Master's Thesis

Title

QoE Estimation Method for Video Streaming using Power Spectral Density of EEG and its Application to Adaptive Bitrate Control for QoE Improvement

Supervisor

Professor Masayuki Murata

Author

Taikei Mori

February 3, 2025

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

QoE Estimation Method for Video Streaming using Power Spectral Density of EEG and its Application to Adaptive Bitrate Control for QoE Improvement

Taikei Mori

Abstract

The number of users using the Internet is increasing every year, and the amount of traffic on the Internet is growing accordingly. In particular, video traffic has a significant impact on communication on the Internet. In recent years, the growing popularity of video streaming services such as YouTube and Netflix, which are services that handle video content, and the increase in the number of Internet users have increased the demand for communication quality on the Internet.

In response to the growing demand for communication resources, there is a limit to the expansion of them. In the area of video streaming technology, there is a technique called adaptive streaming, which is used to maintain and improve the quality of experience (QoE) of users even when communication resources are limited. In adaptive streaming, the server prepares multiple quality video files for a single video, each of which is further divided into several seconds. The client performs playback while switching the requested video quality, taking into account information on the playback device, network conditions, and other factors. The algorithm that determines how the video quality is selected is called the Adaptive Bit Rate (ABR) algorithm, and various methods have been devised to date.

Most methods assume that parameters related to video playback quality (bitrate, frequency of stops, length, etc.) and the user's QoE correspond to some function. An example is the bitrate–QoE relationship, where QoE is defined by a function proportional to the logarithm of the bitrate. Many existing ABR algorithms do not take into account that this mapping is different for each user, and they optimize for the so-called "average user". Although there are ABR algorithms that take into account the fact that the mapping

differs from user to user, they do not take into account the possibility that the mapping may differ depending on the environment or mood in which the user is viewing the video.

Our research group has been studying a method to estimate a user's QoE during video viewing in real-time by using an estimation model based on electroencephalography (EEG). In this thesis, we focused on the power spectrum of the EEG in estimating whether the user is dissatisfied with the current playback quality or not. We built and evaluated our estimation model for binary classification of QoE using the power spectrum as input features. As a result of experiments, an estimation accuracy of 65% was achieved on average. We also devised a bitrate adaptation strategy using this QoE estimator and implemented it on a DASH player. Experiments conducted within the research group confirmed that the control was able to adapt to user feedback during video viewing.

Keywords

Quality of Experience (QoE)

Electroencephalogram (EEG)

Adaptive Streaming

Bitrate Adaptation

Adaptive Bitrate (ABR)

Dynamic Adaptive Streaming over HTTP (DASH)

Contents

List of Figures

List of Tables

1 Introduction

According to the Cisco Annual Internet Report [5], the number of Internet users has been increasing year by year, growing from 3.9 billion in 2018 to 5.3 billion in 2023. This indicates that 66% of the world's population was using the Internet as of 2023, and this percentage is expected to continue rising. In Internet communications, video content particularly has a significant impact on network traffic, with streaming a 2-3 hour video in HD quality generating traffic equivalent to a household's daily traffic [5]. As 4K and even higher quality video content becomes more prevalent, the impact of video streaming services on network communications is expected to become increasingly significant.

To address the increasing demands on Internet communication quality, driven by growing user numbers and the popularization of video content services such as YouTube and Netflix, a technology called adaptive streaming has been developed to maintain and improve users' quality of experience (QoE) within limited communication resources. In adaptive streaming, streaming servers prepare multiple quality versions of a single video file, with each version divided into segments of several seconds in length. The client then plays the video while dynamically switching the requested quality level based on network conditions and other factors. The algorithms that determine which quality level to select are called Adaptive BitRate (ABR) algorithms, and they have been extensively studied in various research [6–8].

FESTIVE [6] is a throughput-based ABR algorithm that estimates network bandwidth during playback and determines bitrate based on these estimates. In contrast, BOLA [7] is a buffer-based ABR algorithm that differs from bandwidth-estimating algorithms like FESTIVE by controlling playback solely based on the state of the playback buffer. BOLA has been implemented in dash.js, a reference client implementation for DASH written in JavaScript. FastMPC and RobustMPC, proposed in [8], are hybrid ABR algorithms that utilize both network estimation and buffer information.

In the algorithms presented in the aforementioned papers [6–8], video playback quality parameters (such as bitrate, frequency and duration of stalls, etc.) are mapped to user QoE using functions designed with weighted sums, logarithms, or similar approaches. These functions are designed to be uniform across different users. This QoE mapping is based on

the assumption of an "average user" to accommodate the majority of users. As a result, these algorithms are designed for this "average user" and may not account for individual preferences and variations in user preferences.

Reference [9] tackled this issue. In Ref. [9], 90 participants were asked to watch videos and assign quality scores, and the degree of variation in scores among users was analyzed. The results showed that in approximately 90% of participant pairs, there was no strong correlation between the assigned scores. This suggests that the concept of an "average user" does not adequately represent the perception of all users. Based on these findings, The authors of [9] propose an ABR algorithm that assumes variations among users in their perception of video playback quality.

In Ref. [9], user-specific QoE is pre-modeled and incorporated into adaptive streaming. However, in reality, in addition to differences in quality perception among users, factors such as the user's environment during video playback and their mood may also influence their perception of quality. Therefore, to optimize users' QoE through adaptive streaming, it is necessary to reflect users' real-time preference in the control process during video playback. To incorporate users' real-time preference into the control process, QoE must be estimated in real-time using some method. Our research group has been studying a QoE estimation method using EEG [10, 11], but there remains some challenges (Sec. 2.2.4).

In this thesis, building on this background, this study aims to develop a QoE estimation method that detects user dissatisfaction in real-time during video playback using EEG data. We propose a method for estimating a user's QoE during video viewing that does not involve an explicit response of QoE by the user. As described below in Sec. 2.2.4, inputting QoE by buttons or verbally responding to a user's QoE will cause EEG changes associated with those actions. This induces estimation errors. For EEG feature extraction, we use the power spectra before and after events related to video playback, referring to the experiments in [12]. Furthermore, based on this real-time estimation, we implement an adaptive streaming system on DASH aiming to improve users' QoE through this system. We conduct experiments to confirm that the proposed control is able to adapt to user feedback during video viewing.

2 Related work

2.1 MPEG-DASH

MPEG-DASH (MPEG Dynamic Adaptive Streaming over HTTP) is one of the most standard adaptive streaming (HTTP Adaptive Streaming, HAS) technologies using the HTTP protocol, and is standardized by the International Organization for Standardization ISO/IEC [13]. By dynamically determining the image quality of the video to be played from the network state, the remaining amount of the client's playback buffer, etc., stable video playback is possible even when the communication environment suddenly deteriorates during video playback.

How Adaptive Streaming works in DASH MPEG-DASH defines two types of files: MPD (Media Presentation Description) and segments. Segments are small files that make up the video to be delivered.

A segment refers to one of the video files that are created by dividing a single video into short segments. MPD is an XML file that holds the metadata necessary for streaming video playback, such as the encoding method and bitrate at the time of video encoding, the segment division unit and acquisition method, etc. In order to provide a streaming service, it is necessary to encode the provided video in advance with multiple qualities, and divide the video of each quality every few seconds. Information on these video qualities, division positions, etc. are described in the MPD file.

Figures 1 and 2 show a diagram illustrating how MPEG-DASH works. The playback client first acquires the MPD file of the video to be played from the server and obtains segment information. Based on this information, the playback client requests the server to specify the video quality of the segment at each segment position, taking into account the network communication quality and other factors. The HTTP protocol is used for these communications, including the transmission of video files from the server to the client.

2.2 Our Previous QoE Estimation Method

Our research group has been studying a method for estimating the QoE of users during video playback [10, 11] using EEG. In this section, we provide an overview of the method

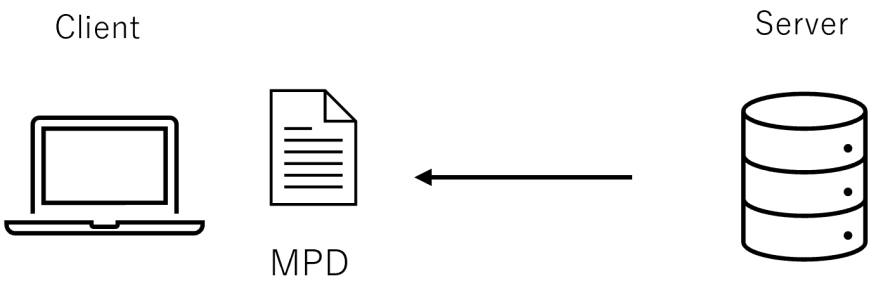

Figure 1: Acquire a MPD

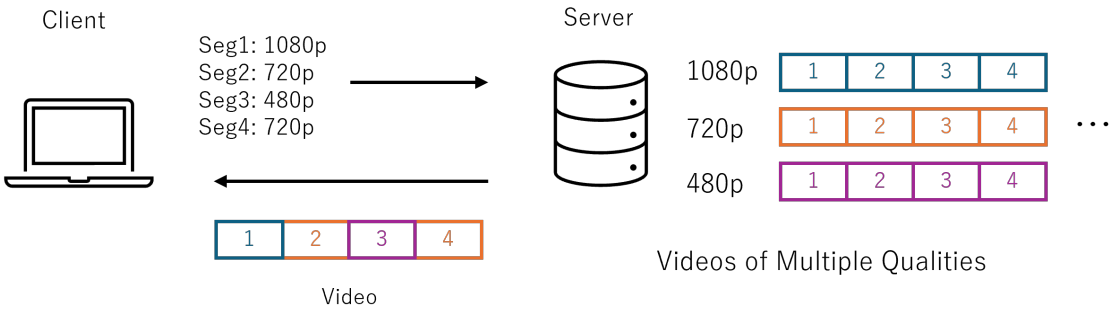

Figure 2: Request Segments

proposed in [10, 11].

2.2.1 EEG Headset

We use Emotiv EPOC X (Fig. 3), a headset equipped with 14-channel sensors capable of measuring EEG signals. EEG data can be recorded from 14 electrode positions (AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4), capturing brain activity from various locations on the scalp (Fig. 4). The EEG data is initially sampled at 2048 sps (samples per second) within the device and is down-sampled to either 128 or 256 sps depending on the output settings. The measurement bandwidth ranges from 0.16 Hz to 43 Hz.

Additionally, the EPOC X is equipped with motion sensors (accelerometer, gyroscope, and magnetometer), which can assist in artifact removal. The sampled data is transmitted to a computer via either a Bluetooth connection or a USB dongle. EmotivPro, a software developed by Emotiv, can be used to record both EEG and motion data transmitted from

the EPOC X.

Figure 3: EPOC X

2.2.2 Overview of Estimation Model

The proposed QoE estimation model in [10,11] is based on a support vector machine that takes multiple features extracted from the user's EEG signals as input and outputs one of three QoE levels: high, neutral, or low.

Input Features For the 14 types of EEG data, various features such as band power, power spectral density, and discrete wavelet transform values are computed. The extracted features are the maximum, minimum, median, and variance of those values, resulting in a total of 546 features.

In the proposed method in [10,11], instead of using all 546 features as input, a subset of features is selected for estimation. This selection aims to reduce computational time and prevent overfitting. The feature selection process is performed using a Genetic Algorithm (GA), which optimizes the cross-validation score of the model trained with the selected features as the objective function.

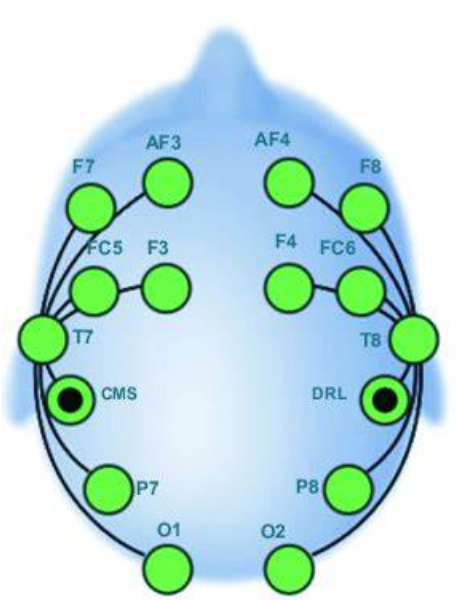

Figure 4: Electronodes of EPOC X

2.2.3 Performances

In [11], QoE estimation models were created for multiple participants, and the recall rate for detecting low QoE was measured using validation data. According to [11], the recall rate reached a maximum of 74%, with an average of 49.3% and a minimum of 19%, indicating significant individual differences.

2.2.4 Challenges

Initially, we planned to implement the ABR system based on the model of [10, 11]. However, when performing QoE estimation for bitrate control, the prediction model consistently outputted "high" level, failing to achieve the estimation accuracy reported in [10,11]. Upon investigation, one of the key experimental issues that emerged as a possible cause was as follows.

In the development process of an estimation model described in [11], participants were required to input their QoE levels while watching videos to create training data. However, this method raised several concerns. The act of inputting QoE involves not only the user's emotional state at that moment but also cognitive processes such as decision-making and physical actions to provide the input. These cognitive and motor activities can influence

brainwave recordings, potentially leading to the capture of brain activity related to "QoE input actions" rather than actual dissatisfaction.

If a machine learning model is trained on such data, it may learn to recognize brain activity associated with the act of QoE input rather than accurately detecting dissatisfaction. In other words, instead of identifying moments of dissatisfaction, the model risks responding to the physical action of providing QoE input. Furthermore, since QoE input does not occur during actual rate control using the trained model, the system encounters a different situation than in the training phase. As a result, even when a user experiences dissatisfaction, the system may fail to detect it accurately.

For those reasons, including QoE input actions in training data poses a risk of hindering accurate dissatisfaction detection. Therefore, necessary was a data collection method that does not rely on user actions.

3 QoE Estimation for Video Streaming using EEG

3.1 Design Overview

In this study, we develop an estimation model that detects user dissatisfaction with video playback based on brainwave data collected during video viewing. The model's input and output are summarized as follows:

- **Input** Features computed from brainwave data. The method for feature computation will be described in §3.3.
- **Output** A binary classification result – either 0 (dissatisfied) or 1 (satisfied).

3.2 Preparation of Training Data

3.2.1 Experiment

We describe the experimental procedure used to prepare training data for the QoE estimation model.

Participants watches videos while wearing an EEG headset (EPOC X, Fig. 3). The videos used in the experiment are from Netflix Open Content [1] and some Vocaloid music videos (MVs). Some producers of Vocaloid MVs make their video and audio materials available for derivative works, so we uses them. Each video was approximately 3 to 10 minutes in length, and a total of 8 videos were used. Examples of the videos used are shown in Figs. 5 and 6.

Figure 5: Netflix Video example [1]

Figure 6: Vocaloid MV example [2]

The prepared videos were edited to include multiple instances of buffering animations (Fig. 7) to simulate pauses during rebuffering. Based on [14], each video contained pauses lasting approximately 3 to 10 seconds at specific points, occurring 1 to 5 times within a 10-second window. These inserted pauses were designed to induce noticeable QoE degradation at their respective positions.

Multiple short-interval pauses (within 10 seconds) were grouped as a "pause group," and the interval between consecutive pause groups (from the end of the last pause to the start of the next) was set to at least 30 seconds. The editing structure is illustrated in Fig. 8.

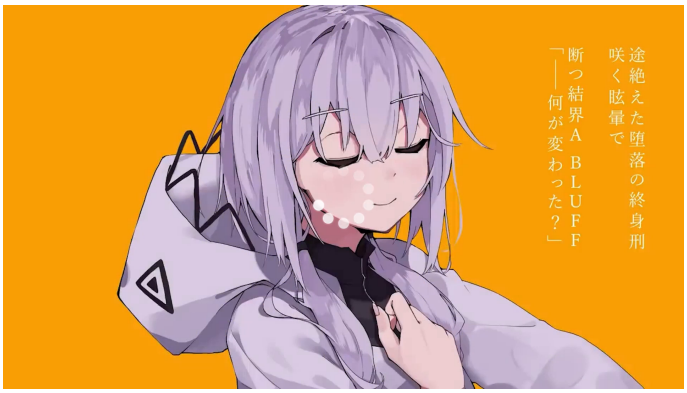

Figure 7: Rebuffering Emulation Example [3]

Figure 8: Illustration of Video Edit [4]

After watching each video, participants are asked to complete a questionnaire regarding their experience with the content. The survey includes the following questions:

- Content of the video (7-point scale): boring, uninteresting ↔ exciting, interesting

- Overall playback quality (7-point scale): Poor ↔ Good
- Which aspects influenced your perception of playback quality? (Multiple choices allowed)
  - Sound quality
  - Video quality
  - Duration of pauses
  - Frequency of pauses

An important point is that, in this experiment, no additional tasks (e.g., inputting QoE) are specified while the participants are watching the video. This measure ensures that the brainwave data does not include any influence from the QoE input behavior, as explained in Sec. 2.2.4.

3.2.2 EEG Epoching and Labeling

The brainwave data obtained during watching video is classified into the following two:

- Brainwaves during the pause group (labeled as "dissatisfied").
- Other (labeled as "satisfied").

For the brainwaves labeled as "dissatisfied," a random 3-second segment of brainwave data is epoched, and, with the start time of the corresponding pause group set to 0, brainwaves from -4 seconds to -2 seconds are also epoched as baseline brainwaves. In this thesis, the former is referred to as the "target brainwave" and the latter as the "baseline brainwave."

Similarly, a random 3-second segment of brainwave data is epoched from the brainwaves labeled as "satisfied," and the baseline brainwave is saved from -4 seconds to -2 seconds based on the start time of the extracted segment. Both the target brainwave and the baseline brainwave are stored as pairs.

In this process, brainwave data obtained from videos where participants indicated they were bored are excluded. Additionally, among the epochs labeled as "dissatisfied"

those that did not mention the duration or frequency of playback interruptions in the post-viewing survey are also excluded.

3.3 Feature Extraction from EEG

A classification model for predicting whether the user is dissatisfied or not with video playback is built based on the relative power (dB) referenced to the power immediately preceding rebuffering as the baseline, as mentioned in [12]. The relative power is calculated by transforming the EEG signal into a time-frequency representation using complex wavelet transform, then computing the power for each frequency component, and averaging the power within the frequency band of interest. The calculation is performed as follows:

First, a mother wavelet $\psi(t)$ is prepared, and by scaling it, we obtain the wavelet $\psi_f(t)$ for the desired frequency f. The frequency f is varied from 1 to 30 Hz with a step of 0.5 Hz, preparing multiple wavelets for the 1-30 Hz range.

The prepared wavelets are then used to perform the complex wavelet transform on the EEG signal $x(t)$. Specifically, the transformation is performed as shown in Eq. (1), where $*$ represents the complex conjugate.

$$T_f(x) = \int_{-\infty}^{\infty} x(t)\psi_f^*(t-x)\,dt \tag{1}$$

The EEG signal $T_f(x)$ after the wavelet transform has a complex number representation. By calculating the squared distance from the origin, the time-frequency representation of the power of the EEG signal, $P(t, f)$, is obtained Eq. (2).

$$P(t, f) = |T_f(t)|^2 \tag{2}$$

The time-domain average of the baseline EEG power is calculated to obtain the baseline $P_B(f)$. Similarly, the time-domain average of the EEG power corresponding to the baseline is also calculated and denoted as $P_M(f)$. Using these, the relative power of the EEG signal with respect to the baseline in the interval of interest is computed as shown in Eq. (3). Since the EEG signal can be obtained from 14 electrodes, as shown in Fig. 4, the relative power can be calculated for each electrode.

$$R(f) = 10 \times \log_{10}(P_M(f)/P_B(f)) \tag{3}$$

The features used in the estimation model are limited to 3 of the 14 electrodes, and for each of these three electrodes, the frequency band of interest is selected from the alpha band (8-13 Hz), beta band (14-30 Hz), theta band (4-7 Hz), or delta band (0.5-3.5 Hz). The final features are obtained by averaging the relative power within these selected frequency ranges. Therefore, for each subject, three features (e.g., F3 (14-30 Hz), O1 (0.5-3.5 Hz), and P8 (14-30 Hz)) are used in the estimation model.

In the wavelet transform described above, the mother wavelet we use is Complex Morlet Wavelet (Eq. (4)) with parameters $B = 0.2$ and $C = 1.0$.

$$\psi(t) = \frac{1}{\sqrt{\pi B}} \exp\left(-\frac{t^2}{B}\right) \exp\left(2\pi C t i\right) \tag{4}$$

Note that the parameters of Complex Morlet Wavelet used here differ from those in [12]. This is due to simplify the implementation with the library used in this study. However, we confirmed that the estimation accuracy remains almost the same when using the parameters from [12].

3.4 Model Building and Training

3.4.1 QoE Estimation Model

We use a Support Vector Machine (SVM) as the estimation model, based on a previously developed model by our research group [11].

3.4.2 Feature Selection and Hyperparameter Tuning

As mentioned in §3.3, without reducing the features, we would obtain (number of electrodes)$\times$ (4 frequency bands) features. In this study, those would be $14 \times 4 = 56$ features. To select promising features, we use a hyperparameter optimization framework to reduce the features. The optimization framework explores which features provide the highest accuracy using k-fold cross-validation. In addition to feature selection, hyperparameters such as the kernel function and cost parameter C of the SVM are also optimized.

3.5 Implementation

We implemented the estimation model using Python with some libraries. The SVM model is implemented using scikit-learn, feature calculation is performed by PyWavelets, and hyperparameter tuning is done by Optuna.

4 Bitrate Adaptation based on Estimated QoE

4.1 System Overview

We propose an ABR (Adaptive Bitrate) control system that can reflect changes in perception of quality due to some factors such as playback environment or user mood in real-time control. By using estimated QoE as feedback and updating the parameters of the QoE function used for bitrate decision, the system incorporates the user's reactions into the control.

The components of our ABR control system proposed in this thesis are as follows:

- **Video Player** acquires video files from a server and plays videos. Proposed ABR Algorithm is implemented in this component.
- **QoE Estimator** estimates QoE of a user watching videos. It outputs 0 or 1, each of which indicates "dissatisfied" or "satisfied" respectively.
- **QoE Function Updater** Based on an estimation from **QoE Estimator**, this component updates the parameters of QoE functions that maps quality metrics and user QoE. It receives playback metrics from **Video Player**, and an estimated QoE from **QoE Estimator**, so that it can update the parameters.

Fig. 9 illustrates the components and their interactions.

4.2 Bitrate Adaptation Algorithm for Improving QoE

The goal of our bitrate adaptation is to optimize QoE while considering available network resources. We devised an ABR algorithm using estimated QoE, based on Model Predictive Control (MPC) approach proposed in [8]. To optimize QoE, estimated QoE is used to model real-time mappings from video bitrate and pause duration to user QoE.

Proposed Algorithm Let the video to be played be divided into K segments. Let $\mathcal{R}$ be the set of available bitrates, and the k-th segment select a bitrate $R_k \in \mathcal{R}$. The remaining playback time in the buffer at time t is denoted by $B(t)$. Let $Q_1(R)$ represent the bitrate–QoE mapping function, which is updated in real-time by the estimated QoE, and

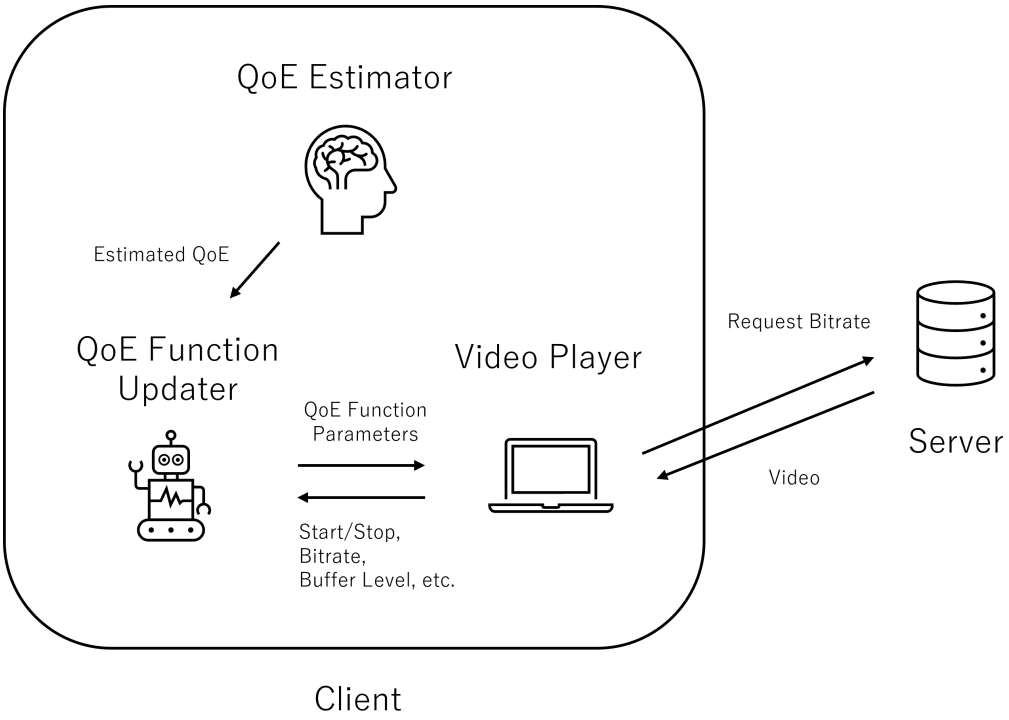

Figure 9: The Architecture of our ABR System

$Q_2(x)$ represent the pause-duration–QoE mapping function, which is similarly updated in real-time by the estimated QoE.

When determining the bitrate of the k-th segment, let the time at that moment be t_k. Denote the predicted throughput from time t_k to t_{k+N} as $\hat{\mathbf{T}}_{[t_k,t_{k+N}]}$.

Let $\mathrm{RT}(B(t_k),\ \hat{\mathbf{T}}_{[t_k,t_{k+N}]}, R_k, \ldots, R_{k+N-1})$ represents the function that calculates the maximum pause time that occurs during the next N segments when the bitrates are set to $R_k, R_{k+1}, \ldots, R_{k+N-1}$.

Using Eq. (5), bitrates $R_k, R_{k+1}, \ldots, R_{k+N-1}$ are selected such that the optimal QoE is obtained over the next N segments, and choose R_k as the next bitrate.

$$\underset{R_k,\ldots,R_{k+N-1}\in\mathcal{R}}{\arg\max} \Big(\frac{1}{N}\sum_{i=k}^{k+N-1} Q_1(R_i)\Big) + Q_2\big(\mathrm{RT}(B(t_k), \hat{\mathcal{T}}_{[t_k,t_{k+N}]}, R_k, \ldots, R_{k+N-1})\big) \qquad (5)$$

4.3 Parameter Update of User QoE Function based on Estimated QoE

Let T denote the estimation period of QoE. Every estimation period T, the user's QoE $q \in \{0, 1\}$ during video playback is estimated, and the estimated QoE q is recorded according to one of the following conditions:

- If the video is playing at the time of estimation, a tuple consisting of the playback label, the estimated QoE, and the current bitrate (Playback, q, b) is stored.
- If the video is paused at the time of estimation, a tuple consisting of the rebuffering label, the estimated QoE, and the elapsed time since the pause started (Rebuffering, q, r) is stored.

From the recorded QoE information, the most recent n_b records related to bitrate and the most recent n_r records related to rebuffering duration are used to derive the mappings between video bitrate and QoE, as well as between pause duration and QoE, by fitting them to sigmoid curves with reference to [12]. Let $\sigma_{a,b}(x)$ denote a sigmoid curve defined as $\sigma_{a,b}(x) = \frac{1}{1+e^{-a(x-b)}}$.

- The most recent n_b QoE records about "Playback" are used to fit the bitrate–QoE relationship to a sigmoid curve ($\sigma_{a_p,b_p}(x)$) via logistic regression, denoted as $Q_1(R)$ (e.g., Fig. 10).
- The most recent n_r QoE records about "Rebuffering" are used to fit the pause-duration–QoE relationship to a sigmoid curve ($1-\sigma_{a_r,b_r}(x)$), denoted as $Q_2(x)$ (e.g., Fig. 11).

4.4 Rebuffering Estimation

Using Algorithm 1, all rebuffering events that occur when selecting the bitrates $R_1, R_2, \ldots, R_N \in \mathcal{R}$ for the next N segments are determined. This algorithm requires not only the selected bitrates but also the current buffer length, the current rebuffering duration, and the length of one segment. Note that this algorithm assumes that the estimated throughput remains constant within the interval where rebuffering is estimated.

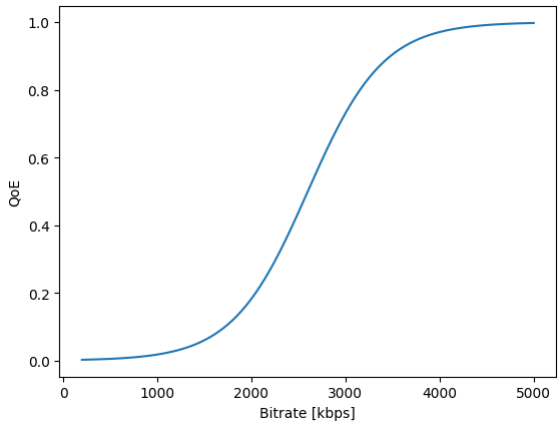

Figure 10: Bitrate–QoE function ($Q_1(R)$) example

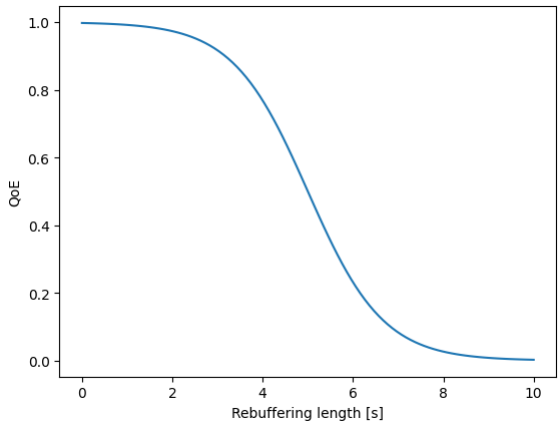

Figure 11: Pause-Duration–QoE function ($Q_2(x)$) example

Since this algorithm determines all rebuffering durations that will occur over the next N segments, we can calculate RT by the maximum of these values.

Algorithm 1 Estimate Future Rebuffering Events

Require: Require these parameters and constants.

- current_buffer_level: the current buffer level [s].
- current_rebuffering_time: the current rebuffering time [s]. if current_buffer_level > 0, this must be 0.
- selected_bitrates: an array of selected bitrates [kbps].
- estimated_throughput: an estimated throughput [kbps].
- SEGMENT_LENGTH: the length of a segment [s].

Ensure: An array of the rebuffering length.

```
rebufferings ← []
buffer_level ← current_buffer_level
rebuffering_time ← current_rebuffering_time
for k = 1 ... N do
    download_size ← selected_bitrates[k] × SEGMENT_LENGTH
    download_time ← download_size/estimated_throughput
    buffer_level ← buffer_level − download_time
    if buffer_level < 0 then
        rebuffering_time ← rebuffering_time + (−buffer_level)
        buffer_level ← 0
    end if
    if rebuffering_time > 0 then
        Add rebuffering_time to rebufferings.
    end if
    buffer_level ← buffer_level + SEGMENT_LENGTH
    rebuffering_time ← 0
end for
```

5 Evaluation

5.1 Performances of QoE Estimation

Five members of our laboratory were recruited to conduct an experiment to create an estimation model. The steps of the experiment is described in §3.2.1.

We show the performances of the estimation models in Tables. 1, 2, 3, 4, and 5. Additionally, we show features used in each model in Table 6.

Table 1: Subject 01

Metric	**QoE=0**	**QoE=1**
Precision	0.6111	0.5135
Recall	0.1692	0.8906
F1-score	0.2651	0.6514
Accuracy	0.5271	

Table 2: Subject 02

Metric	**QoE=0**	**QoE=1**
Precision	0.7059	0.6346
Recall	0.5581	0.7674
F1-score	0.6234	0.6947
Accuracy	0.6628	

Table 3: Subject 03

Metric	**QoE=0**	**QoE=1**
Precision	0.5833	0.6471
Recall	0.7447	0.4681
F1-score	0.6542	0.5432
Accuracy	0.6064	

Table 4: Subject 04

Metric	**QoE=0**	**QoE=1**
Precision	0.7500	0.6579
Recall	0.5806	0.8065
F1-score	0.6545	0.7246
Accuracy	0.6935	

Table 5: Subject 05

Metric	**QoE=0**	**QoE=1**
Precision	0.7419	0.7500
Recall	0.7667	0.7241
F1-score	0.7541	0.7368
Accuracy	0.7458	

Table 6: Selected Features

Subject	Features
Subject 01	F7(4–7Hz), T7(0.5–3.5Hz), T8(0.5–3.5Hz)
Subject 02	F3(14–30Hz), O1(0.5–3.5Hz), P8(14–30Hz)
Subject 03	O2(4–7Hz), F8(0.5–3.5Hz), AF4(0.5–3.5Hz)
Subject 04	F7(4–7Hz), T7(14–30Hz), T8(14–30Hz)
Subject 05	P7(4–7Hz), T8(14–30Hz), F4(8–13Hz)

As a result, the model achieved an estimation accuracy of up to 75%, with a minimum of 53%, and an average of 65%. When examining the Recall value at QoE = 0, which indicates the performance for detecting dissatisfaction when QoE drops, the values were a maximum of 77%, a minimum of 17%, and an average of 56%. This indicates that the developed estimation model can detect just under 60% of the declines in user QoE during video playback.

Furthermore, when checking the Precision value at QoE = 0, the maximum was 75%, the minimum was 58%, and the average was 68%. This means that when the model outputs QoE = 0, there is a slightly less than 70% chance that the user is actually dissatisfied.

Next, we examine the selected features. Different features were selected by different subjects, but the electrode positions (shown in Fig. 4) that were used by multiple subjects include F7, T7, and T8. In regarding to frequency bands, the alpha band (8–13Hz) was used less frequently, while the beta band (14–30Hz), theta band (4–7Hz), and delta band (0.5–3.5Hz) were used more evenly.

5.2 Adaptive Bitrate Control Experiments

5.2.1 Implementation of DASH Client

For the implementation of the actual system, the video player was implemented using the client-side library for DASH, dash.js [15] v4.7.4. The ABR algorithm explained in §4 was implemented using dash.js's `customAbrRule`.

As described in §4, the system components include the **Video Player** implemented with dash.js, **QoE Estimator** for estimating the user's QoE during video playback, and

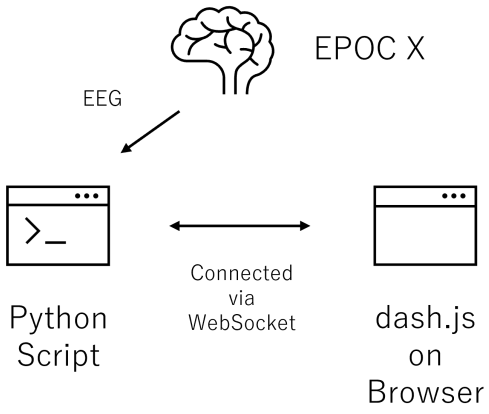

Figure 12: The Architecture of our Implemented System

QoE Function Updater for updating the parameters of the QoE function. Since **QoE Estimator** is implemented in Python, **QoE Function Updater** was also implemented in Python to use **QoE Estimator** as a Python module. The implementation of fitting the QoE function to the sigmoid function in the parameter update of **QoE Function Updater**, as explained in §4, was done using the scikit-learn library.

Due to this implementation, **Video Player** and **QoE Function Updater** run in different processes, so they communicate with each other via WebSocket. The architecture of the implemented system is shown in Fig. 12. As illustrated in the figure, the WebSocket server running **QoE Estimator** and **QoE Function Updater** is on the same machine as **Video Player**. **Video Player** connects to this WebSocket server every time it plays a video.

As for the content of communication via WebSocket, **Video Player** sends event information such as the start/stop of video playback, buffer levels, and changes in video quality to the local WebSocket server. On the other hand, the WebSocket server periodically estimates the user's QoE during video playback. Based on the estimation result and the playback status information received from **Video Player**, it determines the new QoE function parameters and sends them back to **Video Player** via WebSocket.

5.2.2 Experimental Setup

To verify the feasibility of whether bitrate control customized to individual users, we conducted an experiment with two members of our research group as subjects (Fig. 13). The details of the machines and related software used in the experiment are shown in Table 7.

Item	**Details**
Browser	Google Chrome v132.0.6834.112
OS Edition	Windows 11 Education
OS Version	24H2
OS Build	26100.2894
Processor	AMD Ryzen Threadripper 3960X 24-Core Processor 3.80 GHz
Installed RAM	32.0 GB (31.9 GB usable)
Monitor	DELL U2722DE 2560x1440, 60Hz 24bit

Table 7: Experimental Environment

In the experiment, to simulate a poor network environment, bandwidth throttling was applied using the Network Throttling feature in Google Chrome's Developer Tools. We created a Node.js script to apply the throttling using the chrome-remote-interface [16] library. The bandwidth limitations were set based on measured data of video streaming over a 3G/HSDPA mobile connection in Norway [17]. To apply the bandwidth throttling at the moment video playback starts, the Node.js script for throttling and the player were made to communicate also via WebSocket. The script was implemented to apply throttling when it receives the start playback notification from the player. The same throttling profile was used for all sessions in the experiment.

The videos played in the implemented DASH client were encoded in 6 profiles shown in Table 8, based on the experiment conducted in [18]. The encoding was done using ffmpeg [19]. Additionally, the segment length of each video in the experiment was set to 2 seconds.

Listing 1 is our encoding command of ffmpeg.

We used MP4Box [20] to generate MPD files from encoded videos.

Figure 13: Experimental Scene

Table 8: Encode Profiles

Bitrate [bits/s]	**Resolution**	**Maxrate**	**Bufsize**
300k	426x240	360k	720k
750k	640x360	900k	1800k
1200k	854x480	1440k	2880k
1850k	1280x720	2220k	4440k
2850k	1920x1080	3420k	6840k
4300k	2560x1440	5160k	10320k

Listing 1 FFmpeg encode command

```
ffmpeg -i <Input> \
    -vf "scale=<Resolution>" \
    -vcodec libx264 -vb <Bitrate> -maxrate <Maxrate> -bufsize <Bufsize> \
    -r <Frame rate of the input> \
    -x264opts "no-scenecut" -g 15 \
    -acodec aac -ac 2 -ab 128k \
    -frag_duration 2000000 \
    -movflags "empty_moov" \
    <Output>
```

5.2.3 Results

Figures 14 and 15 show the two metrics of the Subject 02's experiment, each of which corresponds to the metrics of the first playback and of the later playback, respectively. In the experiment, buffer levels, changes in quality, estimated QoE, and parameters of QoE Function are stored. These figures show them.

Descriptions of the Figs. 14 and 15 The figure consists of five graphs, with the horizontal axis representing time, where the time of playback start is set to 0. The top graph shows the remaining buffer level at each playback time. It indicates that buffering occurs when the buffer level reaches 0. The second graph from the top shows the changes in the video bitrate and the throughput used in network throttling in Developer Tools. The orange line represents the bitrate, and the red line represents the throughput. The third graph from the top shows the changes in the estimated QoE, which was periodically estimated during video playback by the estimation model from §3. The fourth graph from the top shows the changes in the Bitrate–QoE relationship function $Q_1(R)$ which is used in the ABR algorithm explained in §4. The color of the heatmap corresponds to the QoE values on the color bar on the right of the graph, where yellow indicates a QoE close to one and navy blue indicates a QoE close to zero. The bottom graph shows the changes of the Pause-Duration–QoE relationship function $Q_2(x)$, which is also used in the ABR

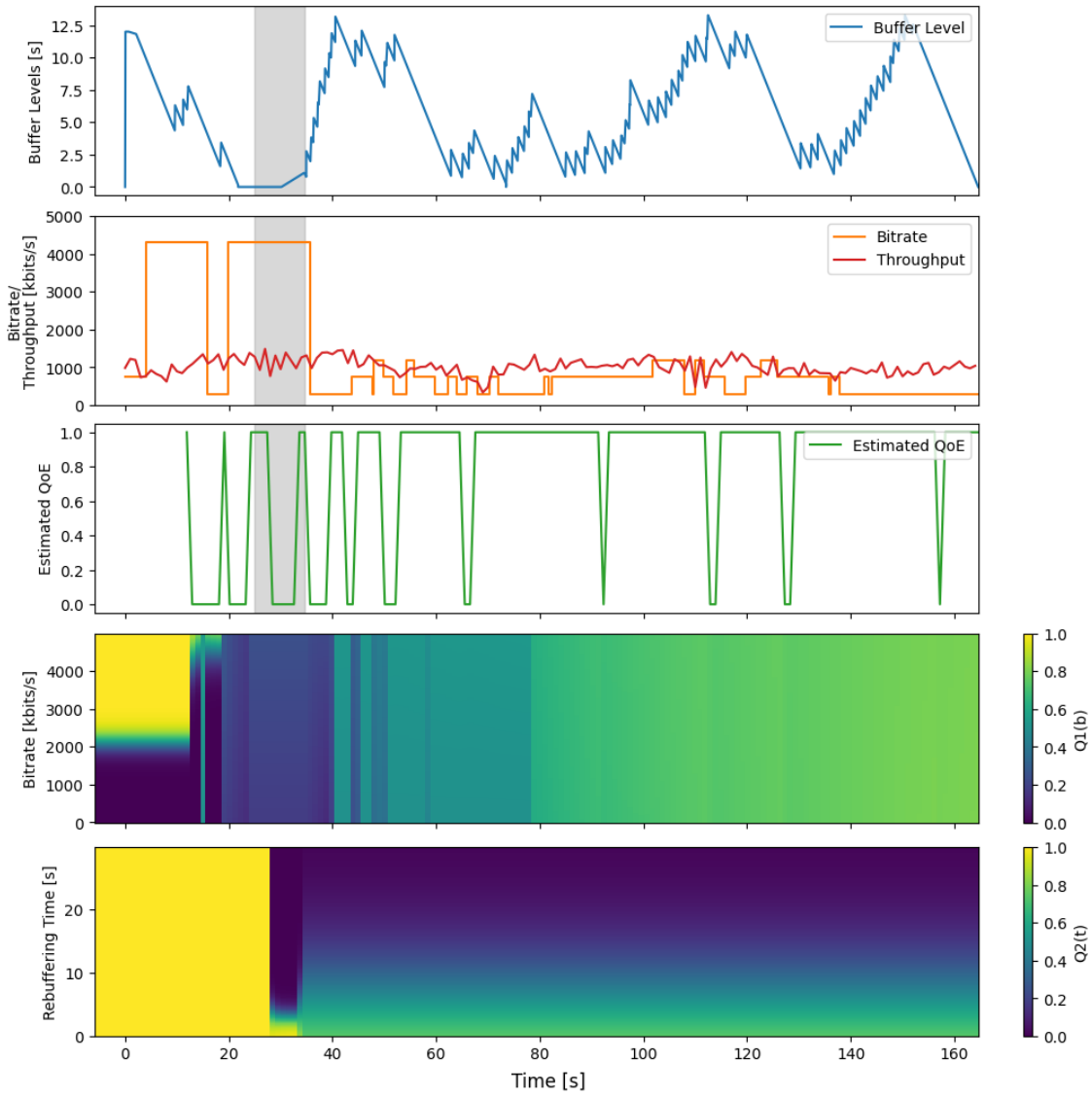

Figure 14: Playback Metrics of Subject 02 (1)

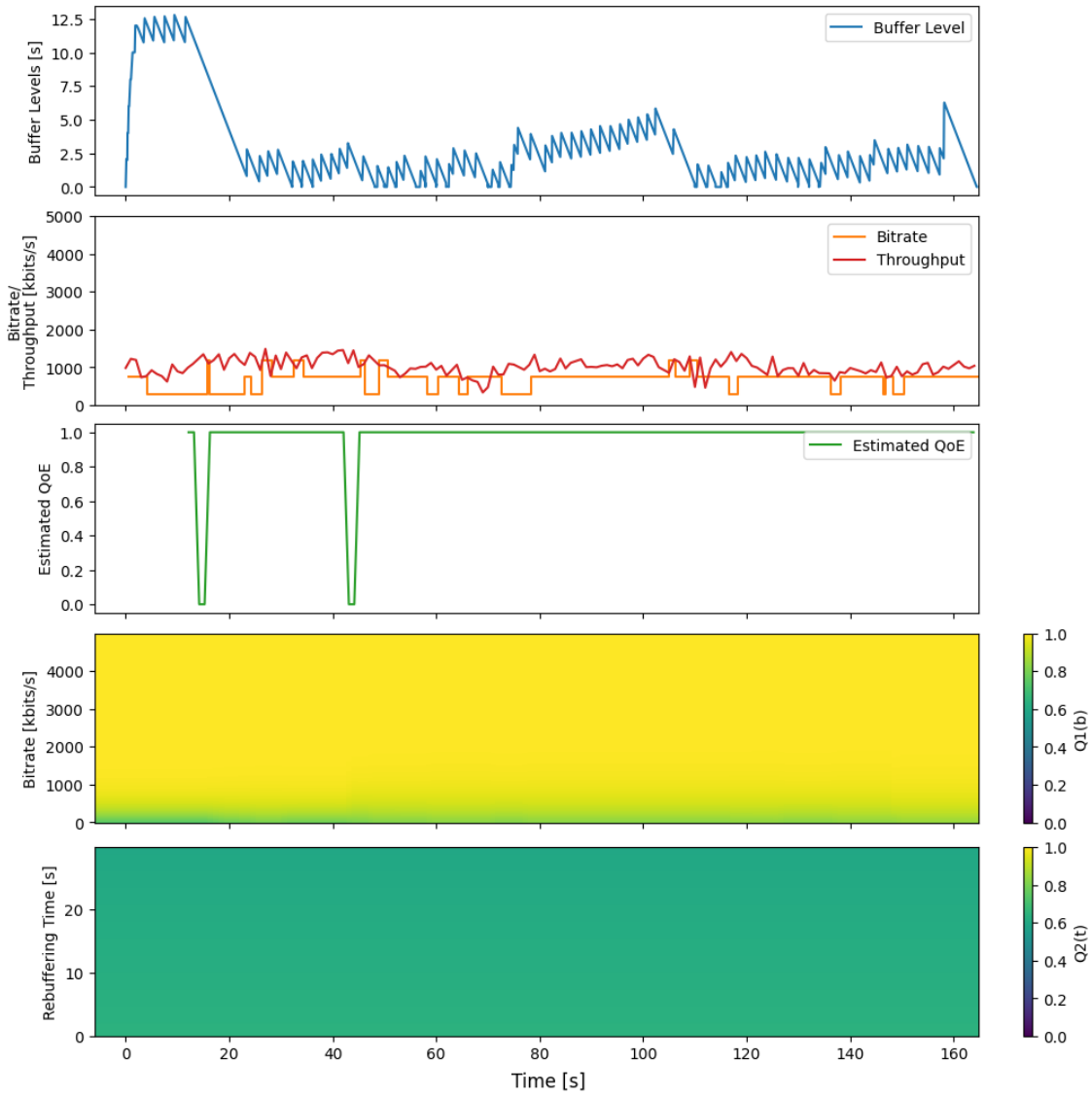

Figure 15: Playback Metrics of Subject 02 (2)

algorithm explained in §4. The color corresponding to QoE is the same as in the $Q_1(R)$ graph. There are gray shaded areas in the graphs, which indicate periods where playback pauses occurred during this time.

Discussion In Fig. 14, a pause occurs around the 20–40 second mark, and during this time, the estimated QoE reaches 0 (dissatisfied), which causes a change in the relationship between pause duration and QoE, represented by $Q_2(x)$. From this point, it can be confirmed that before 40 seconds, a bitrate of 4300 kbps was selected, but afterward, the bitrate is limited to stay below the network throughput.

Next, examining the later playback data after the playback of Fig. 14, shown in Fig. 15, we can see that the bitrate is controlled to stay below the network throughput from the beginning, and since the estimated QoE does not change much, the parameters of the QoE function also remain unchanged.

To summarize, after dissatisfaction with the pause was detected through the estimated QoE, control was adjusted to avoid further pauses, and in later playback, the control was maintained because there was little feedback of dissatisfaction. In other words, it was confirmed that the system implemented in this study was able to perform user-adaptive control during actual viewing by the subjects.

6 Conclusion

In this thesis, we proposed a method for real-time QoE estimation of whether a user is dissatisfied with the current playback quality while watching a video. The proposed method decomposes the user's brainwaves into individual frequency components using complex wavelet transform and calculates the power of specific frequency bands as input. We built a support vector machine model to classify whether the user was dissatisfied or not, and evaluated its performance. Experimental results conducted in our laboratory showed that the estimation accuracy ranged from a minimum of 53% to a maximum of 75%, with an average accuracy of 65%. Additionally, the recall value, which indicates the percentage of QoE degradation that was detected, ranged from a minimum of 17% to a maximum of 77%, with an average of 56%. This suggests that our estimation model was able to detect nearly 60% of QoE degradation on average. Furthermore, we designed an adaptive bitrate control method utilizing this QoE estimation model and implemented it on DASH. Experiments within our research group confirmed that the system was able to adjust playback in response to user feedback.

The estimation method proposed in this study achieved a maximum accuracy of 75% in binary classification. To realize adaptive streaming that better aligns with user preferences, a more accurate estimation method would be necessary. Additionally, since the proposed method produces a binary output, a model capable of providing more gradual estimations would be preferable. Improving these aspects will be a future research topic for our group.

The bitrate decision algorithm for adaptive streaming proposed in this study focuses on bitrate levels and the duration of playback pauses. However, other factors may also impact users' QoE, such as the frequency of playback pauses, etc. Incorporating these elements into the algorithm could enable more user-adaptive control and further improve QoE. Addressing these challenges will also be a future research topic for our group.

Acknowledgments

I would like to express my deepest gratitude to Professor Masayuki Murata, of the Graduate School of Information Science and Technology at Osaka University, for his invaluable guidance and support throughout my research journey. His insightful advice during our weekly meetings was instrumental in shaping this work. Without their continuous mentorship and thoughtful suggestions, this research would not have been possible. His expertise and dedication have greatly contributed to both my academic growth and the success of this project. I am also deeply grateful to Associate Professor Daichi Kominami of the Institute for Open and Transdisciplinary Research Initiatives, Osaka University. He provided me with close and daily research advice. His support was essential in refining my work, and I sincerely appreciate his participation as a subject in my experiments. Without his contributions, my research would not have been completed. Furthermore, I would like to extend my gratitude to Associate Professor Shin'ichi Arakawa of the Graduate School of Information Science and Technology, Osaka University, Associate Professor Yuichi Ohsita of the D3 Center, Osaka University, Assistant Professor Tatsuya Otoshi of the Graduate School of Economics, Osaka University, and Assistant Professor Masaaki Yamauchi of the Graduate School of Information Science and Technology, Osaka University for their valuable advice. Their advice helped define the direction of my research. Their insights and suggestions were crucial in shaping my approach and improving the quality of this work. Finally, I would like to thank my family and my fellow lab members of the Advanced Network Architecture Research Laboratory at the Graduate School of Information Science and Technology, Osaka University. Their support and the comfortable research environment allowed me to continue my studies. Without their encouragement and assistance, this research would not have been able to continue. Once again, I sincerely appreciate the support, guidance, and encouragement from everyone who contributed to this research. Thank you all.

References

[1] Netflix. NETFLIX OPEN CONTENT. https://opencontent.netflix.com. Accessed: 2024-08-16.

[2] Hiiragi Magnetite. Marshall Maximizer. https://www.youtube.com/watch?v=jMKPYg0uhCI. Accessed: 2025-02-03.

[3] ——. Kanon. https://www.youtube.com/watch?v=G07TPX7i56g. Accessed: 2025-02-03.

[4] Yoshida Yasei. Override. https://www.youtube.com/watch?v=LLjfal8jCYI. Accessed: 2025-02-03.

[5] CISCO. 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. Accessed: 2024-12-23.

[6] J. Jiang, V. Sekar, and H. 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*, Dec. 2012, pp. 97–108.

[7] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," *IEEE/ACM transactions on networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[8] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, Aug. 2015, pp. 325–338.

[9] X. Zuo, J. Yang, M. Wang, and Y. Cui, "Adaptive bitrate with user-level QoE preference for video streaming," in *Proceedings of IEEE Conference on Computer Communications*, May 2022, pp. 1279–1288.

[10] K. Kitao, D. Kominami, and M. Murata, "GA-based feature selection for QoE estimation using EEG during video viewing," in *Proceedings of International Conference on Emerging Technologies for Communications*, Dec. 2020.

[11] K. Kitao, "Real-time QoE estimation method using EEG for video delivery services," Master's thesis, Graduate School of Information Science and Technology, Jan. 2021.

[12] X. Tao, Z. Chen, M. Xu, and J. Lu, "Rebuffering optimization for DASH via pricing and EEG-based QoE modeling," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 7, pp. 1549–1565, 2019.

[13] ISO/IEC 23009-1:2022. https://www.iso.org/standard/83314.html. Accessed: 2025-01-31.

[14] H. Nam, K.-H. Kim, and H. Schulzrinne, "QoE matters more than QoS: Why people stop watching cat videos," in *Proceedings of IEEE International Conference on Computer Communications*. IEEE, Apr. 2016, pp. 1–9.

[15] DASH Industry Forum. dash.js. https://dashjs.org/. Accessed: 2024-12-23.

[16] chrome-remote-interface. https://github.com/cyrus-and/chrome-remote-interface. Accessed: 2025-01-30.

[17] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: Analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, Feb. 2013, pp. 114–118.

[18] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with Pensieve," in *Proceedings of the conference of the ACM special interest group on data communication*, Aug. 2017, pp. 197–210.

[19] FFmpeg. https://www.ffmpeg.org/. Accessed: 2025-01-30.

[20] gpac.io - Ultramedia open source infrastructure. https://gpac.io/. Accessed: 2025-01-31.