, but highlight detection and motion-based models believe the scenes that quickly pan across the audience are most important. Video summarization models pick all diverse moments of a video, including every shot/rewind clip, but users pay more attention to shots that might score a goal. We will provide more discussion in Appendix E. In §7, we show that ABR logic based on saliency scores performs poorly.

2.4 Potential gains

Dynamic quality sensitivity is prevalent: We repeat the same experiment from Figure 1 on all 16 source videos in our dataset and three low-quality incidents: 1-second rebuffering, 4-second rebuffering, and a bitrate drop from 3Mbps to 0.3Mbps for 4 seconds. This creates 48 video series in total. Figure 5 plots the *sensitivity variability* defined by $(\Delta_{max}-\Delta_{min})/\Delta_{min}$ for each video series, where Δ_{max} and Δ_{min} are the maximum and minimum QoE drop of the videos in a series. We see that 21 of the 48 video series have a sensitivity variability of over 0.99, while some have less than 0.20 variability. A similar trend holds even if we localize the low-quality incident and sensitivity gap measurement to 12-second windows. The fact that quality sensitivity varies substantially

[4] We notice that some content providers passively monitor the number of viewers at different parts of a video (*e.g.,* [6]), which is an alternative way of identifying highlights or high-interestingness chunks.

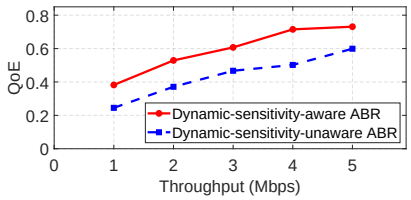


Figure 6: *Being aware of dynamic quality sensitivity can significantly improve QoE and save bandwidth.*

even among very nearby chunks suggests a new opportunity: we can lower the quality when sensitivity is low in order to save bandwidth for nearby chunks whose sensitivity is high.

Potential sensitivity-aware improvement: The above suggests that we can improve ABR algorithms to optimize QoE and save bandwidth by *aligning quality adaptation with dynamic user sensitivity*. We demonstrate the potential gains using an idealistic but clean experiment. We create two simple ABR algorithms whose only difference is the QoE model they optimize: one algorithm optimizes KSQI, the most accurate QoE model from Figure 2 that is *unaware* of dynamic quality sensitivity, and the other optimizes our eventual QoE model from §4, which *is* aware of dynamic quality sensitivity. Both algorithms take as input an entire throughput trace and the same 4-second video chunks encoded using the same bitrate levels. They then decide a bitrate-to-chunk assignment that maximizes their respective QoE model. Note that these ABR algorithms are idealistic because they have access to the entire throughput trace in advance, and hence know the future throughput variability. However, this allows us to eliminate the confounding factor of throughput prediction. We pick one of the throughput traces (results are similar with the other traces) and rescale it to $\{20,40,\ldots,100\}$% to emulate different average network throughput.

For each source video, we create the rendered video as if it were streamed by each ABR algorithm (with bitrate switches, rebufferings, etc.). We use Amazon MTurk as before to assess the true QoE of the rendered video. Figure 6 shows the average QoE of the two ABR algorithms across 16 source videos and different average bandwidths. We see that being aware of dynamic quality sensitivity could improve QoE by 22-52% while using the same bandwidth, or save 39-49% bandwidth while achieving the same QoE.

3 Overview of SENSEI

We have shown that knowing the true quality sensitivity of a video can lead to significant performance improvements when streaming the video. To unleash this potential, we present SENSEI, a video streaming system that unearths and leverages dynamic quality sensitivity. Here, we overview of SENSEI (§3.1) and then introduce our crowdsourcing-based approach to per-video QoE modeling and its limitations (§3.2).

3.1 SENSEI's approach

As shown in Figure 7, SENSEI has two main components.

Per-video QoE modeling: Before streaming a video, SENSEI profiles the quality sensitivity of its chunks. As we saw in

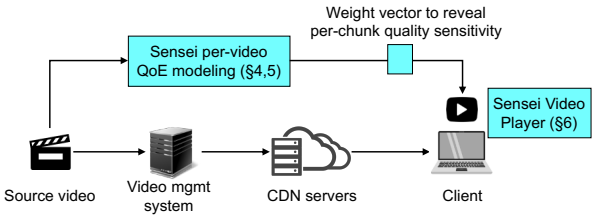


Figure 7: *Overview of* SENSEI.

§2.2, prior QoE models fail to capture content-induced user sensitivity to quality. Instead, we advocate for directly asking human viewers to rate the quality of rendered videos with quality incidents inserted at various chunks. This reveals the true user sensitivity to quality incidents. Since quality sensitivity is unique to each video, this user study must easily scale to many videos. SENSEI uses crowdsourcing to automate and scale the per-video QoE modeling, by addressing two challenges: (1) how many (and which) rendered videos must be rated to build a sensitivity-aware QoE model (§4); and (2) how to get reliable ratings from crowdsourced workers (§5).

Sensitivity-aware ABR: Video players today are designed to maximize bitrate without rebuffering on *every* chunk. This is ill-suited to our goal of aligning quality adaptation with dynamic quality sensitivity: quality should be optimized in proportion to the quality sensitivity of the content. To achieve this, SENSEI refactors the control logic of video players to enable new adaptation actions that "borrow" resources from low-sensitivity chunks and give them to high-sensitivity chunks. We discuss the details in §6.

Instead of building a separate QoE model for each video, SENSEI reuses existing QoE models but reweights the chunks by their quality sensitivity. This is inspired by our observation that relative quality sensitivity is inherent to the video content, rather than the quality incident (§2.2). Thus we can assign a weight to each chunk to encode its inherent quality sensitivity. The abstraction of per-chunk weights has two practical benefits. First, it allows us to reuse existing QoE models by simply reweighting the quality of different chunks; this drastically cuts the cost of QoE crowdsourcing. Second, by using the sensitivity weights as input, the same SENSEI ABR algorithm can be used to optimize QoE for any new video.

3.2 Crowdsourcing quality sensitivity per video

SENSEI directly elicits quality ratings from human viewers to reveal their quality sensitivity to various quality incidents. However, these user ratings must be elicited *per video* and the sheer scale of this feedback can be prohibitive! To put it into perspective, QoE models are usually built from user ratings on just a handful of source videos [21,31], but getting enough user ratings requires a lab environment (or survey platform) to recruit participants and have them watch over two orders of magnitude more video content than the source videos. [5]

[5]For instance, in the WaterlooSQOE-III dataset [31], each video is streamed over 13 throughput traces with 6 ABR algorithms, and each rendered video is then rated by 30 users.

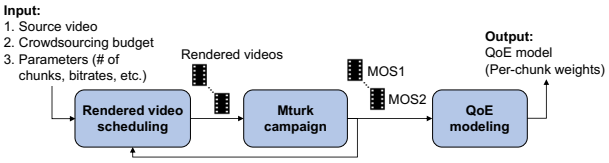


Figure 8: *Workflow of profiling dynamic quality sensitivity using a crowdsourcing platform. The arrow back to the scheduler means that crowdsourced ratings may be used to suggest more rendered videos to iteratively refine the QoE modeling.*

This does not scale if we repeat the process per video.

Consequently, we use crowdsourcing platforms like Amazon MTurk [1] to automate the user studies and scale them out to more videos. Crowdsourcing reduces the overhead of participant recruitment, survey dissemination, and result collection (down to about 78 minutes), and provides a large pool of participants. This allows for repeated experiments to help control for human-related statistical noise. Although the crowdsourcing cost grows with video length, SENSEI offers techniques to reduce the cost (*e.g.,* using fewer crowdsourcing workers). Thus, the content providers can decide whether and how to initiate a profiling given their budgets (See § 4). Note that our reliance on crowdsourcing makes some scenarios, *e.g.,* live contents, currently inapplicable (See § 9).

4 Profiling Quality Sensitivity at Scale

In this section, we show how to build an accurate and cost-efficient QoE model using crowdsourcing. We overview our workflow (§4.1) and then discuss low cost methods for chunk-level reweighting (4.2) and crowdsourcing scheduling (4.3).

4.1 QoE modeling workflow

Figure 8 shows SENSEI's workflow for QoE modeling. SENSEI takes a source video and a monetary budget as input and returns a QoE model that incorporates dynamic quality sensitivity (customized for this video) as output.

- *Rendered video scheduling (§4.3):* We first generate a set of *rendered videos* from the source video. Each rendered video is created by injecting a carefully selected low-quality incident at a certain point in the video.
- *MTurk campaign (§5):* The rendered videos are published on the MTurk platform and we specify how many *participants* (MTurkers) to recruit for this *campaign*. When an MTurker signs up, they start a *survey* that asks them to watch K rendered videos and, after each video, rate its QoE.
- *QoE modeling (§4.2):* Finally, we use the MOS of each rendered video as its QoE and use regression to derive the per-chunk weights, which are then incorporated into an existing QoE model to derive the QoE model for this video.

4.2 Cutting cost via chunk-level reweighting

While crowdsourcing scales QoE profiling elastically, profiling each video can still be prohibitively expensive. Since a QoE model must capture the impact of both quality incidents

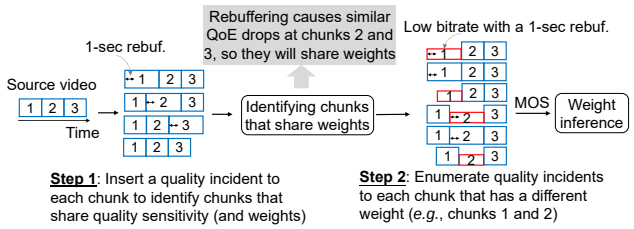


Figure 9: *A running example of the crowdsoucring scheduler for a source video with 3 chunks, 2 bitrate levels (high and low), 2 rebuffering event levels (0 and 1 second).*

and the quality sensitivity of each chunk, a strawman solution would build a QoE model with $O(N \cdot P)$ parameters, where N is the number of chunks and P is the number of parameters in a traditional QoE model. This could require a prohibitive number of ratings to build (*e.g.,* KSQI has tens of parameters).

Encoding quality sensitivity with per-chunk weights: We leverage the insight that quality sensitivity at a chunk is inherent to its video content (§2.2). Thus, SENSEI assigns a single weight to each chunk irrespective of the quality incident, reducing the number of model parameters to $O(N)$. Then, SENSEI *reuses an existing QoE model but reweights the chunks by their quality sensitivity.* If the QoE model is *additive*, *i.e.,* the overall QoE is the average of the QoE estimates of individual chunks q_i, or $Q = \sum_{i=1}^{N} q_i$, then SENSEI can directly reweight the chunks by their quality sensitivity. Though some QoE models are non-additive (*e.g.,* LSTM-QoE), the mainstream QoE models including KSQI and similar ones [56, 85] are. For KSQI, the q_i take into account the impact of visual quality, rebuffering, and quality switches. SENSEI reweights the QoE model as follows:

$$Q = \sum_{i=1}^{N} w_i q_i, \tag{1}$$

where w_i is the weight of the i^{th} chunk, reflecting how much more sensitive users are to quality incidents in this chunk than in other chunks. In theory, q_i, could incorporate impact of the quality of other chunks too.

Weight inference: Given any V rendered videos, if Q_j is the QoE (MOS) of the j^{th} rendered video and $q_{i,j}$ is the estimated QoE of the i^{th} chunk of the j^{th} rendered video, then we can write V equations, $Q_j = \sum_{i=1}^{N} w_i q_{i,j}$ for $j = 1, \ldots, V$. We can then infer the w_i using a linear regression.

In the remainder of the paper, we assume that KSQI reweighted by Equation 1 is the QoE model of SENSEI.

4.3 Crowdsourcing scheduler

We now turn our attention to compiling a small set of rendered videos that, after being rated, will produce enough data to reliably estimate the per-chunk weights.

Two-step scheduling: Given a source video, SENSEI's scheduler uses a two-step process to decide which rendered videos to publish and how many participants to elicit ratings from.

- First, SENSEI creates a set of N rendered videos, each with a single 1-second rebuffering event at a different chunk (N is the number of chunks). It then publishes these videos and asks M_1 participants to rate each video. The total rendered video duration is $O(N \cdot M_1)$. Once the videos are rated, we infer the per-chunk weights as described above.
- Second, we pick $N' \ll N$ chunks whose inferred weights are α-high or low (*e.g.,* 6 % higher or lower than the average weight). We then repeat the first step with two differences: (1) low-quality incidents are added only to these chunks, and (2) the quality incidents include B bitrates (below the highest bitrate) and F rebuffering events (1,2,... seconds). We publish the rendered videos and ask M_2 participants to rate them, for a total video duration of $O(N' \cdot B \cdot F \cdot M_2)$.

The purpose of the first step is to use a small number of participants (M_1) to get a noisy but indicative estimate of which chunks have quality sensitivity that is very high or low, so we can *focus* the second iteration on these chunks using a larger number of participants (M_2). In general, for an ABR algorithm to improve QoE-bandwidth tradeoffs, it is more important to identify which chunks have very high/low quality sensitivity than to precisely estimate the quality sensitivity of every chunk. §5 discusses the number of participants; we evaluate the effect of α, B and F in §7.4. These parameters are empirically selected and held constant throughout our tests.

Figure 9 shows an example two-step schedule for a source video. In the first step, we generate a series of rendered video with the same rebuffering event injected at different chunks. By examining the ratings of these videos from the MTurkers, we determine that chunks 2 and 3 have similar sensitivity to the rebuffering event, allowing them to share the same chunk-level weight. Thus, in the second step, we only need to enumerate the quality incidents for chunks 1 and 2. In practice, for a 20-second video, we generate 5 rendered videos in the first step for the $N = 4$ chunks, of which $N' = 2$ chunks may have high/low sensitivity, and generate 15 rendered videos in the second step for these chunks.

Quality incidents used in profiling: For the set of B bitrates, we use the bitrate levels of YouTube videos and pick three of them to cover high, medium and low visual quality; we found this to be a practical compromise. The set of rebuffering events F are chosen to match those we plan to proactively add to the video (see §6). Testing on a larger set of quality incidents would yield more data points, but our microbenchmarking results in §7.4 show that this only marginally improves model accuracy, while dramatically increasing the cost.

5 Reliable QoE Crowdsourcing

SENSEI's QoE model crucially depends on the *reliability* of MTurkers' quality ratings. This section describes our user survey procedure and techniques for increasing reliability. While SENSEI mostly follows known practices [41,43,57], we provide some key details that arose from multiple iterations.

Single-survey procedure: As shown in Figure 21, each survey starts with the instructions and rejection criteria under which the ratings will be rejected (the criteria are described

below and Appendix B). The MTurker then watches an example video that includes a quality incident, so they know what their ratings should be based on. After that, the MTurker is asked to watch a sequence of rendered videos (determined by the scheduler) and, after each video, rate the quality on a scale of 1-5. Finally, the MTurker does an exit survey.

Quality control per survey: Several measures are taken to prevent and filter out spurious user ratings. First, we show the test videos in *randomized orders* to each MTurk. This eliminates biases due to viewing order and which videos were previously watched. Second, we add *reliability checks*: we show a video without any quality incident at a random position among the test videos, and if an MTurker does not give the highest score to this video, we discard all of their ratings. We also ask the MTurker what quality incident(s) they just saw in the last video, and if they report more quality incidents than were included, that rating is discarded. This may occur if the MTurker's network connection is poor and new quality incidents are introduced. Third, we implement an engagement test to verify if the MTurker watched the video in its entirety, by monitoring the time spent on the video playback page and discarding the rating if the time is shorter than the video length. We also implement other filters, such as limiting the number or length of videos per MTurker to prevent fatigue.

Use of Master MTurkers: We follow a common practice (*e.g.,* [53]) and restrict our tests to *Master MTurkers*, a class of reliable MTurkers who have participated in over 1000 surveys and whose feedback was accepted for over 99% of their prior surveys. We find that our rejection rate from Master MTurkers is over 4× lower than normal MTurkers. One lessons we learned is that Master MTurkers are more willing to participate if the publisher (us) historically has a low rejection rate, because they wish to maintain their rejection rate below 1%.

Sanity check of our dataset: To check if MTurker ratings are similar to prior lab studies [41, 77], we select three 12-second videos from a public dataset [31] whose quality ratings are collected in a lab environment, and obtain MTurker ratings for these videos. We find that the MTurker responses are similar to the in-lab study: after normalizing the ratings to the same range, the MTurkers' MOS differs by less than 3% from the in-lab study's MOS on the same video.

How many MTurkers are needed? We did a head-to-head comparison with WaterlooSQOE-III [31] and found that we need 17% more MTurkers to reduce the variance of QoE ratings down to the levels of the in-lab study. §7.4 shows how the number of MTurkers affects SENSEI's performance.

Despite the above, we acknowledge that our MTurk survey methodology could always be susceptible to human factors, and we continue to improve it.

6 SENSEI's ABR Logic

The key difference between SENSEI's ABR logic and traditional ABR logic is that it aligns quality adaptation with the temporal variability of quality sensitivity. We first show how SENSEI modifies a traditional ABR framework (6.1), and then show how existing ABR algorithms can be minimally modified to benefit from SENSEI (§6.2).

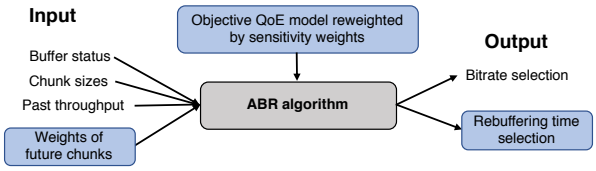


Figure 10: *ABR framework of* SENSEI. *The differences with traditional ABR framework are highlighted.*

6.1 Enabling new adaptation actions

SENSEI takes a pragmatic approach by working within the framework of existing players. It proposes specific changes to their input and output, as highlighted in Figure 10.

Input: Besides the current buffer length, next chunk sizes, and history of throughput measurements, SENSEI's ABR algorithm takes as input the sensitivity weights of the next h chunks. A larger h allows us to look farther into the horizon for opportunities to trade current quality for future quality, or vice versa. In practice, we are also constrained by the reliability of our bandwidth prediction for future chunks. We microbenchmark the selection of h in §7.5.

Output: SENSEI's ABR algorithm selects the bitrate for future chunks as well as when the next rebuffering event should occur.[6] In contrast, traditional players only initiate rebuffering events when the buffer is empty.

Objective QoE model: If the ABR algorithm explicitly optimizes an additive QoE model, SENSEI can modify its objective as described in §4.2. While SENSEI can be applied to most ABR algorithms (*e.g.,* [56, 83, 85]), some (*e.g.,* BBA) do not have explicit objective to which SENSEI can be applied.

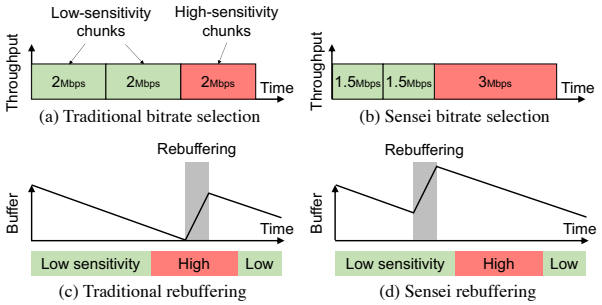
In theory, these changes are sufficient to enable at least the following optimizations, which traditional ABR algorithms are unlikely to explicitly do. (1) Lowering the current bitrate so that it can raise the bitrate for the next few chunks, if they have higher quality sensitivity (Figure 11(a) and (b)); (2) Raising the current bitrate slightly over the sustainable level if quality sensitivity is expected to decrease in the next few chunks; and (3) Initiating a short rebuffering event now in order to ensure smoother playback for the next few chunks, if they have higher quality sensitivity (Figure 11(c) and (d)).

6.2 Refactoring current ABR algorithms

We apply SENSEI to two ABR algorithms: Pensieve [56], based on deep reinforcement learning, and Fugu [83], a more traditional algorithm based on bandwidth prediction.

Applying SENSEI to Pensieve: SENSEI leverages the flexibility of deep neural networks (DNNs) and augments Pensieve's input, output and QoE objective—its states, actions,

[6] SENSEI currently makes adaptation decisions only for the next chunk, but in principle it could plan adaptations for multiple chunks in the future.



(a) Traditional bitrate selection (b) Sensei bitrate selection

(c) Traditional rebuffering (d) Sensei rebuffering

Figure 11: *Illustrative examples of* SENSEI *vs traditional ABR logic: how* SENSEI *improves quality (a vs. b) or avoids bad quality (c vs. d) for high-sensitivity chunks.*

and reward, in the terminology of reinforcement learning—as described in §6.1. It then retrains the DNN model in the same way as Pensieve; we call this variation SENSEI-Pensieve. SENSEI-Pensive makes two minor changes to reduce the action space (which now includes rebuffering). First, we restrict possible rebuffering times to three levels ($\{0,1,2\}$ seconds) that can only happen at chunk boundaries. Second, instead of choosing among combinations of bitrates and rebuffering, SENSEI-Pensieve either selects a bitrate or initiates a rebuffering event at the next chunk. If it chooses the latter, SENSEI-Pensieve will increment the buffer state by the chosen rebuffering time and rerun the ABR algorithm immediately.

Applying SENSEI to Fugu: Let us first explain how Fugu works. At a high level, before downloading the i^{th} chunk, Fugu considers the throughput prediction for the next h chunks. For any throughput variation γ (with predicted probability $p(\gamma)$) and bitrate selection $B = (b_i, \ldots, b_{i+h-1})$, where b_j is the bitrate of the j^{th} chunk, it simulates when each of the next h chunks will be downloaded and estimates the rebuffering time $t_j(B,\gamma)$ of the j^{th} chunk (which could be zero). It then picks the bitrate vector $(b_i, \ldots, b_{i+h-1})$ that maximizes the expected total quality over the next h chunks and possible throughput variations: $\sum_\gamma p(\gamma) \sum_{j=i}^{i+h-1} q(b_j, t_j(B,\gamma))$. Here, $q(b,t)$ estimates the quality of a chunk with the bitrate b and rebuffering time t using a simplified model of KSQI.

Now, the SENSEI variation of Fugu, which we call SENSEI-Fugu, uses Fugu's throughput prediction and the sensitivity weights w_j of the next h chunks. SENSEI-Fugu picks the bitrate vector $B = (b_i, \ldots, b_{i+h-1})$ *and* the rebuffering time vector $T = (t_i, \ldots, t_{i+h-1})$, where t_j is the rebuffering time of the j^{th} chunk, that maximizes the expected total quality over the next h chunks and possible throughput variations:

$$\sum_\gamma p(\gamma) \sum_{j=i}^{i+h-1} w_j q(b_j, t_j) \tag{2}$$

Here, the chosen rebuffering times must be feasible, *i.e.,* the buffer length can never be negative.

In short, SENSEI-Pensieve and SENSEI-Fugu add an extra action (rebuffering time per chunk), and their objective function reweights the contribution of each chunk's quality using the sensitivity weights provided by our QoE model.

6.3 Player implementation and integration

We implement SENSEI on DASH.js [3], an open source player that several commercial players are based on. We add a new field in the DASH manifest file (under `Representation`) to represent per-chunk sensitivity weights and change the parser `ManifestLoader` to parse these weights. Unlike other ABR players, SENSEI may initiate rebuffering when the buffer is not empty. We use Media Source Extensions [7] (an API that allows browsers to change player states) to delay a downloaded chunk in the browser buffer from being loaded into the player buffer. We also describe the implementation of our crowdsourcing pipeline for MTurk surveys in Appendix D.

7 Evaluation

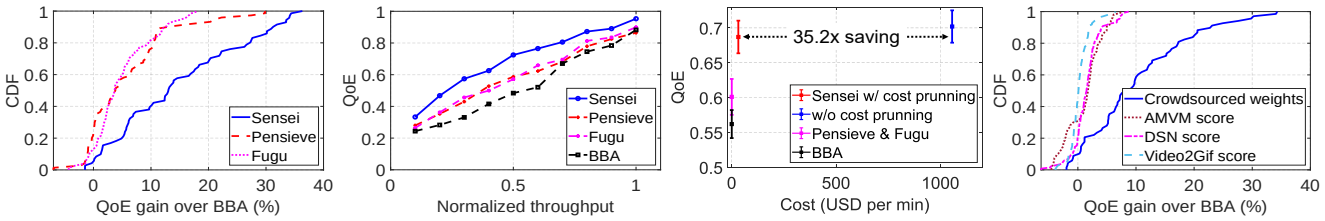
Our evaluation of SENSEI shows several key findings:

- Compared to recent proposals, SENSEI can improve QoE by 7.7-52.5% without using more bandwidth or can save 12.1-50.3% bandwidth while achieving the same QoE.
- The performance gains of SENSEI come at a cost of $31.4 for a 1-minute video, which is marginal compared to the investments made by content providers.
- SENSEI can improve QoE prediction accuracy by 11.8-37.1% over state-of-the-art QoE models.
- SENSEI's ABR algorithm consistently outperforms baseline ABR algorithms even when bandwidth fluctuates.

7.1 Experimental setup

Test videos and throughput traces: Our test videos are selected from four datasets: LIVE-MOBILE [36], LIVE-NFLX-II [21], and WaterlooSQOE-III [31] are professional-grade datasets, often used to train/compare QoE models in the literature. We complement these sources with videos from a user-generated dataset, YouTube-UGC [78]. The videos are randomly selected from four video genres: sports, gaming, nature, and animation. Appendix §A provides more details of the videos. To create an adaptive video streaming setup, we chop videos into 4-second chunks and encode each chunk at 5 bitrate levels: $\{300, 750, 1200, 1850, 2850\}$Kbps. We randomly select 10 throughput traces from two public datasets, FCC [24] and 3G/HSDPA [65], restricting our selection to those whose average throughput is between 0.2Mbps and 6Mbps, forcing the ABR algorithms to adapt their bitrates.

Baselines: We compare SENSEI's ABR algorithm with three baselines: Buffer-based adaptation (BBA) [45], Fugu [83], and Pensieve [56]. We keep their default settings (*e.g.,* same DNN architecture and training network traces for Pensieve, etc.). For fairness, we use KSQI as the QoE model for Pensieve, Fugu, and the SENSEI variants. This modification should improve the quality of Pensieve and Fugu, because the QoE models used in their original implementations are special cases of KSQI. We use SENSEI-Pensieve (*i.e.,* the application of SENSEI to Penseive) as SENSEI, but confirm that the improvements of SENSEI-Fugu are similar (Figure 18a).



(a) QoE gains over BBA (b) QoE vs. bandwidth usage (c) MTurk cost vs. QoE (d) SENSEI with crowdsourced weight vs. saliency-based weights

Figure 12: *End-to-end performance of* SENSEI *over traditional and saliency-based ABR baselines, across all videos.*

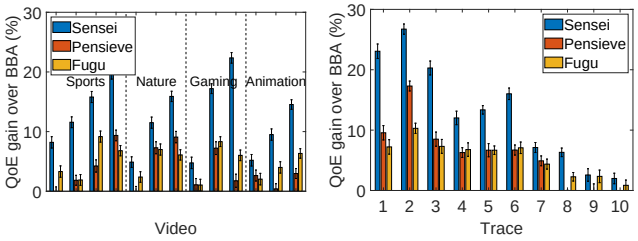
Performance metrics: We use three performance metrics. For a given source and video and throughput trace, we report the **QoE gain** of one ABR algorithm (Q_1) over another (Q_2), *i.e.,* $(Q_1 - Q_2)/Q_2$, where Q_1 and Q_2 are *rated by MTurkers*. We calculate SENSEI's **bandwidth saving** by scaling down the throughput traces and determining how much bandwidth each ABR algorithm needs to achieve the same QoE. We normalize all QoE values to the range $[0, 1]$. We measure the **crowdsourcing cost** paid to MTurk to get enough ratings to profile a 1-minute video. Only SENSEI incurs this cost.

7.2 End-to-end improvement

QoE gains: Figure 12a shows the distributions of QoE gains of SENSEI, Pensieve, and Fugu over BBA, across all combinations of the 16 source videos and 10 network traces. Compared to BBA, SENSEI has at least 14.4% QoE gain for half of the trace-video combinations, whereas Pensieve's and Fugu's median QoE gains are about 5.7%. The tail improvement of SENSEI is greater: SENSEI's QoE gain at the 80th percentile is 4.8%, whereas Pensieve's and Fugu's are 0.2% and 0.7% respectively. The fact that SENSEI's gains over Pensieve (its base ABR logic) are similar to Pensieve's gains over BBA suggests the significant potential in making an existing ABR algorithm aware of dynamic quality sensitivity.

Bandwidth savings: Figure 12b shows the average QoE of different ABR algorithms across the source videos, under one throughput trace scaled down by different ratios (x-axis). We confirm the results are consistent across different throughput traces. We see that when setting a target QoE of 0.8, the bandwidth savings of SENSEI is about 27% higher compared to Pensieve and Fugu, and 32% higher compared to BBA.

QoE vs. crowdsourcing cost: Figure 12c shows the crowdsourcing cost and resulting QoE of SENSEI relative to Pensieve, both with and without the cost-pruning optimization (which is evaluated separately in Figure 16). Compared to enumerating all combinations of the quality incidents, we see that costs can be reduced by more than 32× with only a 3.1% degradation in QoE, and SENSEI is still 14.7% better on average than its base ABR logic (Pensieve with KSQI). This cost is equivalent to ∼$31.4 per 1-minute video, which is a negligible cost for large content providers that usually spend on the order of $10 billion annually [4] for licensing popular TV shows (or making such shows).



(a) QoE gain grouped by genre (b) QoE gain grouped by trace

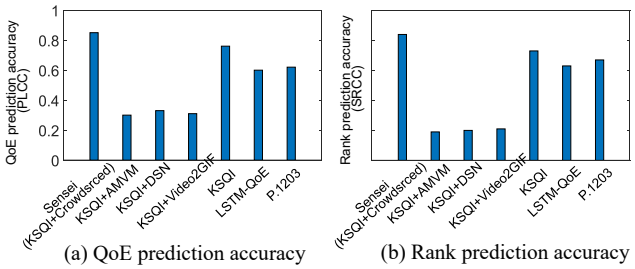
Figure 13: *QoE gains over BBA for genre and for each throughput trace (ordered by increasing average throughput)*

Improvements by video and trace: Figure 13a shows the QoE gains for each video across the network traces. We see a significant variability in the QoE gains across videos and even within the same genre. Figure 13b shows the QoE gains for each network trace across all videos. Overall, SENSEI yields more improvement when the average throughput is lower (towards the left). This shows that SENSEI can better maintain high QoE even when the network is under stress.

SENSEI vs. saliency-reweighted ABR: Finally, Figure 12d shows the QoE gains of SENSEI when the per-chunk weights are based on crowdsourcing results (our approach) and when the weights are based on the saliency scores produced by various saliency models (see §2.3). We normalize each model's saliency scores to sum to the sum of chunk weights of SENSEI. We see that SENSEI's gain significantly reduces if the weights are based on these saliency models, because as explained in §2.3, they fail to capture users' quality sensitivity.

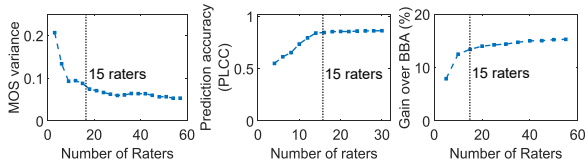
7.3 QoE prediction accuracy

We now microbenchmark SENSEI's QoE model introduced in §4 using all (640) rendered videos generated by running SENSEI and the baseline ABR algorithms on all combinations of source videos and network traces. We obtain the "ground truth" QoE of each rendered video using our MTurk survey procedure (§4,§5). We calculate Pearson's linear correlation coefficient (**PLCC**) and Spearman's rank correlation coefficient (**PRCC**) between the predicted QoE and actual user-rated QoE. Figure 14 compares SENSEI with three baselines QoE models (KSQI, LSTM-QoE, P.1203). PLCC (and PRCC) of SENSEI's QoE prediction is over 0.85 (and 0.84), whereas the baselines are below 0.76 (and 0.73). We evaluated several variants of KSQI (the best baseline QoE model) re-weighted by per-chunk saliency scores from the saliency



(a) QoE prediction accuracy (b) Rank prediction accuracy

Figure 14: *QoE prediction accuracy of* SENSEI*,* SENSEI*'s variants and baseline QoE models.*



(a) Rating variance vs. # of raters (b) Model accuracy vs. # of raters (c) QoE gain vs. # of raters

Figure 15: *The effect of the number of raters*

models in §2.3, but their accuracies are even lower.

7.4 Cost savings on crowdsourcing

We microbenchmark the effects of different crowdsourcing parameters on SENSEI's QoE model.

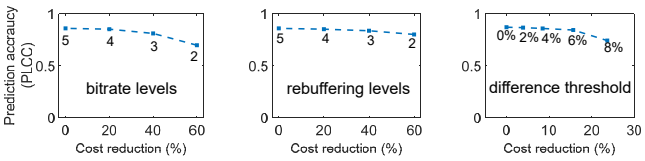
Impact of number of raters per video: Figure 15(a) shows that while the quality ratings have substantial variance with less than 5 raters, their mean value (MOS) stabilizes with more than 15 raters. As a result, having 15 raters per video (as used in SENSEI) produces similar QoE prediction accuracy (b) and QoE gains (c) as having 30 raters.

Impact of crowdsourcing schedule granularity: Figure 16 shows the effect of reducing MTurk cost by considering (a) less bitrate levels (B), (b) less rebuffering events (F), or (c) higher threshold α used to pick which chunks to investigate in the second step. These terms are defined in §4.3. By reducing B to 3, F to 2, or raising α to 6%, we greatly reduce the cost while incurring less than 3% drop in accuracy.

7.5 SENSEI's ABR logic

Finally, we microbenchmark SENSEI's ABR logic (§6). To scale this experiment out, we use real videos and throughput traces but use the QoE predicted by SENSEI (instead of real user ratings) to evaluate QoE. We have confirmed that this yields the same QoE estimation on average as real user ratings under the same setting.

Impact of bandwidth variance: Figure 17 shows the performance of SENSEI under increasing throughput variance. We pick one throughput trace and increase its throughput variance by adding unbiased Gaussian noise. The graph begins at the variance of the original throughput trace; as variance increases, SENSEI's QoE degrades gracefully, but it still maintains a significant gain over its base ABR logic (Pensieve or Fugu). This is because SENSEI only needs to predict how likely low throughput will occur on high quality-sensitivity



(a) # of bitrate level (b) # of rebuffering level (c) Filtering threshold

Figure 16: *QoE model accuracy changes with cost.*

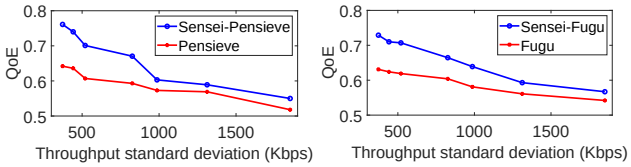


Figure 17: *QoE under increasing bandwidth variance.*

chunks, not all future chunks, so if the average throughput until the next such chunk is predictable, it will work well. We confirm the results are similar on other throughput traces.

Performance breakdown: Figure 18a shows that SENSEI achieves comparable improvement when either Pensieve or Fugu are the base ABR logic. This suggests that SENSEI's gains do not depend on the choice of the base ABR logic. Figure 18b shows that both aspects of SENSEI's control logic contributes to its improvements: (1) making ABR logic aware of dynamic quality sensitivity (1st vs. 2nd bar), and (2) injecting rebuffering judiciously (2nd vs. 3rd bar). Thus, even if a content provider cannot control rebuffering, it can still benefit signficantly from SENSEI's dynamic quality sensitivity.

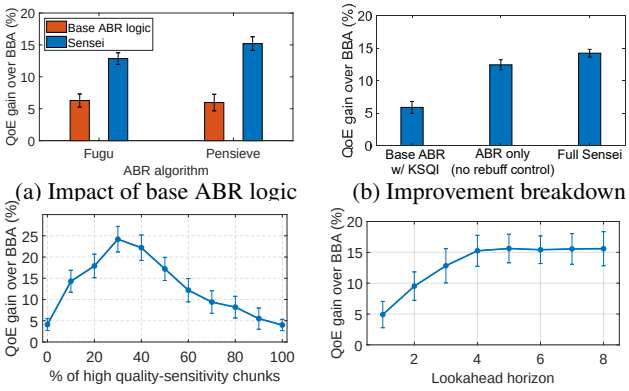
Impact of video contents: While the videos in our dataset have varying fractions (from 20% to 60%) of high-sensitivity chunks, Figure 18c tests Sensei's performance under an even wider range of high-sensitivity chunk fractions (from 0% to 100%). We create source videos with the specific fractions of high and low quality-sensitivity chunks and randomize the positions of the chunks. SENSEI has marginal improvement when the video is dominated by either high quality-sensitivity or low quality-sensitivity chunks. However, SENSEI significantly improves QoE when high quality-sensitivity chunks are 20-40% of a video (most of our videos fall in this range).

Lookahead horizon: Figure 18d tests the impact of *lookahead horizon*—the number of future chunks whose quality-sensitivity weights will be revealed to the ABR. A longer horizon increases SENSEI's ability to schedule quality events between low and high quality-sensitivity chunks. Empirically in our dataset, the QoE gains diminish after the lookahead horizon is greater than 4 chunks.

Systems overhead: We confirm empirically that compared to a video player without SENSEI, the runtime overhead of SENSEI is less than 1% in both CPU cycles and RAM usage.

8 Related Work

ABR algorithms: Mainstream ABR algorithms maximize bitrate under dynamic available bandwidth. Traditional ones are buffer-based (*e.g.,* [46, 50]) or rate-based algorithms



(a) Impact of base ABR logic (b) Improvement breakdown

(c) Chunk sensitivity (d) Looking-ahead horizon

Figure 18: *Understanding* SENSEI*'s improvements.*

(*e.g.,* [45,69,70]). Recent ABR algorithms explicitly optimize a given QoE objectives, via control theory [85], ML-based throughput prediction [73, 83], or deep reinforcement learning [34,35,56]. Some ABR algorithms also rely on server-side assist [17, 47, 84]. Key parameters of the ABR logic can be customized to the network conditions or devices [16]. Though SENSEI reuses existing ABR algorithms, its contribution lies in identifying minimum changes (*e.g.,* adaptation actions they never would have taken) needed for these algorithms to fully leverage users' dynamic quality sensitivity.

Modeling and optimizing user-perceived quality: Visual quality assessment (VQA) traditionally models user's perception of encoded video using pixel-level patterns (*e.g.,* [38, 64, 68, 79]) as well as advanced data-driven models, such as SVM [11] and deep learning models (*e.g.,* [48]). Adaptive quality assessment (AQA), on the other hand, models streaming-related incidents, including join time, bitrate switches, rebuffering (*e.g.,* [18, 28, 49]). Recent QoE models combine VQA and AQA (*e.g.,* [19, 20, 22, 25, 29, 31, 33]) and sometimes uses spatial/temporal visual attention (*e.g.,* [30, 34, 34, 37, 59, 60, 82]). These perception-centric QoE models have inspired a large body of work that maximizes user-perceived quality with bitrate adaptation [58, 61], adaptive video encoding [17,67,89], adaptive bitrate levels [14,15], dynamic chunk lengths [51], and super resolution [47, 84, 88]. Since the user-perceived quality metrics can vary across chunks, they may also treat video chunks differently, like SENSEI does. However, as elaborated in §2.3, SENSEI is complementary to these efforts: while they propose heuristics to how pixel-/motion-based visual features affect QoE, SENSEI customizes itself for each video (in a cost-efficient way) to capture the impact of the *substance of video content* on true user sensitivity to video quality. That said, actions like dynamic bitrate levels, chunk lengths, and super resolution could be used in SENSEI too, though SENSEI only considers actions directly supported by current DASH players.

QoE research using crowdsourcing: Prior work (*e.g.,* [23, 42, 43, 57, 62, 81]) provides methodologies for using commercial crowdsourcing platforms, *e.g.,* Amazon MTurk [1] and Microworkers [8], to systematically model user perception using objective quality metrics [23, 42, 43, 62, 81], investigate QoE impact of different types of low quality events (*e.g.,* [32]), and build crowdsourcing platforms themselves for similar purposes (*e.g.,* [77, 83]). While SENSEI follows conventional crowdsourcing methodology (§5), SENSEI faces a unique challenge of scaling crowdsourcing to *per-video* QoE modeling. The cost of modeling QoE of each video separately is prohibitive, and SENSEI drastically prunes the cost by reusing an existing QoE model while profiling only a single weight per chunk to encode the content-induced quality sensitivity of each chunk.

9 Discussion

Participant selection bias: A concern of any crowdsourced user study is that the results could be biased because the workers who are more willing to participate in the user study might have different characteristics as the real video viewers. A common idea to address this bias is to reweight the participant responses based on the demographics of real users (*e.g.,* [26, 55, 71, 76]). SENSEI could either apply reweighting to the user study if we have knowledge of the demographics of the target viewers or directly recruit the user study participants from the potential viewers (*e.g.,* subscribers of content providers).

Inapplicable scenarios: SENSEI does not apply to live videos and copyrighted videos. Live videos have strict delay requirements which SENSEI's crowdsourcing-based video profiling cannot meet. Showing copyrighted videos to crowdsource workers poses a risk of copyright violation, though SENSEI could be used on released videos. Moreover, the profiling cost of $\sim$ \$31.4 per minute of video may still be overwhelming for videos with few views. Instead, we envision that SENSEI works well for popular on-demand video providers who struggle to achieve the highest quality: *e.g.,* mainstream video providers such as Amazon, Netflix and YouTube lowered the default bitrate in Europe due to increased network traffic during the covid lockdown [9].

10 Conclusion

We have descried SENSEI, a video streaming system that optimizes video quality by exploiting dynamic quality sensitivity. Observing that quality sensitivity is inherent to video content and hence unique to each video, SENSEI scales out the profiling of quality sensitivity using a reliable crowdsourcing methodology. We show that with minor modifications to state-of-the-art ABR algorithms, SENSEI can improve their QoE by 15.1% or save bandwidth by 26.8% on average.

Acknowledgement

We thank the anonymous reviewers and our shepherd Dongsu Han. Xu Zhang and Junchen Jiang are supported by NSF (CNS-1901466), CERES Center, and a Google Faculty Research Award.

References

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

[2] Cisco Annual Internet Report (2018–2023) White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html/.

[3] DASH.js. https://github.com/Dash-Industry-Forum/dash.js/wiki.

[4] How Netflix Pays for Movie and TV Show Licensing. https://www.investopedia.com/articles/investing/062515/how-netflix-pays-movie-and-tv-show-licensing.asp.

[5] HSDPA dataset. http://skuld.cs.umass.edu/traces/mmsys/2013/pathbandwidth/.

[6] Live stream on YouTube, See your live stream's metrics. https://support.google.com/youtube/answer/2853833.

[7] Media Source Extensions. https://www.w3.org/TR/media-source/.

[8] Microworkers. https://www.microworkers.com/.

[9] Reuters: YouTube, Amazon Prime forgo streaming quality to relieve European networks. https://uk.reuters.com/article/us-health-coronavirus-youtube-exclusive/exclusive-youtube-to-reduce-streaming-quality-in-europe-due-to-coronavirus-idUKKBN2170OP.

[10] Video Quality of Experience: Requirements and Considerations for Meaningful Insight. https://www.sandvine.com/hubfs/downloads/archive/whitepaper-video-quality-of-experience.pdf.

[11] VMAF - Video Multi-Method Assessment Fusion. https://github.com/Netflix/vmaf.

[12] YouTube joins Netflix in reducing video quality in Europe. https://www.theverge.com/2020/3/20/21187930/youtube-reduces-streaming-quality-european-union-coronavirus-bandwidth-internet-traffic.

[13] Golnaz Abdollahian, Cuneyt M Taskiran, Zygmunt Pizlo, and Edward J Delp. Camera motion-based analysis of user generated video. *IEEE Transactions on Multimedia*, 12(1):28–41, 2009.

[14] Hasti Ahlehagh and Sujit Dey. Adaptive bit rate capable video caching and scheduling. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1357–1362. IEEE, 2013.

[15] Hasti Ahlehagh and Sujit Dey. Video-aware scheduling and caching in the radio access network. *IEEE/ACM Transactions on Networking*, 22(5):1444–1462, 2014.

[16] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. Oboe: auto-tuning video abr algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 44–58, 2018.

[17] Ramon Aparicio-Pardo, Karine Pires, Alberto Blanc, and Gwendal Simon. Transcoding live adaptive video streams at a massive scale in the cloud. In *Proceedings of the 6th ACM Multimedia Systems Conference*, pages 49–60, 2015.

[18] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. *ACM SIGCOMM Computer Communication Review*, 43(4):339–350, 2013.

[19] Christos G Bampis and Alan C Bovik. An augmented autoregressive approach to http video stream quality prediction. *arXiv preprint arXiv:1707.02709*, 2017.

[20] Christos G Bampis, Zhi Li, and Alan C Bovik. Continuous prediction of streaming video qoe using dynamic networks. *IEEE Signal Processing Letters*, 24(7):1083–1087, 2017.

[21] Christos G Bampis, Zhi Li, Ioannis Katsavounidis, Te-Yuan Huang, Chaitanya Ekanadham, and Alan C Bovik. Towards perceptually optimized end-to-end adaptive video streaming. *arXiv preprint arXiv:1808.03898*, 2018.

[22] Chao Chen, Lark Kwon Choi, Gustavo De Veciana, Constantine Caramanis, Robert W Heath, and Alan C Bovik. Modeling the time—varying subjective quality of http video streams with rate adaptations. *IEEE Transactions on Image Processing*, 23(5):2206–2221, 2014.

[23] Kuan-Ta Chen, Chi-Jui Chang, Chen-Chi Wu, Yu-Chun Chang, and Chin-Laung Lei. Quadrant of euphoria: a crowdsourcing platform for qoe assessment. *IEEE Network*, 24(2):28–35, 2010.

[24] Federal Communications Commission et al. Raw data-measuring broadband america. https://www.fcc.gov/reports-research/reports/. *Retrieved June*, 19:2018, 2016.

[25] Johan De Vriendt, Danny De Vleeschauwer, and David Robinson. Model for estimating qoe of video delivered using http adaptive streaming. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1288–1293. IEEE, 2013.

[26] Djellel Difallah, Elena Filatova, and Panos Ipeirotis. Demographics and dynamics of mechanical turk workers. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 135–143, 2018.

[27] Li Ding, Hua Huang, and Yu Zang. Image quality assessment using directional anisotropy structure measurement. *IEEE Transactions on Image Processing*, 26(4):1799–1809, 2017.

[28] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. *ACM SIGCOMM Computer Communication Review*, 41(4):362–373, 2011.

[29] Zhengfang Duanmu, Wentao Liu, Diqi Chen, Zhuoran Li, Zhou Wang, Yizhou Wang, and Wen Gao. A knowledge-driven quality-of-experience model for adaptive streaming videos. *arXiv preprint arXiv:1911.07944*, 2019.

[30] Zhengfang Duanmu, Abdul Rehman, and Zhou Wang. A quality-of-experience database for adaptive video streaming. *IEEE Transactions on Broadcasting*, 64(2):474–487, 2018.

[31] Zhengfang Duanmu, Kai Zeng, Kede Ma, Abdul Rehman, and Zhou Wang. A quality-of-experience index for streaming video. *IEEE Journal of Selected Topics in Signal Processing*, 11(1):154–166, 2016.

[32] Sebastian Egger, Bruno Gardlo, Michael Seufert, and Raimund Schatz. The impact of adaptation strategies on perceived quality of http adaptive streaming. In *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, pages 31–36, 2014.

[33] Nagabhushan Eswara, S Ashique, Anand Panchbhai, Soumen Chakraborty, Hemanth P Sethuram, Kiran Kuchi, Abhinav Kumar, and Sumohana S Channappayya. Streaming video qoe modeling and prediction: A long short-term memory approach. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[34] Guanyu Gao, Linsen Dong, Huaizheng Zhang, Yonggang Wen, and Wenjun Zeng. Content-aware personalised rate adaptation for adaptive streaming via deep video analysis. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–8. IEEE, 2019.

[35] Guanyu Gao, Huaizheng Zhang, Han Hu, Yonggang Wen, Jianfei Cai, Chong Luo, and Wenjun Zeng. Optimizing quality of experience for adaptive bitrate streaming via viewer interest inference. *IEEE Transactions on Multimedia*, 20(12):3399–3413, 2018.

[36] Deepti Ghadiyaram, Janice Pan, and Alan C Bovik. A subjective and objective study of stalling events in mobile streaming videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):183–197, 2017.

[37] Deepti Ghadiyaram, Janice Pan, and Alan C Bovik. Learning a continuous-time streaming video qoe model. *IEEE Transactions on Image Processing*, 27(5):2257–2271, 2018.

[38] Rafael C Gonzalez, Richard Eugene Woods, and Steven L Eddins. *Digital image processing using MATLAB*. Pearson Education India, 2004.

[39] Michael Gygli, Yale Song, and Liangliang Cao. Video2gif: Automatic generation of animated gifs from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1001–1009, 2016.

[40] Kotaro Hara, Abigail Adams, Kristy Milland, Saiph Savage, Chris Callison-Burch, and Jeffrey P Bigham. A data-driven analysis of workers' earnings on amazon mechanical turk. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2018.

[41] Tobias Hoßfeld, Matthias Hirth, Judith Redi, Filippo Mazza, Pavel Korshunov, Babak Naderi, Michael Seufert, Bruno Gardlo, Sebastian Egger, and Christian Keimel. Best practices and recommendations for crowdsourced qoe-lessons learned from the qualinet task force" crowdsourcing". 2014.

[42] Tobias Hossfeld, Christian Keimel, Matthias Hirth, Bruno Gardlo, Julian Habigt, Klaus Diepold, and Phuoc Tran-Gia. Best practices for qoe crowdtesting: Qoe assessment with crowdsourcing. *IEEE Transactions on Multimedia*, 16(2):541–558, 2013.

[43] Tobias Hoßfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. Quantification of youtube qoe via crowdsourcing. In *2011 IEEE International Symposium on Multimedia*, pages 494–499. IEEE, 2011.

[44] Sudeng Hu, Lina Jin, Hanli Wang, Yun Zhang, Sam Kwong, and C-C Jay Kuo. Compressed image quality metric based on perceptually weighted distortion. *IEEE Transactions on Image Processing*, 24(12):5594–5608, 2015.

[45] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 187–198, 2014.

[46] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 97–108, 2012.

[47] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 107–125, 2020.

[48] Woojae Kim, Jongyoo Kim, Sewoong Ahn, Jinwoo Kim, and Sanghoon Lee. Deep video quality assessor: From spatio-temporal visual sensitivity to a convolutional neural aggregation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 219–234, 2018.

[49] S Shunmuga Krishnan and Ramesh K Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *IEEE/ACM Transactions on Networking*, 21(6):2001–2014, 2013.

[50] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran. Probe and adapt: Rate adaptation for http video streaming at scale. *IEEE Journal on Selected Areas in Communications*, 32(4):719–733, 2014.

[51] Jan Lievens, Shahid M Satti, Nikos Deligiannis, Peter Schelkens, and Adrian Munteanu. Optimized segmentation of h. 264/avc video for http adaptive streaming. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 1312–1317. IEEE, 2013.

[52] Yao Liu, Sujit Dey, Fatih Ulupinar, Michael Luby, and Yinian Mao. Deriving and validating user experience model for dash video streaming. *IEEE Transactions on Broadcasting*, 61(4):651–665, 2015.

[53] Matt Lovett, Saleh Bajaba, Myra Lovett, and Marcia J Simmering. Data quality from crowdsourced surveys: A mixed method inquiry into perceptions of amazon's mechanical turk masters. *Applied Psychology*, 67(2):339–366, 2018.

[54] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

[55] Jianguo Lu and Dingding Li. Estimating deep web data source size by capture–recapture method. *Information retrieval*, 13(1):70–95, 2010.

[56] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 197–210, 2017.

[57] M Sajid Mushtaq, Brice Augustin, and Abdelhamid Mellouk. Crowd-sourcing framework to assess qoe. In *2014 IEEE International Conference on Communications (ICC)*, pages 1705–1710. IEEE, 2014.

[58] Vikram Nathan, Vibhaalakshmi Sivaraman, Ravichandra Addanki, Mehrdad Khani, Prateesh Goyal, and Mohammad Alizadeh. End-to-end transport for video qoe fairness. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 408–423. 2019.

[59] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1190–1198, 2018.

[60] Cagri Ozcinar, Julian Cabrera, and Aljosa Smolic. Visual attention-aware omnidirectional video streaming using optimal tiles for virtual reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):217–230, 2019.

[61] Benjamin Rainer, Stefan Petscharnig, Christian Timmerer, and Hermann Hellwagner. Statistically indifferent quality variation: An approach for reducing multimedia distribution cost for adaptive video streaming services. *IEEE Transactions on Multimedia*, 19(4):849–860, 2016.

[62] Benjamin Rainer, Markus Waltl, and Christian Timmerer. A web based subjective evaluation platform. In *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 24–25. IEEE, 2013.

[63] ITUT Rec. H. 264: Advanced video coding for generic audiovisual services. *ITU-T Rec. H. 264-ISO/IEC 14496-10 AVC*, 2005.

[64] Abdul Rehman, Kai Zeng, and Zhou Wang. Display device-adapted video quality-of-experience assessment. In *Human Vision and Electronic Imaging XX*, volume 9394, page 939406. International Society for Optics and Photonics, 2015.

[65] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. Commute path bandwidth traces from 3g networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 114–118, 2013.

[66] Werner Robitza, Marie-Neige Garcia, and Alexander Raake. A modular http adaptive streaming qoe model—candidate for itu-t p. 1203 (“p. nats"). In *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2017.

[67] Ivan Slivar, Lea Skorin-Kapov, and Mirko Suznjevic. Cloud gaming qoe models for deriving video encoding adaptation strategies. In *Proceedings of the 7th international conference on multimedia systems*, pages 1–12, 2016.

[68] Rajiv Soundararajan and Alan C Bovik. Video quality assessment by reduced reference spatio-temporal entropic differencing. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(4):684–694, 2012.

[69] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. From theory to practice: Improving bitrate adaptation in the dash reference player. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(2s):1–29, 2019.

[70] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[71] Neil Stewart, Christoph Ungemach, Adam JL Harris, Daniel M Bartels, Ben R Newell, Gabriele Paolacci, Jesse Chandler, et al. The average laboratory samples a population of 7,300 amazon mechanical turk workers. *Judgment and Decision making*, 10(5):479–491, 2015.

[72] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[73] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 272–285, 2016.

[74] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[75] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[76] Beth Trushkowsky, Tim Kraska, Michael J Franklin, and Purnamrita Sarkar. Answering enumeration queries with the crowd. *Communications of the ACM*, 59(1):118–127, 2015.

[77] Matteo Varvello, Jeremy Blackburn, David Naylor, and Konstantina Papagiannaki. Eyeorg: A platform for crowdsourcing web quality of experience measurements. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 399–412, 2016.

[78] Yilin Wang, Sasi Inguva, and Balu Adsumilli. Youtube ugc dataset for video compression research. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–5. IEEE, 2019.

[79] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[80] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[81] Chen-Chi Wu, Kuan-Ta Chen, Yu-Chun Chang, and Chin-Laung Lei. Crowdsourcing multimedia qoe evaluation: A trusted framework. *IEEE transactions on multimedia*, 15(5):1121–1137, 2013.

[82] Mai Xu, Ali Borji, Ce Zhu, Edward Delp, Marta Mrak, and Patrick Le Callet. Introduction to the issue on perception-driven 360 video processing. *IEEE Journal of Selected Topics in Signal Processing*, 14(1):2–4, 2020.

[83] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 495–511, 2020.

[84] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. Neural adaptive content-aware internet video delivery. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 645–661, 2018.

[85] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno 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*, pages 325–338, 2015.

[86] Junyong You, Touradj Ebrahimi, and Andrew Perkis. Attention driven foveated video quality assessment. *IEEE Transactions on Image Processing*, 23(1):200–213, 2013.

[87] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. In *European conference on computer vision*, pages 766–782. Springer, 2016.

[88] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2472–2481, 2018.

[89] Yuanhuan Zheng, Di Wu, Yihao Ke, Can Yang, Min Chen, and Guoqing Zhang. Online cloud transcoding and distribution for crowdsourced live game video streaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(8):1777–1789, 2016.

[90] Kaiyang Zhou, Yu Qiao, and Tao Xiang. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A Dataset

Figure 19 provides screenshots and descriptions of the 16 source videos used in our dataset. Table 1 summarizes the test videos.

Name	Genre	Length	Source dataset
(a) Basket1	Sports	3:40	LIVE-MOBILE
(b) Soccer1	Sports	3:20	LIVE-NFLIX-II
(c) Basket2	Sports	3:40	YouTube-UGC
(d) Soccer2	Sports	3:40	YouTube-UGC
(e) Discus	Sports	3:40	YouTube-UGC
(f) Wrestling	Sports	3:40	YouTube-UGC
(g) Motor	Sports	3:40	YouTube-UGC
(h) Tank	Gaming	3:40	YouTube-UGC
(i) FPS1	Gaming	3:40	YouTube-UGC
(j) FPS2	Gaming	3:40	YouTube-UGC
(k) Mountain	Nature	1:24	LIVE-MOBILE
(l) Animal	Nature	3:40	YouTube-UGC
(m) Space	Nature	3:40	YouTube-UGC
(o) Girl	Animation	3:40	YouTube-UGC
(n) Lava	Animation	3:40	LIVE-NFLIX-II
(p) BigBuckBunny	Animation	9:56	WaterlooSQOE-III

Table 1: *Summary of the test video set.*

B MTurk rating sanitization

As mentioned in §4, the user ratings can be noisy and biased for multiple reasons. To get reliable quality ratings, we take the following steps.

- *Rejection criteria:* There is always one *reference* video as one (at a random position) of the rendered videos in each survey. The reference video has the highest quality (highest bitrate and no rebuffering) and if a Turker gives any other rendered video a higher rating than the reference video, we will reject the scores of this Turker. We will reject a Turker if one of the rendered videos is viewed for less than its length (*i.e.,* the video is not fully watched). The rejection criteria help to ensure that the Turkers give quality-induced ratings after watching each video in full. When a Turker's data is rejected, the Turker will not receive payment.[7]
- *Randomizing viewing orders:* Every time a Turker joins the survey, we randomize the order in which rendered videos are shown. This allows us to perform postanalysis to check (and confirm) that the position of a video within a survey has little impact on its rating.
- *Biases by Turker population:* We also check that about 43.8% (or 67.3%) ratings from Turkers who participate our survey only once (or at most twice). This confirms that the Turker pool is large enough avoid small population bias. This corroborate with the sanity-check result that on average MTurk quality ratings are strongly correlated with in-lab survey results.

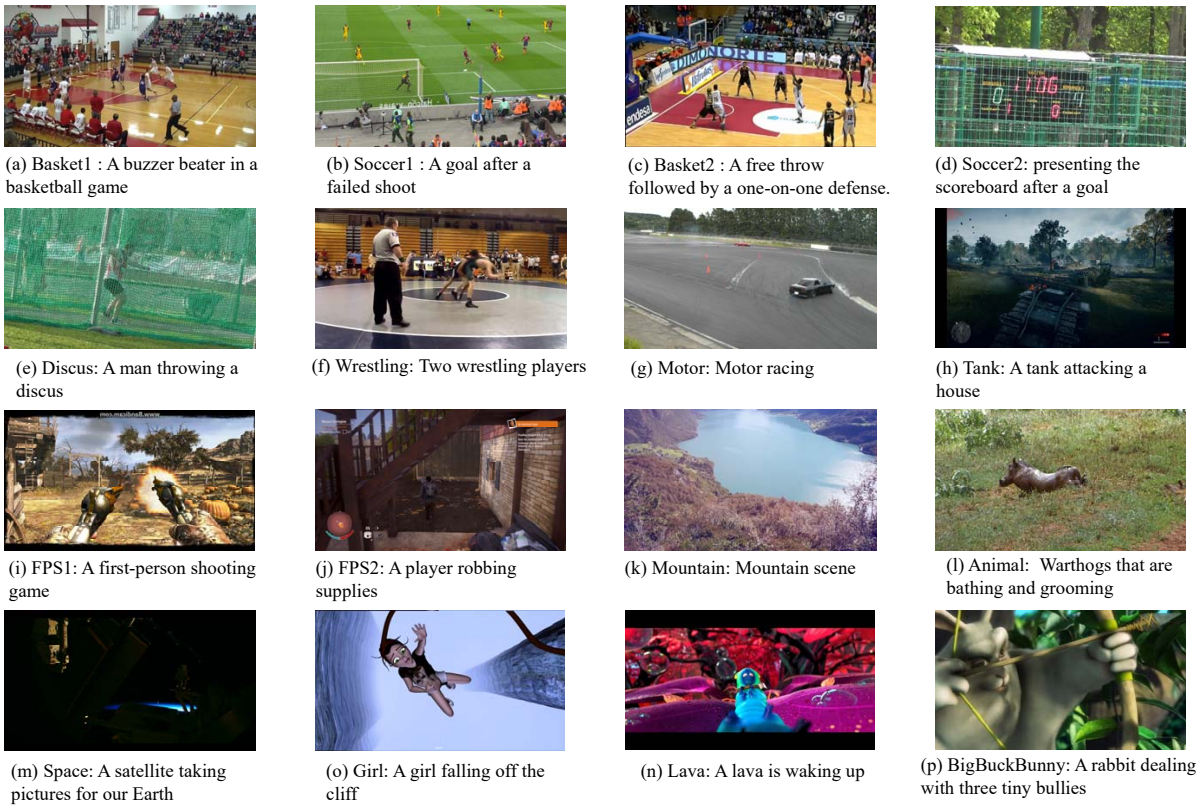
To prevent people from gaming the system by sitting on a job for too long, we pay each participant a fixed amount equal to the fixed hourly rate times the estimated amount of time needed for the survey (which is proportional to the total length of the videos per participant). We use an hourly rate of $10 (roughly matching the minimum wage standard in our state) considering that <4% Turkers are paid more than $7.25/h [40], though we have not explored the impact of raising/lowering this wage. Of course, participants will not get paid if they only watch part of the videos.

C Lessons from MTurk Experiments

We conclude with key lessons learned from running MTurk experiments and focus on the difference between in-lab and crowdource-based QoE studies.

How many Turkers are needed: An essential tradeoff between crowdsourcing and in-lab study (*e.g.,* [21,31]) is crowd-

[7] Of course, this criteria is highlighted on the first page of the survey and we have received disputes from Turkers but not often.



(a) Basket1 : A buzzer beater in a basketball game

(b) Soccer1 : A goal after a failed shoot

(c) Basket2 : A free throw followed by a one-on-one defense.

(d) Soccer2: presenting the scoreboard after a goal

(e) Discus: A man throwing a discus

(f) Wrestling: Two wrestling players

(g) Motor: Motor racing

(h) Tank: A tank attacking a house

(i) FPS1: A first-person shooting game

(j) FPS2: A player robbing supplies

(k) Mountain: Mountain scene

(l) Animal: Warthogs that are bathing and grooming

(m) Space: A satellite taking pictures for our Earth

(o) Girl: A girl falling off the cliff

(n) Lava: A lava is waking up

(p) BigBuckBunny: A rabbit dealing with three tiny bullies

Figure 19: Summary of source videos in our dataset. They span four genres: sports (a - g), gaming (h - j), natural (k - m), and animation (n - p). They are compiled randomly from public QoE datasets: LIVE-MOBILE [36](a,k), LIVE-NFLX-II [21] (b, n), YouTube-UGC [78] (c - j and l - o); WaterlooSQOE-III [31] (p).

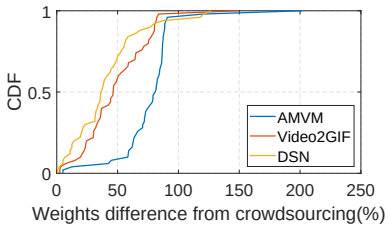


Figure 20: Chunk weight difference

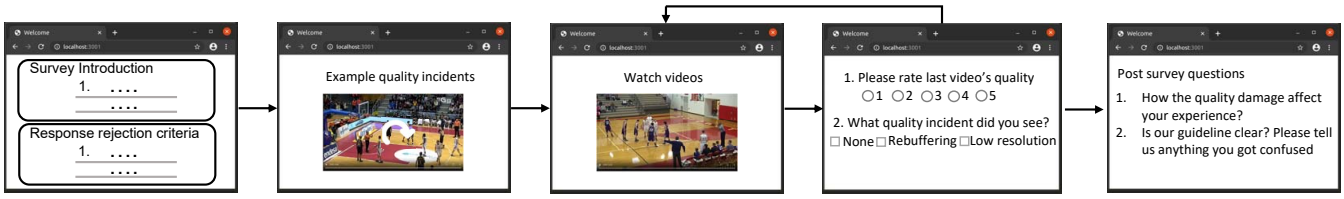
sourcing can easily find more participants for us but their work is less reliable. So in chooseing between in-lab and crowdsourcing, a key parameter is how many Turkers are needed to get a reliable signal. We did a head-to-head comparison with WaterlooSQOE-III [31] and found that as far as QoE assessment is concerned, we need 17% more Turkers to reduce the variance of QoE rating down to the same level if the same video is rated in an in-lab study.

Reputation matters: MTurk platform (and other crowdsourcing tools) dramatically cuts the cost of making our survey visible to potential Turkers, but if Turkers sign up slowly this can slow down the whole process. The speed at which they sign up depends largely on the reputation of the publisher (us) and in general, Turkers are more willing to sign up if the publisher historically has a low rejection rate. This means we must be very clear upfront about the studies rejection criteria.

Reputation is crucial to get quality feedback. We follow a common practice (*e.g.,* [53]) and restrict ourselves to *master Turkers*, a class of *reliable* Turkers who have at least participated in 1000 surveys and whose overall rejection rate is lower than 1%. It also corroborates our experience: rejection rate from these Turkers over 4× lower than normal Turkers, though their hourly rate is generally higher than normal Turkers.

D Implementation

Automation of MTurk tests: We implement the pipeline shown in Figure 8 in Python (for the logic) and Javascript (for the video server). Given a source video, it first creates the rendered videos by adding specific low-quality incidents in the source video (via ffmpeg). It then uploads these videos to a video server, from which MTurkers later download the video. After that, it generates a unique link for this campaign and posts it on the MTurk website (the only step that requires manual intervention). MTurkers can join the test by clicking the link, which redirects them to our video server. Once an



(a) Instruction page (b) Tutorial page (c) Video player page (d) Rating page (e) Post survey question page

Figure 21: *A diagram of our QoE survey interface. In each survey, a participant is asked to rate K rendered videos; after watching each rendered video, a participant is asked to rate its quality on the scale of 1 (worst) to 5 (best).*

MTurker has rated all assigned videos, the server logs the ratings and notifies us. Once enough ratings are received, the server trains the per-chunk weights as described in §4.2.

Single survey procedure: As shown in Figure 21:

- (a) Each survey starts with the instructions and rejection criteria under which ratings of a MTurk will be rejected. Each MTurker is expected to read these instructions carefully.
- (b) The MTurker will then watch an example video that includes a quality incident so that they know what their ratings should be based on.
- (c, d) After that, the MTurker will be asked to watch a sequence of rendered videos (determined by the scheduler) and, after each video, rate the quality in a scale of 1-5.
- (d) Finally, the MTurker will do an exit survey.

E More discussion on saliency models

Saliency models used in §2.3: We test four models in total, one traditional motion-based heuristic (AMVM [52]), one highlight detection models (video2GIF [39]), and a video summarization model (DSN [90], dppLSTM [87]). We used the pretrained models of Video2GIF (https://github.com/gyglim/video2gif_code), DSN (https://github.com/KaiyangZhou/pytorch-vsumm-reinforce), and dppLSTM (https://github.com/kezhang-cs/Video-Summarization-with-LSTM).

Intuition behind saliency score vs. quality senssitivity: While better saliency models will be proposed, we argue that these vision tasks can be misaligned with video quality sensitivity. For example, in the soccer video (Figure 1), the part right before the goal is most quality sensitive, but the highlight detection models and the motion-based heuristics believe the scenes showing the audience are the most important (probably because they show more human movements). Video summarization models pick all diverse moments of a video, but many of them may not be quality-sensitive. For example, in the same soccer video, the video summarization models identify every shot, rewind, and celebration clip as important, but the users pay more attention to shots that might score a goal.

We also acknowledge the potential use of the viewership information to detect the video highlights. However, while popularity of a chunk is closely related to highlights (or high interestingness parts) of a video, but, as we found in §2.3 and §7.3, these incidents may not perfectly align with users' sensitivity with video quality. Here are two examples specifically regarding the potential misalignment between content popularity and quality sensitivity.

- Example 1: A less popular part of a video can still be quality-sensitive. The "animal" video in our dataset is a part of videos about the wildlife in Africa, and it is not supposed to be as "popular" or "interesting" as other scenarios such as the lions chasing antelopes. But the users are still highly quality-sensitive where the warthogs are jumping into a small pond for bathing.
- Example 2: A quality-sensitive part video may be a small fraction of a popular segment. One of the soccer videos in our dataset is a compilation of highlight moments from a long game, so all its content is supposed to be "popular". However, we still see that have heterogeneous sensitivity to its chunks.

F FAQ

From the early feedback of the draft, we have identified several minor yet important concerns:

- *How do you design these interventions (is this based on the content of the video or is it random, if random why do you expect it to be sufficient)?* The scheduler (described in §4.3) determines the exact quality incidents and what time points in the video the incidents are applied (the Figure 9 gives a schematic example). The algorithm does not randomly choose where to add intervention. The first step of the schedule will scan the whole video (by introducing intervention at every chunk) to automatically identify chunks that share similar weights (so only one of them will proceed to the second step), so the behavior of the algorithm will be dependent on the video content. However, the same scheduling algorithm is applied to different video content.
- *In Section 2.2, it says that "even the most accurate QoE model has over 10.4% error on average." Is there a way to help the reader grasp how bad this is?* Figure 2 shows that a 10.4% error in the QoE model is significant, since it can lead the QoE model to mistakenly rank the QoE of more 10% rendered video pairs (which can mislead ABR algorithms in practice). The 3% percent prediction error will be much better. In Figure 2, we can see that a 5% prediction error could only cause 3% discordant video pairs in their QoE.

- *This is encoded in the vision community in a slightly different way: saliency models. Video saliency models is a model that tells you where a user is more likely going to be looking at. For example, if there is an "action" seen where a ball is moving, then the ball movement is where the user is going to look at and not the background.* The this general concern regarding saliency models is discussed in §E. To the specific point of using action or interesting incidents as indicators, we have included motion-based heuristics (AMVM) and highlight detection models (Video2GIF, DSN) that might automatically identify these incidents. Video saliency models can identify the highlightness or interestingness of chunks (or frames), which seem to indicate the quality sensitivity of the chunks. Technically, AMVM uses the motion-based features (i.e., magnitude of the motion vectors) to identify the object movement between frames, while Video2GIF uses the input from C3D features to get the interestingness of the chunks, and DSN uses GoogLeNet features to detect the interestingness of video chunks. Note that C3D features can encode the scenes, objects and the actions of the video chunks, and it is widely used as the training set of many other video saliency detection models. For example, in the soccer video (Figure 4), the part right before the goal is most quality sensitive, but the highlight detection models and the motion-based heuristics believe the scenes showing the audience are the most important (probably because they show more human movements). Even if the model does detect all the motions, then every shot, rewind, and celebration clip will be treated as important/interesting, but the users pay more attention to shots that might score a goal.
- *The paper mainly relies on anecdotal evidence and proposes to generalize the observation made from a small scale study beyond what has been actually demonstrated to generalize.* SENSEI is tested on 16 sources videos, ranging 4 content genres, randomly selected from three public video QoE datasets. This dataset scale is on par with other papers in this space (QoE model work like Waterloo-III with 20 source videos, LIVE-NFLX-II with 15 source videos, and video streaming work like NAS with 27 source videos, Minerva with 19 source videos). the 16 videos in our dataset have very different fractions of high-sensitivity chunks (e.g., exciting moments), ranging from 20% to 60%. In addition, Section 7.4 (Figure 18(c)) tests SENSEI's performance under an even wider range of high-sensitivity chunk fractions (from 0% to 100%). It confirms that when all chunks are equally quality-sensitive or quality-insensitive, unsurprisingly the QoE gain is zero. And the QoE gain reaches the peak (20%) gain at 30% sensitive chunks.
- *If you are willing to pay more then, there are other techniques such as video super-resolution such as NAS and a subsequent work that would appear in sigcomm 2020. There they pay cost to train the super-resolution network and use client computation to improve QoE. The improvement their is also video dependent.* While both SENSEI and NAS and LiveNAS incurs additional costs for higher quality, they are complementary. They optimize the visual quality by leveraging client-side GPU capacity to run ML to recover a high-res video from a low-res one, while SENSEI is to trade the quality of low sensitivity chunks for the higher quality at high sensitivity chunks. In addition, SENSEI explores a different type of cost: its crowdsourcing cost is per (popular) video, whereas NAS incurs GPU cost per user.
- *One thing that might be useful is to use audio as a way to figure out at which scene important events in sports videos happened. Usually, commentators get quite excited about such a scene and, it is probably noticeable simply from the volumes.* Audiotrack is absolutely related to quality-sensitivity of chunks, since the audio track might tell you to focus on some objects on video. Interestingly, this is an aspect that none of the visual saliency models can capture, and our crowdsourcing-based user study has the unique advantage of asking real users to rate video quality while the sound is on.