Optimizing Video Streaming in Dynamic Networks: An Intelligent Adaptive Bitrate Solution Considering Scene Intricacy and Data Budget

Weihe Li, Jiawei Huang, *Member, IEEE,* Yu Liang, Qichen Su, Jingling Liu, Wenjun Lyu, and Jianxin Wang, *Senior Member, IEEE*

Abstract—Adaptive Bitrate (ABR) algorithms have become increasingly important for delivering high-quality video content over fluctuating networks. Considering the complexity of video scenes, video chunks can be separated into two categories: those with intricate scenes and those with simple scenes. In practice, it has been observed that improving the quality of intricate chunks yields more substantial improvements in Quality of Experience (QoE) compared with focusing solely on simple chunks. However, the current ABR schemes either treat all chunks equally or rely on fixed linear-based reward functions, which limits their ability to meet real-world requirements. To tackle these limitations, this paper introduces a novel ABR approach called CAST (Complex-scene Aware bitrate algorithm via Self-play reinforcemenT learning), which considers the scene complexity and formulates the bitrate adaptation task as an explicit objective. Leveraging the power of parallel computing with multiple agents, CAST trains a neural network to achieve superior video playback quality for intricate scenes while minimizing playback freezing time. Moreover, we also introduce a new variant of our proposed approach called CAST-DU, to address the critical issue of efficiently managing users' limited cellular data budgets while ensuring a satisfactory viewing experience. Furthermore, we present CAST-Live, tailored for live streaming scenarios with constrained playback buffers and considerations for energy costs. Extensive trace-driven evaluations and subjective tests demonstrate that CAST, CAST-DU, and CAST-Live outperform existing off-the-shelf schemes, delivering a superior video streaming experience over fluctuating networks while efficiently utilizing data resources. Moreover, CAST-Live demonstrates effectiveness even under limited buffer size constraints while incurring minimal energy costs.

Index Terms—Video streaming, bitrate adaption, data budget, parallel computing, limited buffer, self-play reinforcement learning, scene complexity.

1 Introduction

The proliferation of intelligent mobile devices and the widespread availability of wireless connectivity have resulted in a substantial surge in network traffic attributed to video streaming. As a preferred method for delivering video content over fluctuating networks, HTTP Adaptive Bitrate Streaming, also known as Dynamic Adaptive Streaming over HTTP (DASH) [2], has gained prominence. In the DASH system, video content is pre-encoded and pre-chunked at different quality levels (bitrates) on the server side. On the client side, the player dynamically selects the most appropriate bitrate for each chunk, considering an estimation of network capacity and measured buffer occupancy, with the aim of providing viewers with a high QoE.

Existing DASH system utilizes two primary encoding methods for video content delivery: Constant Bitrate (CBR) and Variable Bitrate (VBR). Under the CBR approach, the entire video is encoded using a fixed bitrate for a given quality level, leading to uniform bit allocation across all video chunks. Consequently, this uniformity may result in inconsistent quality across chunks with different scenes [3]. Conversely, VBR employs a more dynamic allocation of bits, which allocates more bits to intricate scenes characterized by high dynamics while giving fewer bits to low-motion scenes, aiming to achieve consistent quality across chunks with the same bitrate level. The benefits of VBR, including the capacity to attain equivalent quality with a reduced bit budget, have prompted content providers to shift their encoding strategies from CBR to VBR in recent years [4].

Limitations of Prior Arts. (i) In spite of the efforts made by VBR encoding to achieve consistent quality across video chunks with varying scene complexity, the quality of chunks with intricate scenes (referred to herein as "intricate," "complex," or "dynamic" chunks) remains noticeably lower than that of chunks with simple scenes (termed as "simple" chunks) due to limitations inherent in existing encoding techniques, as discussed in Section 3.1. Given the significant role played by intricate chunks in determining viewing quality, enhancing the quality of such chunks can lead to more substantial improvements in QoE [6]. Unfortunately,

Weihe Li is with the School of Informatics, The University of Edinburgh, Edinburgh, Scotland, EH8 9YL, United Kingdom. e-mail: weihe.li@ed.ac.uk.
Jiawei Huang, Qichen Su, Jingling Liu, and Jianxin Wang are with the School of Computer Science and Engineering, Central South University, Changsha, 410083, China. e-mail: jiaweihuang@csu.edu.cn, qichensu.csu@gmail.com, jinglingliu@csu.edu.cn, jxwang@csu.edu.cn.
Yu Liang is with the School of Computing, Lancaster University, Lancaster, LA1 4YW, United Kingdom. e-mail: y.liang20@lancaster.ac.uk
Wenjun Lyu is with the Department of Computer Science, Rutgers University, New Jersey, 08854, United States. e-mail: wenjun.lyu@rutgers.edu
A preliminary version of this paper appeared in International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), 2023 [1].
Jiawei Huang is the corresponding author.

the majority of existing ABR algorithms rarely consider the impact of scene complexity, resulting in subpar quality for intricate chunks and subsequent degradation in QoE. Although some approaches account for scene complexity by aiming to optimize a score using a linear-based formula with fixed weighted sum metrics, accurately calculating the optimization function remains challenging [3], [55]. The design of an appropriate function is critical, as an inadequately formulated one can mislead ABR algorithms to make inappropriate bitrate decisions, ultimately compromising users' viewing experience.

(ii) Moreover, video streaming represents one of the most bandwidth-intensive applications, with even a single hour of High Definition (HD) video consumption significantly impacting a user's monthly data plan [18]. Therefore, it becomes crucial to optimize mobile video streaming to strike a balance between reducing data usage and minimizing its potential impact on users' QoE. This optimization enables users to enjoy good-quality content within the confines of their specific monthly data budgets. Moreover, the reduction in data usage per streaming session not only benefits individual users but also has broader implications for the overall network performance. By alleviating the load on cellular networks, this optimization can lead to an enhanced QoE for all users who share the same network infrastructure. Nevertheless, it is noteworthy that existing ABR approaches primarily concentrate on maximizing video quality within the constraints of network bandwidth, rather than actively limiting data usage. As such, there is a pressing need for novel ABR strategies that explicitly address data usage optimization in video streaming.

(iii) Live streaming has become a significant part of society recently, covering live sports events, virtual concerts, real-time tutorials, interactive product launches, and more. To guarantee an outstanding viewing experience with improved interactivity, content providers strive to achieve low end-to-end latency, along with high QoE, encompassing high playback quality and low freezing time. To achieve low latency, the player is equipped with a small buffer capacity, approximately 6 seconds [60], distinguishing it from traditional Video on Demand (VoD) streaming, where the buffer size is typically on the minute level [14]. Therefore, maintaining a low rebuffering time under a limited buffer size is more challenging, as a larger buffer size is better suited to absorb network variations and chunk variations.

Our Proposed Solution and Contributions. Motivated by the aforementioned challenges, this paper aims to leverage parallel training to introduce a novel ABR algorithm. Machine learning is chosen as the methodology due to the ability of learning-based schemes to derive bitrate selection strategies by interacting with the environment without being constrained by any specific assumptions, such as network capacity stability over a short time frame [3], [6]. Thus, learning-based approach typically leads to superior performance compared to traditional heuristic-based methods in most cases [11]. In practical scenarios, the task of bitrate adaptation can be perceived as a straightforward objective or rule. For instance, the primary goal of most ABR schemes is to minimize rebuffering time while ensuring a high playback quality [45]. In this paper, we propose an approach that utilizes multi-agent self-play reinforcement learning to parallelly train the neural network. For CAST, the objective is to deliver high-quality video chunks with intricate scenes without excessively compromising the quality of simple chunks and minimizing rebuffering time on networks with variable conditions. To address the challenge of explicitly constraining the amount of data used during a streaming session without significantly compromising the viewing experience, we further propose a variant called CAST-DU. Given the rising popularity of live streaming, we acknowledge the necessity of maintaining robust ABR algorithm performance even with constrained buffer sizes. Moreover, we tackle the energy consumption related to download and playback operations by introducing an energy-conscious variant, CAST-Live. By training via distinct actual goals, CAST, CAST-DU, and CAST-Live can accurately fulfill specific requirements. In summary, our contributions in this research can be outlined as follows:

(i) *Integration of VBR Features*: Our approach integrates VBR features, considering that different chunks have varying sizes based on scene complexity due to VBR encoding. While some works have recognized the importance of intricate chunks, such as [3], [55], they either select bitrates using a heuristic approach relying on naive assumptions like the network bandwidth remaining stable in the short term or learn the selection policy using a fixed linear reward function, posing challenges in reflecting user requirements in practice. In contrast to existing works that consider intricate chunks, we employ self-play reinforcement learning to train the bitrate selection policy, learning from explicit video streaming requirements. This approach better reflects user demand in practice, resulting in improved performance across various metrics.

(ii) *Data Budget Consideration*: Unlike many existing methods that focus on maximizing overall QoE without considering data usage, we prioritize the user's data budget during video content downloading. Acknowledging the substantial impact of video consumption on a user's monthly data plan, we carefully restrict data usage below the user's designated budget while ensuring a commendable viewing experience. Importantly, CAST-DU can be easily built upon the existing CAST approach, providing an effective solution to manage data usage during video streaming sessions.

(iii) *Adaptability to Live Streaming*: Recognizing the rising prevalence of live streaming, our approach places importance on ensuring effective ABR algorithm performance even under limited buffer sizes, such as 6 seconds, to guarantee low latency. Additionally, we address the energy costs associated with the download and playback processes by introducing an energy-aware CAST-Live.

To assess the performance of CAST, CAST-DU, and CAST-Live, we conduct an extensive trace-driven evaluation encompassing various network environments. The experimental findings illustrate the remarkable improvements achieved by CAST in enhancing the quality of intricate chunks and minimizing playback freezing time. Regarding CAST-DU, we observe that our variant significantly reduces the disparity between the actual data usage and the target usage across various traces, while concurrently providing a satisfactory viewing experience. For CAST-Live, our method excels in both operating under tight buffer

constraints and reducing energy costs when compared to existing approaches. To validate the practical effectiveness of our proposed method, we perform a subjective assessment by recruiting 20 volunteers to watch an online video using different ABR algorithms. The results of this experiment show that 19 out of the 20 participants affirm that CAST provides them with good quality for intricate chunks while avoiding extended playback interruption time. This positive feedback underscores the efficacy of our approach in delivering an enhanced viewing experience for users.

2 Related Work

2.1 Heuristic-based

The classic heuristic-based approaches include FESTIVE [35], which estimates available bandwidth based on past throughput and selects the highest bitrate not exceeding the estimated capacity. BBA [36] employs a function to map buffer occupancy to available video bitrate. On the other hand, BOLA [37] is a more sophisticated buffer-based scheme that optimizes an objective function using the Lyapunov optimization formulation. MPC [6] maximizes an optimization problem over a horizon of several chunks ahead by combining the signal of rate and buffer. QUETRA [7] transforms the video streaming task into a queuing model and utilizes queuing theory along with bandwidth prediction to select the bitrate for each chunk, maintaining buffer occupancy fluctuating around half of the maximum buffer size. CAVA [3] considers scene complexity and obtains the playback bitrate by optimizing a predefined function based on a set of assumptions. PIA [8] strategically harnesses Proportional-Integral-Derivative (PID) control concepts and incorporates novel strategies for various requirements of ABR streaming to improve overall QoE. MSPC [9] leverages the Kalman filter to predict network bandwidth and adopts multi-step prediction to provide responsive adaptation and smooth playback for mobile video applications.

Limitations: Although heuristic methods have demonstrated efficacy in some cases, they do come with certain limitations. One significant drawback is that these methods often require careful parameter tuning, and they heavily rely on some naive assumptions, such as negligible bandwidth variations over short periods. Consequently, their performance might be inadequate in fluctuating network conditions, particularly when dealing with highly variable cellular networks [11].

2.2 Learning-based

Due to the challenge of tuning parameters for heuristic algorithms, considerable efforts have been directed towards enhancing the performance of ABR schemes using learning-based approaches. For instance, Pensieve [10] utilizes Deep Reinforcement Learning (DRL) to train a neural network for bitrate adaptation, maximizing a linear QoE function. Stick [20] employs DRL to determine optimal parameter settings for BBA under various network conditions, thereby maximizing QoE. BayesMPC [12] leverages Bayesian neural network models to improve bandwidth predictions and employs MPC for bitrate selection in upcoming video segments. Fugu [13] employs online learning techniques to estimate download times for video chunks at each level and applies MPC to select the optimal bitrate for subsequent chunks. Comyco [11] and DAVS [5] employ imitation learning for attaining the bitrate selection for each video chunk. RAV [21] and SPA [22], [23] respectively utilize deep reinforcement learning and self-play reinforcement learning techniques to acquire bitrate selection policies for both audio and video content, effectively reducing the bitrate gap between audio and video chunks with the same index. PRIOR [14] introduces an accurate network bandwidth prediction method with an attention mechanism and uses DRL to learn a bitrate selection policy, maximizing QoE. Zwei [16] employs self-play reinforcement learning to develop a bitrate selection policy based on a pre-defined rule.

Limitations: However, these approaches have limitations as they often overlook the differences in scenes between diverse chunks, which can result in inferior quality for intricate chunks. Furthermore, other learning-based methods, such as [11], [26], [27], also suffer from the same drawback, lacking the consideration of scene differences and appropriate methods for scene-aware bitrate adaptation.

2.3 Data-usage-aware

The majority of existing ABR approaches primarily prioritize optimizing video quality under network bandwidth constraints, with limited consideration for the user's data budget. QUAD [4], in contrast, assumes a user-specified target quality and aims to avoid segments whose qualities exceed the target quality, resulting in data savings.

Limitations: However, some of these capped approaches, including [4], [18], may be overly conservative, leading to significant data discrepancy.

2.4 Data-wastage-aware

Approaches such as [39], [40] focus on mitigating data wastage by adjusting the buffer size, but their emphasis differs from ours. DeepBuffer [39] and PSWA [40] primarily concentrate on adjusting bitrate and playback buffer size to mitigate data wastage resulting from various user-triggered behaviors such as early departure and video skip. In contrast, our work, CAST-DU, specifically aims to align the data usage for downloading video content explicitly with the allocated data budget.

2.5 Live Streaming

Recognizing the growing demand for live streaming, several bitrate adaptation algorithms have been developed [24], [25], [60]. Cratus [60] aims to regulate buffer dynamics by designing an ideal buffer model to achieve target levels through heuristic methods. Fleet [25] comprises four main modules: bandwidth measurement, throughput prediction, bitrate adaptation, and latency management, with a stochastic MPC controller at the heart of its bitrate adaptation module. However, their heuristic designs may lead to inferior robustness due to reliance on naive assumptions.

2.6 Summary

In summary, existing methods often fail to account for variations in scene complexity, data budget constraints, and the challenges of operating effectively under tight buffer sizes, leading to suboptimal performance such as compromised

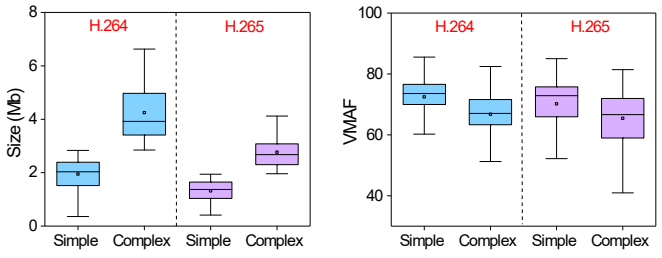
quality for intricate chunks, inappropriate data usage, or high rebuffering times. While some recent approaches have addressed scene complexity [3], [55], they either rely on meticulous parameter tuning and network bandwidth assumptions or utilize fixed-weighted linear reward functions for bitrate selection, resulting in reduced robustness. Given these limitations, there is a pressing need for a novel and robust approach capable of effectively managing fluctuations in scene complexity, data budget constraints, and live streaming scenarios.

3 Motivation

3.1 Scene Complexity

The content of video chunks can vary in terms of scene complexity, with some chunks featuring low-motion and straightforward scenes, while others present high-motion and intricate scenes. In order to investigate the impact of chunk size on quality, we conducted experiments using the video "Big Buck Bunny" [28] as our test case. The video was encoded into six different tracks, ranging from 144p to 1080p, using two different encoding schemes: H.264 and H.265. Each track was then divided into chunks with a fixed duration of 2 seconds.

When assessing the scene complexity, several metrics such as Spatial Information (SI) and Temporal Information (TI) [30] may be utilized. Nevertheless, implementing these metrics in commercial streaming services is challenging due to the computationally intensive content-level analysis they require, which would necessitate significant modifications to the streaming pipeline [3]. Fortunately, in VBR encoding, the size of a chunk can effectively indicate relative scene complexity, with intricate chunks allocated more bits than simple ones. Given that scene complexity is constant across quality levels at a given playback point, we designate a middle-quality track (480p) as a reference track and classify chunks based on their size, with larger chunks in the first half classified as intricate and the remaining chunks classified as simple [5]. It is important to note that the classification method is not fixed, and other strategies, such as using four classes, can be employed. In this study, we utilize the Video Multi-method Assessment Fusion (VMAF) metric [38] to evaluate the quality of each chunk, as it is known to accurately reflect users' subjective viewing experience [11], [15], [31], [33].



(a) Chunk size. (b) Chunk quality.

Fig. 1: Analysis of chunk size and quality. (a) Comparison of the size of simple and intricate chunks. (b) Comparison of the quality of simple and intricate chunks.

As depicted in Figure 1(a), we can observe that intricate chunks are considerably larger than simple ones, with an average size of around 1.18× and 1.10× higher under the H.264 and H.265 codecs, respectively. However, despite utilizing more bits for encoding, the quality of intricate chunks is still inferior to that of simple ones, as shown in Figure 1(b). Specifically, under the H.264 and H.265 formats, the average quality of intricate chunks is respectively 7.88% and 6.73% lower than that of simple chunks. These findings have prompted us to consider different treatment strategies for intricate and simple chunks to achieve optimal viewing experience for users. It is critical to allocate bandwidth to prioritize the transmission of intricate chunks while ensuring that the quality of simple chunks is not significantly compromised. Additionally, minimizing stall time is of utmost importance, as this can significantly impact the overall viewing experience [23].

3.2 Data Usage

To investigate the data usage of different ABR algorithms across various network scenarios, we conduct experiments using the "Envivo-Dash3" video [5]. This video is encoded with six distinct bitrates: {300, 750, 1200, 1850, 2850, 4300}Kbps, and each chunk has a duration of 4 seconds. For our evaluation, we select MPC [6] and Zwei [16] as the ABR algorithms and employ 142 HSDPA [51] and 200 FCC [52] traces to represent different network conditions.

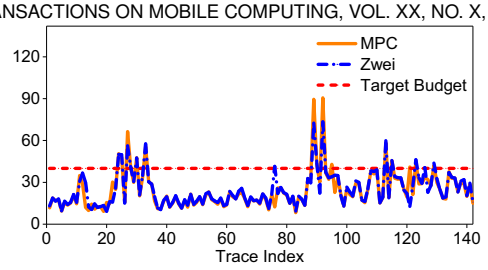
For the data budget setting, we designate a reference track at 1850Kbps from the test video. Subsequently, the data budget scale factor γ is established at 1.05. This indicates that the data budget is calculated as 1.05 times the cumulative size of all chunks within the reference track. As suggested by [18], users have the flexibility to adjust the data budget scale factor within a broad range, such as from 1.0 to 1.5. This adjustment allows exploration of different levels of stringency in the data budget relative to the size of the reference track. Accordingly, the final data budget is determined as 40MB.

Figure 2 presents the data usage of downloading the video over various network traces, indicating substantial variations in the data usage of different ABR algorithms across diverse network scenarios. For example, under the HSDPA and FCC traces, the data usage of MPC ranges from 8.31MB to 90.53MB and 6.69MB to 70.53MB, respectively. During low network rates, these algorithms tend to select lower bitrates to avoid playback interruptions. However, when the network capacity is high, these algorithms become more aggressive in selecting high bitrates for each chunk, without adequately considering the users' data budget.

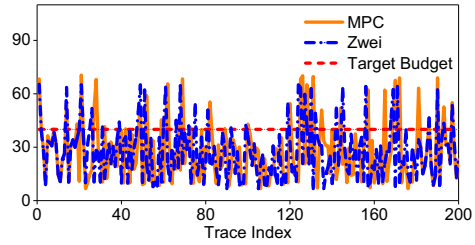
TABLE 1: Percentage of network traces exceeding the target data budget.

Percentage	MPC	Zwei
HSDPA	10.56%	10.56%
FCC	17%	21.82%

In Table 1, we list the proportion of traces that exceed the data budget 40MB. For instance, using MPC, we observe that 10.56% and 17% of traces exceed the target data budget over the HSDPA and FCC traces, respectively. These results underscore the fact that existing ABR approaches

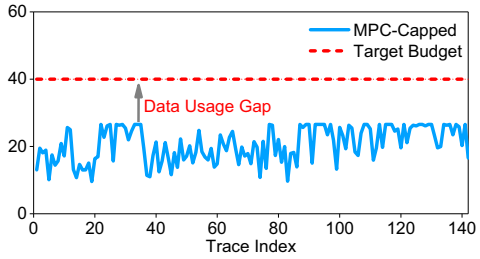


(a) HSDPA.

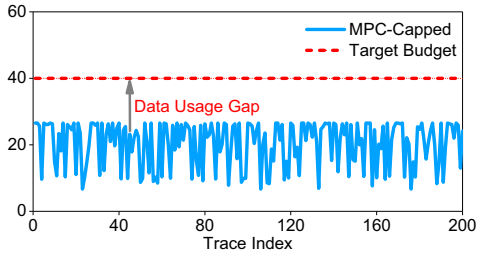


(b) FCC.

Fig. 2: Data usage (MB) over different network traces.



(a) HSDPA.



(b) FCC.

Fig. 3: Data usage (MB) with capped MPC over different network traces.

lack control over data usage when selecting the bitrate for each chunk.

While there are data-capped methods that limit the maximum selected bitrate to ensure the sum of all chunk sizes at that bitrate level does not exceed the data budget [4], [46], this strategy often results in significant data usage gaps. Specifically, when the cumulative size of all chunks in the i-th bitrate level does not exceed the predetermined data budget N, and the $(i+1)$-th bitrate level surpasses N, the upper capped bitrate is set as the i-th bitrate level. For example, when configuring the target budget to 40MB and implementing the data-capped method on MPC, as depicted in Figure 3, a substantial data gap emerges. This gap is evident in the disparity between the total downloaded size of the video on each trace (indicated by the blue line) and the designated target budget. Such a discrepancy results in inefficient utilization of network resources and may lead to potential degradation of playback quality. Furthermore, the fixed target bitrate fails to fully consider scene complexity in different chunks, potentially resulting in suboptimal playback quality for segments with intricate scenes.

3.3 Summary

Based on the above analysis, we draw the following conclusions: (i) existing ABR algorithms mainly treat all chunks indiscriminately, resulting in inferior quality for intricate chunks due to their larger size and lower VMAF; (ii) recent bitrate adaptation solutions rarely consider the data budget when selecting the bitrate for each chunk, leading to excessive data usage. The naive data-capped method also incurs significant data discrepancy. An alternative strategy involves dynamically adjusting the target bitrate by recalculating it each time a video chunk selects a playback bitrate based on the remaining data budget for subsequent chunks. This continuous adjustment results in a new target bitrate, potentially differing from the previous one. However, this method heavily depends on making real-time adjustments for each individual chunk, which could lead to challenges in swiftly adapting to sudden changes in network conditions or scene complexity [18]. Additionally, ABR schemes face challenges in delivering good performance under constrained buffer sizes in live streaming scenarios [60]. These limitations of current ABR approaches inspire the design of a new algorithm that optimizes the video streaming experience by considering both chunk complexity and data budget, while also performing effectively under tight buffer constraints.

4 Design

4.1 CAST Design

4.1.1 The Selection of the Learning Method

This paper introduces a novel approach, CAST, which employs self-play reinforcement learning to learn a bitrate selection policy that takes scene complexity into account. Unlike the widely adopted deep reinforcement learning (DRL) that trains using a linear reward function, CAST formulates the ABR problem as an explicit objective and treats the learning task as a competition between different trajectories collected by itself. By training the neural network to approach the predefined goal's gradient, the converged policy can effectively meet the actual demands [47]. The decision to employ self-play reinforcement learning is grounded in its proven effectiveness when compared to existing learning methods for various tasks, including video streaming, network congestion control, and games, as demonstrated in [16], [56], [57]. Although some related works in video streaming such as [16], [23] also employ self-reinforcement learning, our approach differentiates itself in several ways. In contrast to [16], our method considers a wider array of crucial features essential to video streaming, including scene complexity, data usage, energy cost, and constrained buffer size. Additionally, we conduct an extensive evaluation in Section 5 to effectively demonstrate the superiority of our method over [16]. On the other hand, literature [23] focuses on audio bitrate adaptation, considering video streaming from a different perspective, and thus we do not include a comparison with it.

4.1.2 Training Process

The training process begins by sampling D distinct trajectories denoted as $T_d = \{s_0^d, a_0^d, s_1^d, a_1^d, \cdots, s_t^d, a_t^d\}$ from the same environment and starting point using a deep neural network. These trajectories are recorded in a collection M, where D is the total number of trajectories collected, $d \in D$, s_t represents the environment status at the t-th time, and a_t denotes the action for video bitrate selection at the t-th time. Further details about the status and action can be found in Section 4.4.

As shown in Algorithm 1, CAST evaluates the performance of the current policy by comparing each trajectory's performance. For any two different samples T_p and T_q from the sample set M, CAST first computes their average quality for intricate chunks C_p and C_q, average rebuffering time B_p and B_q, and average quality for simple chunks S_p and S_q (Line 2). The comparison process follows a deterministic rule with two cases.

Case 1: Given three thresholds α, β, and η, if the absolute differences $|C_p - C_q| < \alpha$, $|B_p - B_q| < \beta$, and $|S_p - S_q| < \eta$, the competition between T_p and T_q is considered a *draw* (Lines 4-6).

Case 2: Otherwise, the competition process is based on a priority (Lines 7-14). First, CAST compares the average rebuffering time between T_p and T_q (Line 7). If the absolute difference $|B_p - B_q|$ exceeds the threshold β, the sample with a shorter rebuffering time is the winner (Line 13), and the competition ends. If not, the competition continues by comparing the absolute difference in intricate chunks' quality (Line 8), followed by comparing the absolute difference in simple chunks' quality (Line 9). The thresholds α, β, and η are set as 1, 0.01, and 1, respectively.

After completing the competitions, each trajectory receives a win count A_i based on how many times it won in all sampled trajectories D. Consequently, the win rate of each sample is $\frac{A_i}{|D|-1}$, as there is no battle between a trajectory and itself.

The final step of CAST involves training a neural network using the collected trajectory samples and their corresponding win rates. To achieve this, the neural network is trained using the state-of-the-art Dual-clip Proxy Policy Optimization (Dual-PPO) algorithm [48], known for its improved stability and convergence compared with existing learning methods, including the original PPO algorithm. The neural network comprises a policy network and a value network. During training, the policy network is optimized to increase the probability of winning samples and decrease the probability of losing samples using a specific loss function. The loss function for the policy network is expressed as follows.

$$\mathcal{L} = \begin{cases} \hat{\mathbb{E}}_t[\max(\min(p_t(\theta)\hat{A}_t, clip(p_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t), c\hat{A}_t)], \hat{A}_t < 0, \\ \hat{\mathbb{E}}_t[\min(p_t(\theta)\hat{A}_t, clip(p_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)], \hat{A}_t \geq 0, \end{cases}$$

The loss function of the policy network incorporates the advantage function $\hat{A}t$ and the policy ratio $p_t(\theta)$, which represents the ratio between the current policy π_θ and the old policy $\pi_{\theta_{old}}$. The hyper-parameters ε and c are assigned the same values as in [48]. Conversely, the value network is trained to minimize the estimated error of the advantage function, denoted as $\mathcal{L}^V = \frac{1}{2}\hat{\mathbb{E}}_t[\hat{A}_t]^2$. Additionally, an entropy term is introduced to the loss function to encourage exploration. Therefore, the overall loss function is defined as follows:

$$\nabla\mathcal{L}^{CAST} = -\nabla_\theta\mathcal{L} + \nabla_{\theta_p}\mathcal{L}^V + \nabla_\theta\gamma H^{\pi_\theta}(s_t).$$

where γ denotes the entropy weight.

Algorithm 1 Battling Rule for CAST

Input: Agent results for two agents.
Output: Battling outcome f.

1: **function** CAST RULE(agent_results)
2: Compute the average quality for complex chunks, represented by C_p and C_q, as well as the average rebuffering time denoted by B_p and B_q. Additionally, calculate the average quality for simple chunks, denoted as S_p and S_q, all based on trajectories T_p and T_q.
3: $f \leftarrow [-1, -1]$;
4: **if** $|B_p - B_q| < \beta$ and $|C_p - C_q| < \alpha$ and $|S_p - S_q| < \eta$ **then**
5: $f \leftarrow [0, 0]$; ▷ a tie competition.
6: **return** f
7: **if** $|B_p - B_q| < \beta$ **then**
8: **if** $|C_p - C_q| < \alpha$ **then**
9: $f[\text{argmax}([S_p, S_q])] \leftarrow 1$;
10: **else**
11: $f[\text{argmax}([C_p, C_q])] \leftarrow 1$;
12: **else**
13: $f[\text{argmin}([B_p, B_q])] \leftarrow 1$;
14: **return** f

4.2 CAST-DU Design

To ensure that each viewing session adheres to the target data budget, we introduce a new variant called CAST-DU, which builds upon the original CAST algorithm. In contrast to the basic CAST, CAST-DU incorporates an additional metric, data usage, into the evaluation process of each trajectory, denoted as U_p and U_q, respectively. The battling rule of two trajectories T_p and T_q is shown in Algorithm 2.

Like CAST, the process commences by computing various metrics from the collected trajectories T_p and T_q, including the average quality of intricate and simple chunks, the average rebuffering time, and the total data usage (Line 2). We initialize the target data budget N (Line 3). To set the data budget for a video, we use a relative approach with respect to a reference track, typically the middle track [18]. The data budget scale factor, denoted as γ, is subject to flexible adjustments over a broad range of 0.5 to 1.5. This approach allows for the exploration of different levels of data budget constraints in relation to the cumulative size of all chunks within the reference track. Subsequently, employing four thresholds α, β, η, and κ, we determine whether the absolute differences in the average rebuffering time, quality of intricate/simple chunks, and the deviation of actual data usage from the target budget are smaller than their respective thresholds. If this condition is met, we consider the competition between T_p and T_q as a draw game (Lines 5-6). In this case, the thresholds α, β, η, and κ are configured to 1, 0.01, 1, and 20, respectively.

Conversely, we approach the competition with a prioritized strategy (Lines 8-18). Firstly, we prioritize the rebuffering time as the top consideration since it holds the utmost importance for viewers when they watch a video in practice [45]. Subsequently, given that the data budget is often linked to users' monthly data plans and associated with monetary costs [34], we establish this metric as the second priority. The next priority is to maximize the quality of intricate chunks, and finally, we concentrate on improving the quality of simple chunks. To begin the competition, we compare the average rebuffering time (Line 8). If the absolute difference in rebuffering time is smaller than β, we proceed to compare the deviation of actual data usage from the target budget N. However, suppose the absolute difference in rebuffering time is no smaller than β. In that case, we identify the trajectory with the shorter rebuffering time as the winner and assign a value of 1 to that trajectory (Line 17). The subsequent stages of the competition follow the same logical approach. Ultimately, we determine the winning trajectory and update the neural network by optimizing for the gradient that increases the likelihood of winning samples, in line with the fundamental principle of CAST. By employing this approach, we can effectively achieve the goal of reducing rebuffering time and maintaining data usage within a range close to the target budget.

Algorithm 2 Battling Rule for CAST-DU

Input: Agent results for two agents.
Output: Battling outcome f.

1: **function** CAST-DU RULE(agent_results)
2: Calculate the average quality for intricate chunks, represented by C_p and C_q, as well as the average rebuffering time denoted by B_p and B_q. Additionally, compute the average quality for simple chunks, denoted as S_p and S_q, along with the sum of data usage denoted by U_p and U_q, all based on trajectories T_p and T_q;
3: $N \leftarrow$ Target Budget; ▷ a self-defined value.
4: $f \leftarrow [-1, -1]$;
5: **if** $|B_p - B_q| < \beta$ and $|C_p - C_q| < \alpha$ and $|S_p - S_q| < \eta$ and $||U_p - N| - |U_q - N|| < \kappa$ **then**
6: $f \leftarrow [0, 0]$; ▷ a draw competition.
7: **return** f
8: **if** $|B_p - B_q| < \beta$ **then**
9: **if** $||U_p - N| - |U_q - N|| < \kappa$ **then**
10: **if** $|C_p - C_q| < \alpha$ **then**
11: $f[\text{argmax}([S_p, S_q])] \leftarrow 1$;
12: **else**
13: $f[\text{argmax}([C_p, C_q])] \leftarrow 1$;
14: **else**
15: $f[\text{argmin}([|U_p - N|, |U_q - N|])] \leftarrow 1$;
16: **else**
17: $f[\text{argmin}([B_p, B_q])] \leftarrow 1$;
18: **return** f

4.3 CAST-Live Design

To capture content complexity for live streaming, we can leverage a metadata field on the server side [41]. The metadata is generated during the content preparation stage as part of the encoding/packaging process, incorporating a single numeric value representing the event intricacy for each chunk. This intricacy serves as a measure of the chunk's content-wise importance, with values ranging from 0 to 1. A value of 0 indicates that the chunk is not important (featuring simple scenes), while a value of 1 indicates that the chunk is important (with intricate scenes). Once the server completes the encoding/packaging process for this chunk, it can transmit the corresponding metadata to the client first. Since the metadata is typically small in size, we can transmit it without consuming a significant amount of bandwidth. We can determine whether a chunk is an intricate one by checking its metadata. If the event intricacy exceeds a threshold, we can deem this chunk as an intricate one.

Furthermore, given the pressing environmental concerns highlighted by the climate crisis, there is an urgent need for research to address the energy costs associated with video streaming and promote eco-friendly practices. The energy cost measurement method we employ is not limited to live streaming tasks but can also be adapted for other variants. Here, we specifically focus on live streaming as a representative instance. In practice, energy consumption primarily consists of mobile data transmission and screen display cost [42]. In accordance with [43], the model for data download energy based on throughput can be accurately defined as $D_i = (\frac{\omega}{rate} + \delta) * S_i$. In this equation, E_i denotes the energy consumption for downloading the i-th chunk, $rate$ represents the average rate during the download of the i-th chunk, and S_i signifies the total size (in bits) of the i-th chunk. The recommended values for the constants ω and δ are 210 and 28, respectively [43]. Then we take into account the energy cost associated with screen display. Generally, the display cost is intricately tied to screen brightness, wherein higher brightness levels result in increased energy consumption. Additionally, longer rebuffering times contribute to increased display costs. For example, let's use the Motorola Moto G5 phone as an instance. When the screen brightness is set to 50% or 80% of the full brightness, the associated power consumption values are 573 and 858 mW, respectively [44]. Assuming the screen brightness is adjusted to 50%, the display cost Y_i for the i-th chunk is computed as 573 times the total playback time, considering both the playing time and the rebuffering time during the download of the i-th chunk. Therefore, we estimate the energy cost E_i associated with downloading and playing the i-th chunk during the live streaming session as the sum of D_i and Y_i. Other factors, such as speaker operations, also contribute to the overall energy cost. However, we have omitted consideration of these factors in the current analysis. A more detailed energy estimation will be part of our future work.

Note that we employ a straightforward linear-based calculation method for display energy cost in current study, primarily focusing on screen brightness and display duration. While our initial methodology lays a foundation for understanding energy costs, we acknowledge the necessity of incorporating more sophisticated display energy models in future research endeavors. A promising direction for improvement entails integrating models like Active Matrix OLED (AMOLED), which consider the nuanced luminance efficiencies of red, green, and blue (RGB) pixel elements [64].

The separation of display energy costs within our current paper aims to provide a more nuanced depiction and evaluation. We recognize that energy consumption may not solely be influenced by rebuffering events but can also be affected by various factors, including device-specific characteristics. Hence, the choice to separate display energy costs lays the groundwork for future enhancements in our methodology.

To optimize CAST performance in a live streaming scenario, it is essential to control buffer occupancy within a specific range, preventing playback stalls while maintaining low latency. Additionally, our objective includes delivering superior playback quality for chunks with high complexity and effectively constraining data usage under data limitations. To achieve this, we establish the priorities for CAST-Live as outlined in Algorithm 3. Our primary emphasis is on minimizing rebuffering time, given its significant impact on the user's viewing experience. Following this, we prioritize constraining data usage, as it is directly associated with user costs. Subsequently, our focus shifts to enhancing playback quality for intricate chunks. Additionally, we concentrate on reducing energy costs and maximizing the playback quality for simple chunks. Finally, we address controlling buffer fluctuations around a target buffer length to prevent prolonged latency. Note that the priority setting is not fixed; users can adjust the priority based on their preferences.

Throughout the training process, we use the following parameter values: β is set to 0.01, α, ϵ, and η are set to 1, while ϑ and κ are both set to 20. The selection of parameter values, such as κ, is flexible, and assigning values like 1 will not substantially affect the performance [19].

4.4 Neural Network Structure

We will now delve into the neural network in greater detail, focusing on its inputs, outputs, and network architecture.

Inputs. CAST takes inputs that fall into three distinct categories:

Player environment features, denoted as $P = \left(\vec{c_k}, \vec{b_k}, e_t\right)$, include essential information about the player's environment. Here, $\vec{c_k}$ represents the network capacity for downloading the last k chunks, $\vec{b_k}$ denotes the buffer occupancy of the last k chunks, and e_t indicates the quality of the last chunk. To maintain consistency with previous research [10], the value of k is set to 8.

Intricate chunk features, denoted as $C = \left(\vec{Q_t}, \vec{G_t}, I_t, R_t\right)$, provide relevant details about the intricate chunks. Specifically, $\vec{Q_t}$ represents the quality of each track for the next intricate chunk, $\vec{G_t}$ is the size of each track for the next intricate chunk, I_t is the index of the next intricate chunk, and R_t indicates the number of unretrieved intricate chunks in the video.

Simple chunk features, denoted as $S = (\vec{q_t}, \vec{g_t}, r_t)$, encompass pertinent information about the simple chunks. Here, $\vec{q_t}$ denotes the quality of each track for the next simple chunk, $\vec{g_t}$ represents the size of each track for the next simple chunk, and r_t indicates the number of unretrieved simple chunks in the video.

It is crucial to note that the classification of intricate and simple chunks is based on their size distribution, and further details regarding the classification methodology can be found in Section 5.1.

Algorithm 3 Battling Rule for CAST-Live

Input: Agent results for two agents.
Output: Battling outcome denoted as f.

function CAST-LIVE RULE(agent_results)
  Calculate the average quality for intricate chunks, denoted as C_p and C_q, along with the average rebuffering time represented by B_p and B_q, and the average buffer utilization denoted by M_p and M_q. Additionally, determine the average quality for simple chunks, expressed as S_p and S_q, the cumulative data usage denoted by U_p and U_q, and the cumulative energy cost denoted by E_p and E_q, all based on trajectories T_p and T_q;
  $N \leftarrow$ Target Budget; ▷ An adjustable parameter.
  $F \leftarrow$ Target Buffer Length; ▷ An adjustable parameter.
  $f \leftarrow [-1, -1]$;
  if $|B_p - B_q| < \beta$ and $|C_p - C_q| < \alpha$ and $||M_p - F| - |M_q - F|| < \epsilon$ and $|S_p - S_q| < \eta$ and $|E_p - E_q| < \vartheta$ and $||U_p - N| - |U_q - N|| < \kappa$ **then**
    $f \leftarrow [0, 0]$;
    return f
  if $|B_p - B_q| < \beta$ **then**
    if $||U_p - N| - |U_q - N|| < \kappa$ **then**
      if $|C_p - C_q| < \alpha$ **then**
        if $|E_p - E_q| < \vartheta$ **then**
          if $|S_p - S_q| < \epsilon$ **then**
            $f[\text{argmin}([|M_p - F|, |M_q - F|])] \leftarrow 1$;
          else
            $f[\text{argmax}([S_p, S_q])] \leftarrow 1$;
        else
          $f[\text{argmin}([E_p, E_q])] \leftarrow 1$;
      else
        $f[\text{argmax}([C_p, C_q])] \leftarrow 1$;
    else
      $f[\text{argmin}([|U_p - N|, |U_q - N|])] \leftarrow 1$;
  else
    $f[\text{argmin}([B_p, B_q])] \leftarrow 1$;
  return f

For CAST-DU, we introduce an additional feature named the used data budget R. This feature is derived by tracking and recording the data already utilized for downloading previous chunks. For CAST-Live, we exclude the inputs I_t, R_t, and r_t as they may not be available in a live streaming scenario.

Outputs. The output of CAST and CAST-DU is represented by a v-dimensional vector, indicating the probabilities of selecting each video bitrate under the current environment. In CAST, CAST-DU, and CAST-Live, we set v to 6, corresponding to the bitrate levels used in [10].

Network Architecture. In Figure 4, the neural network utilized in CAST consists of six Conv1D layers with a feature number of 128 and a kernel size of 4, followed by four fully connected layers with 128 neurons. For CAST-DU, we introduce an additional fully connected layer with 128 neurons to accommodate the used data budget as in-

put. These layers are responsible for extracting the player environment feature, intricate chunk feature, and simple chunk feature. The network is then divided into the policy and value networks by combining these features through a concatenated layer.

The policy network produces a v-dimensional vector that represents the probabilities of selecting each bitrate, while the value network generates a scalar value. In the policy network, the activation function ReLU is used for each layer, and *softmax* is applied to the last layer to obtain the output probabilities. Meanwhile, the value network employs *tanh* as the activation function.

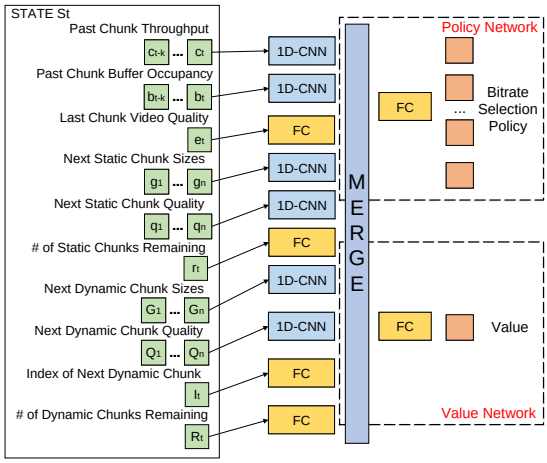


Fig. 4: Neural network architecture of CAST.

5 Evaluation

5.1 Methodology

Implementation. The implementation of CAST, CAST-DU, and CAST-Live is carried out using TensorFlow [49], and the neural network is constructed using the TFlearn [50] library. The policy network and value network share the same network structure and hyperparameters. To achieve convergence, we set the number of sampled trajectories D as 16 and the learning rate as 10^{-4}. We leverage ***eight parallel agents*** instead of a single process on a workstation equipped with an Intel Xeon W-2255 Processor and an NVIDIA RTX3090 GPU card to expedite the training process. The training lasts approximately 12 hours, after which a stable convergence is reached.

Virtual Player. To train CAST and CAST-DU, we employ a faithful Python-based virtual player with a maximum buffer size of 60 seconds that accurately simulates the dynamics of video streaming using network traces and video information [10], [11]. For training CAST-Live, the maximum playback buffer is set to 6 seconds [60]. This approach is more efficient than training on an actual streaming platform and can significantly reduce the training time.

Network Traces. To assess the performance of CAST and CAST-DU under various real-world network conditions, we utilize two widely used public datasets, namely HSDPA [51] and FCC [52], to generate bandwidth traces. For our evaluation, we partition the dataset by randomly selecting 80% of the traces for training, while reserving the remaining 20% for testing [10].

To thoroughly evaluate the robustness and generalizability of our method in handling unseen network scenarios, we extend our evaluation to include network traces from three additional datasets, namely Oboe [59], LTE [58], and Puffer 2022 [13]. The Oboe dataset comprises measurements collected from wired, WiFi, and cellular networks, providing a diverse range of network conditions. The LTE dataset comprises throughput measurements captured while mobile devices streamed video content over LTE networks in China. The Puffer dataset is sourced from the Puffer platform, and for this analysis, we have selected traces from the year 2022.

Video Parameters. For our evaluation of CAST and CAST-DU, we opt for the widely used test video, "Envivo-Dash3" [5], which is encoded with six different bitrates: 300Kbps, 750Kbps, 1200Kbps, 1850Kbps, 2850Kbps, and 4300Kbps. Each video chunk has a duration of 4 seconds.

For CAST-Live, To assess the performance of CAST-Live, we opt for the "Big Buck Bunny (BBB)" video, which is divided into 60 chunks. These chunks are encoded at bitrates of $\{263, 791, 1245, 2134, 3079, 4220\}$ Kbps, each with a duration of 2 seconds.

Chunk Classification. We differentiate video chunks into intricate and simple categories based on their size distribution. For this study, we designate the 1850Kbps bitrate as the reference track and divide the chunks into four categories, namely Q1-Q4, with Q4 having the largest chunk size. Chunks falling into Q4 are classified as intricate chunks, while the remaining chunks belong to the simple chunk category. Note that the classification method is flexible, and alternative approaches can also be utilized, such as dividing the chunks into five categories.

Additionally, for CAST-Live, considering the complexity of each chunk event, we set the number of intricate chunks to 15 out of 60 chunks.

Benchmarks. To assess the effectiveness of our proposed CAST algorithm, we have chosen to compare its performance against several established ABR schemes, encompassing both heuristic-based and learning-based approaches. Specifically, we have selected the following representative ABR schemes for evaluation:

(i) FESTIVE [35]: This method utilizes the harmonic mean of the past five throughput measurements as the capacity prediction value and selects the highest available bitrate below the predicted capacity. (ii) BOLA [37]: BOLA reformulates the adaptive bitrate streaming task as a utility maximization problem addressed through the Lyapunov function. (iii) MPC [6]: This hybrid-based approach combines both the rate and buffer signal and chooses the bitrate by maximizing a linear-based reward function using information on estimated throughput and buffer occupancy. (iv) RobustMPC (RMPC) [6]: Similar to MPC, RobustMPC also maximizes the linear-based reward function using estimated throughput and buffer occupancy information. However, to mitigate the impact of estimation errors, RobustMPC employs a normalization technique for capacity estimations, which involves dividing the capacity estimations by the maximum prediction error observed in the past five chunks. (v) Zwei [16]: Zwei, a state-of-the-art learning-based method, utilizes self-play reinforcement learning to derive a bitrate selection policy without relying on any prior environmental assumptions. This approach has demonstrated

substantial improvements in QoE compared to heuristic-based and learning-based algorithms, such as Comyco [11] and Pensieve [10]. As a result, we chose to compare our approach with Zwei and excluded Comyco and Pensieve due to its superior performance relative to other learning-based methods.

Furthermore, there is limited work considering chunk complexity [3], [5], with DAVS [5] being the most state-of-the-art method that employs imitation learning for training. To facilitate a clear comparison, we evaluate our scheme against DAVS separately in Section 5.5.

Evaluation Metrics. In this study, we adopt the Elo rating system [53], a well-established method for determining the relative skill levels of players in zero-sum games, as our evaluation metric. In self-play reinforcement learning, the Elo score is also an important metric for assessing an agent's performance and skill level compared to other agents. It provides a quantitative measure that ranks agents based on their performance, thereby helping to identify more effective strategies that meet users' actual needs. The Elo rating system computes a player's skill level as a numerical value that changes depending on the outcome of each game. Specifically, after each game, the winner gains points from the loser, and the number of points gained or lost is determined by the difference between the ratings of the two players [54]. For this experiment, we set the initial Elo score to 1000, which is consistent with previous studies [16], [17].

QoE Representation. To assess the overall performance of different ABR schemes, we adopt the following QoE function. In practice, quality switching is often considered negligible in most cases and thus omitted in the QoE metric, as reported in prior studies [16], [45]. Here, V represents VMAF, T denotes rebuffering time, and φ, ζ, ψ are set as 3, 1, and 100, respectively [55].

$$QoE = \sum_{h \in Intricate} \varphi * V(R_h) + \sum_{h \in Simple} \zeta * V(R_h) - \psi \sum_{i=1}^{N} T_i.$$

The inclusion of the QoE function in our evaluation is solely aimed at providing an assessment using the widely-used linear-based reward function (recall that our method is not trained by the reward function). The differentiation between intricate and simple chunks in the QoE function is based on their varying impact on users' viewing experiences. Acknowledging the higher significance of intricate chunks for viewers, we assign them higher priority, while simple chunks receive lower priority. The weight configuration is adjustable, and we set their weight ratio to 3:1. Such settings for the reward-engineering methods encourage the policy to select higher quality for intricate chunks. This differential QoE function design choice for intricate and simple chunks is inspired by insights from [32], [55].

5.2 Elo Score Comparison

We initially employ the Elo rating system to assess the performance of various ABR methods. As illustrated in Figure 5, our analysis reveals that CAST surpasses these ABR benchmarks, showcasing performance enhancements of 11.9% (Zwei), 19.62% (RobustMPC), 97.93% (MPC), 43.71% (BOLA), and 39.13% (FESTIVE), respectively.

Specifically, the fluctuations in the Elo scores of our CAST method represent the dynamic evolution of performance during the offline training process, reflecting changes in Elo over training epochs. In contrast, the Elo scores for other comparative schemes, such as the learning-based method Zwei, are derived from pre-trained models, reflecting their performance levels. The superiority of CAST can be attributed to two pivotal factors: (i) its ability to prioritize intricate chunks without excessively compromising the overall quality of all chunks and triggering prolonged rebuffering time; and (ii) its reliance on well-defined objectives rather than the fixed-weight linear reward function.

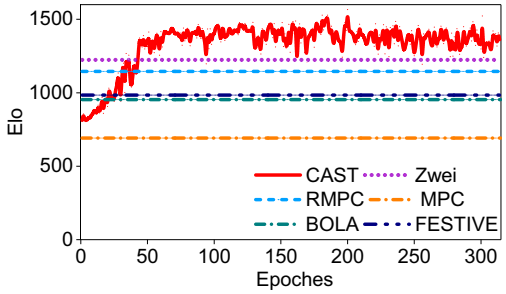


Fig. 5: Elo rating for different ABR algorithms over the HSDPA traces.

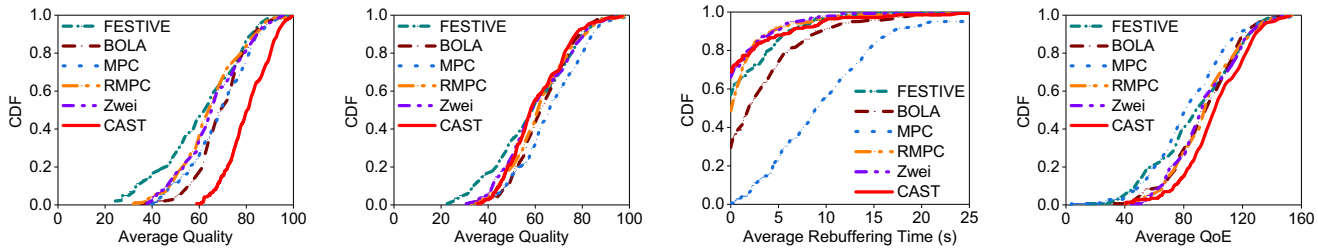
5.3 CAST vs. Existing ABR Approaches

Figure 6 - Figure 8 offer a comprehensive comparative analysis of the performance of different ABR schemes concerning intricate/all chunk quality, rebuffering time, and QoE across the HSDPA and FCC datasets. The evaluation focused only on CAST trained with the FCC and HSDPA datasets. Several key insights can be derived from these findings.

Firstly, in Figure 6(a) and Figure 7(a), CAST consistently outperforms existing algorithms in terms of the quality of intricate chunks on both network datasets. The average quality of intricate chunks achieved by CAST, as depicted in Figure 8, hovers around 80 for both datasets. This represents an improvement of up to 32.87% on HSDPA traces and 32.55% on FCC traces compared with existing methods. Additionally, compared to the heuristic MPC and RobustMPC, our method enhances the average playback quality for intricate chunks by 14.9% and 18.66%, respectively, over the FCC traces. These results affirm the efficacy of prioritizing the optimization of intricate chunk quality as a primary objective. Additionally, according to [61], a VMAF score exceeding 80 corresponds to good quality, further validating CAST's ability to provide superior quality for intricate chunks.

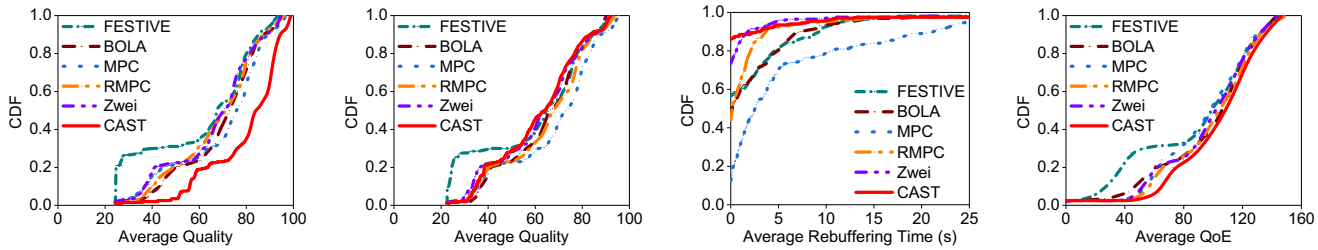
Secondly, as depicted in Figure 6(b) and Figure 7(b), the overall quality of all chunks produced by CAST remains comparable to existing methods. Additionally, we also conduct a comparison of the quality of simple chunks in CAST with those of existing methods. For example, in the HSDPA dataset, the quality of simple chunks in CAST is on par with the state-of-the-art learning-based method, Zwei, with no more than a difference of 6 on VMAF. Such a difference is not easily perceivable by users, as a VMAF difference of 6 or higher is considered noticeable to viewers [62].

Thirdly, from Figure 6(c) and Figure 7(c), we observe that CAST effectively minimizes rebuffering time. Notably, in the HSDPA dataset, CAST achieves zero rebuffering time in 69% of the traces, and this figure rises to 86% for the FCC dataset.



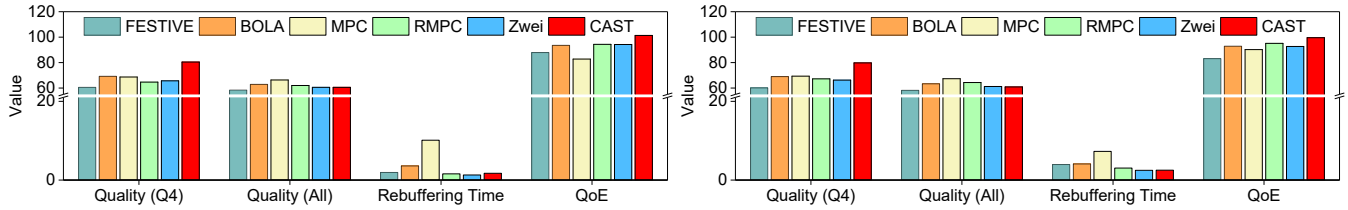
(a) Average Quality (Intricate Chunks). (b) Average Quality (All Chunks). (c) Average Rebuffering Time. (d) Average QoE.

Fig. 6: Performance comparison of CAST vs. existing algorithms over the HSDPA dataset.



(a) Average Quality (Intricate Chunks). (b) Average Quality (All Chunks). (c) Average Rebuffering Time. (d) Average QoE.

Fig. 7: Performance comparison of CAST vs. existing algorithms over the FCC dataset.



(a) Average Value (HSDPA). (b) Average Value (FCC).

Fig. 8: Average value of CAST vs. existing algorithms over the HSDPA and FCC datasets.

When compared to the heuristic MPC and RobustMPC, our method decreases rebuffering time by 188.23% and 20.41%, respectively, over the FCC traces. On average, CAST reduces rebuffering time by up to 486.14% on the HSDPA dataset and 188.27% on the FCC dataset compared with existing approaches. This underscores CAST's capability to deliver a seamless viewing experience by substantially mitigating playback interruptions.

Lastly, CAST persistently achieves the highest QoE compared with existing methods (Figure 6(d) and Figure 7(d)). On average, CAST improves the average QoE by 7.45% to 22.41% and 4.74% to 19.89% over the HSDPA and FCC traces, respectively, validating its remarkable effectiveness.

5.4 Performance under Different Settings

5.4.1 Different Network Environments

In the previous experiments, CAST was trained and evaluated on the HSDPA and FCC datasets. However, in real-world scenarios, CAST might encounter various network conditions. To assess the generalization capability of CAST, we conducted tests on three additional datasets: LTE [58], Oboe [59], and Puffer 2022 [13]. The results are presented in Figure 9 - Figure 12.

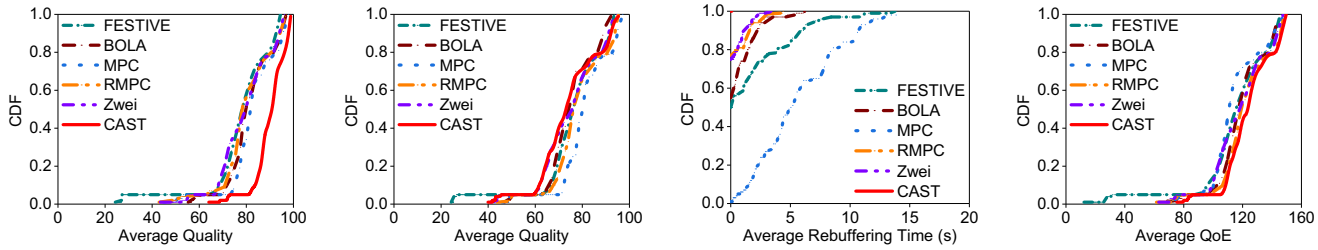
As depicted in Figure 9(a), Figure 10(a) and Figure 11(a), we observe that CAST's ability to enhance the quality of intricate chunks remains optimal even under new network conditions, resulting in improvements of up to 16.08%, 13.32% and 21.08% across the LTE, Oboe, and Puffer 2022 datasets, respectively (Figure 12). Furthermore, CAST demonstrates similar overall quality for all chunks and exhibits low rebuffering time. As indicated in Figure 9(c), the rebuffering time for CAST is 0 in all traces. Overall, CAST achieves the highest QoE in both LTE and Oboe traces, surpassing existing methods by 1.9%-8.14%, 1.13%-8.11%, and 3.63%-22.93% in the LTE, Oboe, and Puffer 2022 traces, respectively, validating its effectiveness and robustness.

5.4.2 Different Player Settings

Here, we perform a sensitivity analysis of CAST using HSDPA traces by altering player configurations and varying the maximum buffer size to 30 seconds. The results, depicted in Figure 13, demonstrate the consistent superiority of CAST. Specifically, it excels in maintaining the quality for intricate chunks (as shown in Figure 13(a)), displaying improvements ranging from 15.93% to 33.64%. Furthermore, CAST effectively minimizes rebuffering time, as evidenced by 73% of traces achieving a zero rebuffering time. Overall, CAST stands out by delivering the highest QoE.

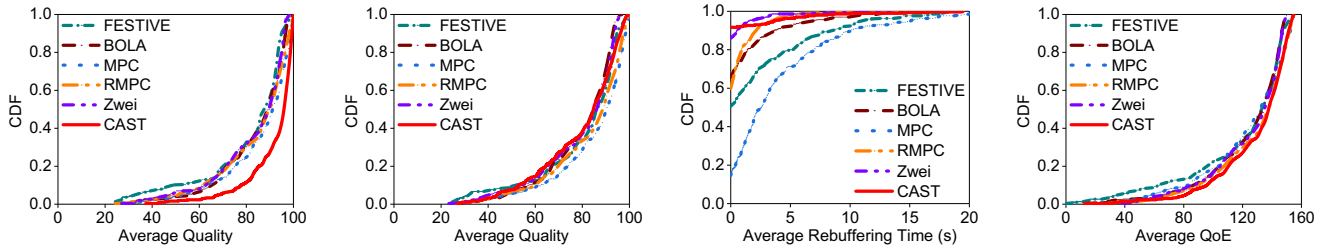
5.5 CAST vs. Complexity-aware scheme DAVS

Figures 14 and 15 depict the performance of CAST and DAVS. As observed in Figures 14(a) and 15(a), CAST



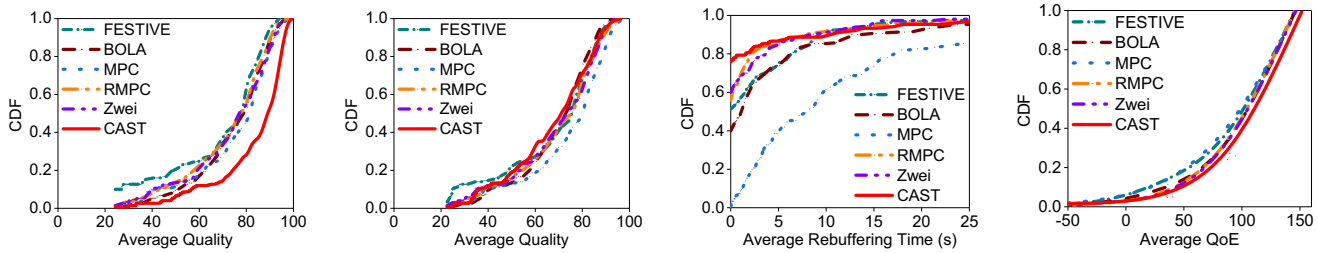
(a) Average Quality (Intricate Chunks). (b) Average Quality (All Chunks). (c) Average Rebuffering Time. (d) Average QoE.

Fig. 9: Performance comparison of CAST vs. existing algorithms over the LTE dataset.



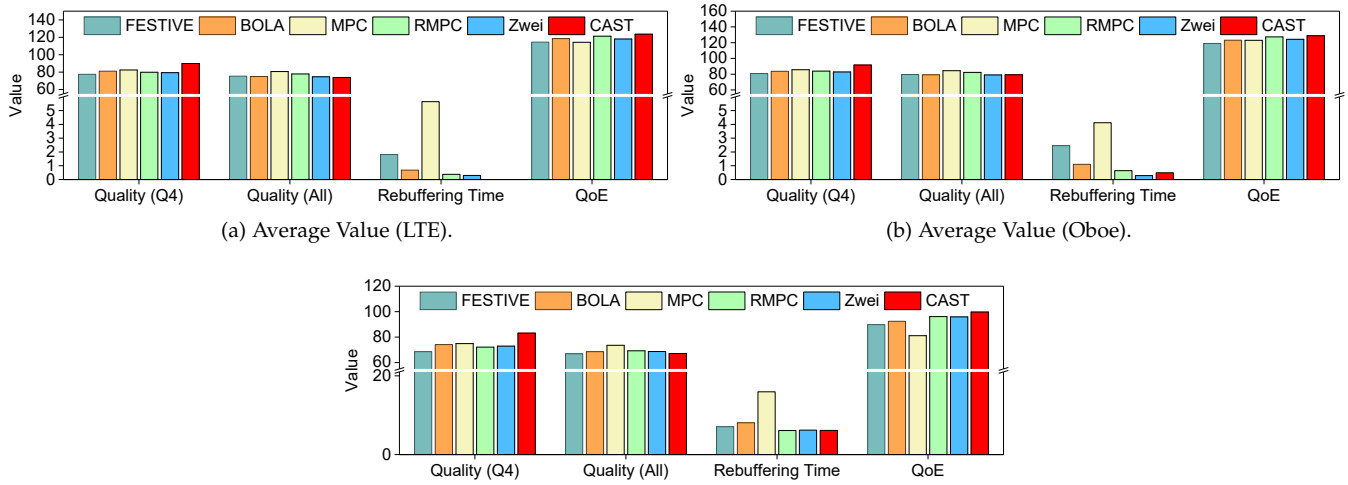
(a) Average Quality (Intricate Chunks). (b) Average Quality (All Chunks). (c) Average Rebuffering Time. (d) Average QoE.

Fig. 10: Performance comparison of CAST vs. existing algorithms over the Oboe dataset.



(a) Average Quality (Intricate Chunks). (b) Average Quality (All Chunks). (c) Average Rebuffering Time. (d) Average QoE.

Fig. 11: Performance comparison of CAST vs. existing algorithms over the Puffer 2022 dataset.

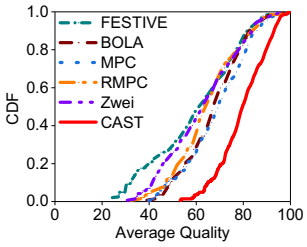


(a) Average Value (LTE).

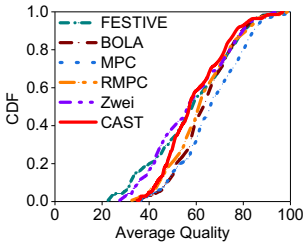
(b) Average Value (Oboe).

(c) Average Value (Puffer 2022).

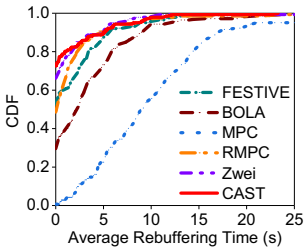
Fig. 12: Average value of CAST vs. existing algorithms over the LTE, Oboe, and Puffer 2022 datasets.



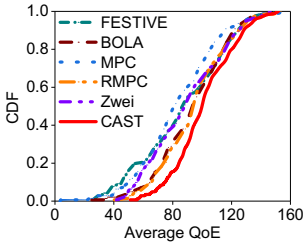
(a) Average Quality (Intricate Chunks).



(b) Average Quality (All Chunks).



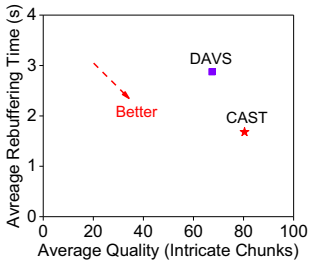
(c) Average Rebuffering Time.



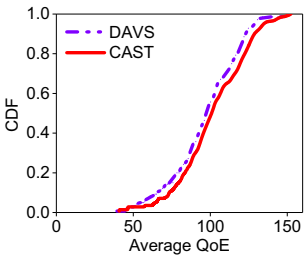
(d) Average QoE.

Fig. 13: CAST vs. existing algorithms over the HSDPA dataset with a different buffer size setting (30 seconds).

achieves higher playback quality for intricate chunks and lower rebuffering time compared to DAVS. This improvement is primarily attributed to DAVS relying on imitation learning with guidance from MPC. However, MPC's bitrate selection depends on a fixed-weighted linear QoE function without explicit buffer occupancy control, resulting in more rebuffering events. In contrast, CAST prioritizes the metrics of rebuffering time and playback quality for intricate chunks, leading to superior performance. Overall, as shown in Figures 14(b) and 15(b), CAST enhances the average QoE by 4.97% and 16.5% over the HSDPA and FCC traces, respectively, compared to DAVS.

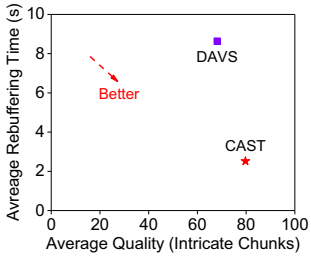


(a) Average Quality (Intricate Chunks).

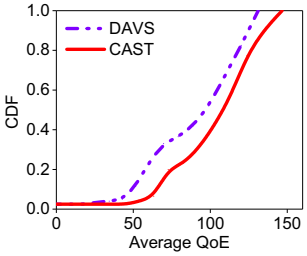


(b) Average QoE.

Fig. 14: CAST vs. DAVS over the HSDPA dataset.



(a) Average Quality (Intricate Chunks).

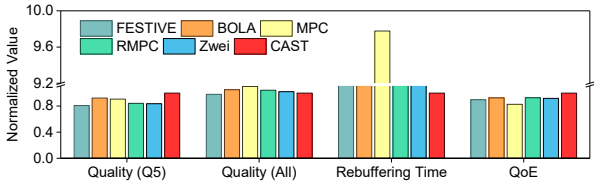


(b) Average QoE.

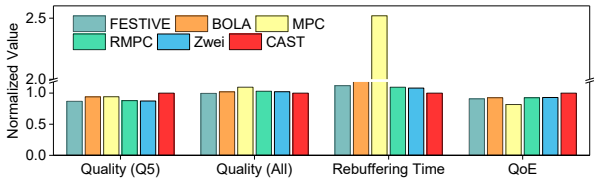
Fig. 15: CAST vs. DAVS over the FCC dataset.

5.6 CAST with Different Chunk Classification Methods

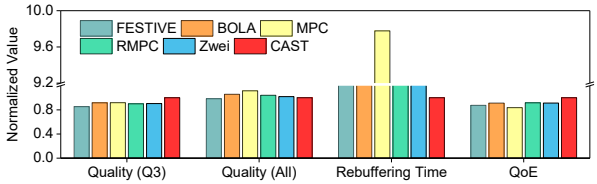
Here, we categorize video chunks into three and five groups based on their sizes. In the three-category classification (Q1-Q3), chunks in Q3 are considered intricate, while the rest are deemed simple. For the five-category classification (Q1-Q5), chunks in Q5 are labeled intricate, with the remaining chunks labeled simple. Figure 16 illustrates the performance of our method and the compared baselines under different category methods using both the HSDPA and Puffer 2022 traces. The results of each metric are standardized based on the outcomes of CAST. We observe that our method maintains high playback quality for intricate chunks across different category methods. For instance, when categorizing into five groups, our method improves the quality for intricate chunks by up to 15.18% according to the Puffer 2022 traces. Additionally, our method achieves low rebuffering times across various network traces and classification approaches. Overall, our method enhances the viewing experience by up to 20.76% across the HSDPA dataset when divided into five categories.



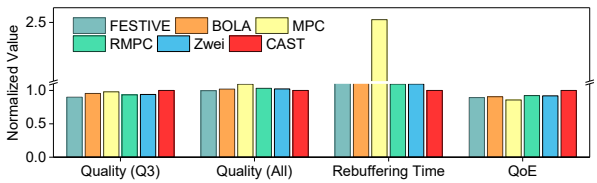
(a) HSDPA (Q5).



(b) Puffer 2022 (Q5).



(c) HSDPA (Q3).



(d) Puffer 2022 (Q3).

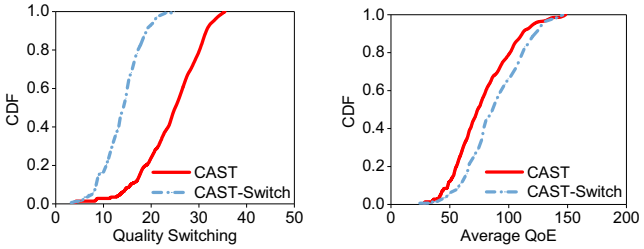
Fig. 16: Average value of CAST vs. existing algorithms over different classification methods.

5.7 CAST with Quality Switching

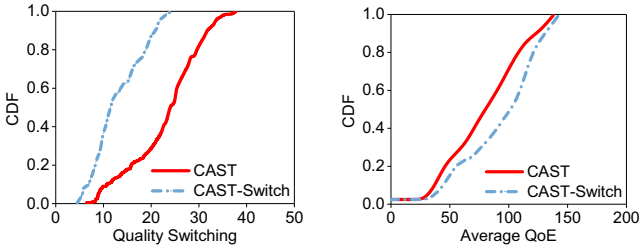
We integrate quality switching considerations into the training rule by establishing a priority order as follows: 1) minimizing rebuffering time, 2) maximizing quality for

intricate chunks, 3) maximizing quality for simple chunks, and 4) minimizing the quality switching between two consecutive chunks. The evaluation function is defined as: $QoE_{switch} = \sum_{h \in Intricate} \varphi * V(R_h) + \sum_{h \in Simple} \zeta * V(R_h) - \psi \sum_{i=1}^{N} T_i - \omega \sum_{i=2}^{N} |R_{i+1} - R_i|$, where φ, ζ, ψ, and ω are set as 3, 1, 100, and 1, respectively.

As depicted in Figure 17, the integration of quality switching considerations in CAST-Switch leads to a substantial reduction in the frequency of quality switches compared with the conventional CAST approach, with decreases of 74.57% and 77.39% observed in the HSDPA and FCC traces, respectively. CAST-Switch also exhibits significantly smaller quality fluctuations compared to BOLA, with a reduction of 32.49% observed in HSDPA traces (*figures omitted*). Moreover, when assessing the QoE improvement, CAST-Switch exhibits enhancements of 14.06% and 17.85% for the HSDPA and FCC traces, respectively.



(a) Quality Switching (HSDPA). (b) Average QoE (HSDPA).



(c) Quality Switching (FCC). (d) Average QoE (FCC).

Fig. 17: Comparative performance analysis of CAST with and without quality switching.

Here, we demonstrate how integrating bitrate switching into the training rule can mitigate quality fluctuations. In this analysis, bitrate switching was assigned the lowest priority. However, prioritizing bitrate switching can further reduce its impact. By assigning it a higher priority, we can tailor the algorithm to accommodate different user preferences, effectively meeting the needs of users who prioritize bitrate switching.

5.8 CAST-DU vs. Existing ABR Schemes

In this section, we assess the performance of CAST-DU under a data budget constraint. Specifically, we choose the 2850Kbps bitrate as the reference track and set the data budget scale factor γ to 0.7. This implies that the data budget is calculated as 70% of the cumulative size of all chunks within the reference track. *Note that other values of γ can also be considered for analysis. For instance, we also set γ to 1.4, and CAST-DU continues to exhibit superior performance.*

Figure 18 depicts the Elo scores of various ABR algorithms when factoring in the data budget. We note that CAST-DU achieves the highest Elo score among the compared benchmarks, showing improvements ranging from 6.32% to 103.11%.

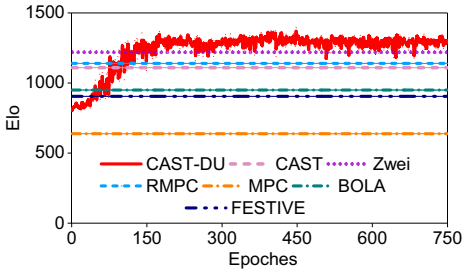


Fig. 18: Elo rating for different ABR algorithms over the HSDPA traces considering the data budget.

Next, we examine the correlation between playback quality and data usage across the HSDPA dataset. As demonstrated in Figure 19, the data usage of CAST-DU remains within the constrained data budget for all traces, thereby affirming the successful integration of data budget considerations. Furthermore, the average QoE of CAST-DU is also comparable to the compared baselines, such as Zwei and RMPC, hovering around 94 (figure omitted), underscoring the efficacy of CAST-DU in ensuring a satisfying viewing experience while achieving data savings. Additionally, since DAVS also does not consider data usage, it easily exceeds the designated data budget, making it less effective compared to CAST-DU (figure omitted).

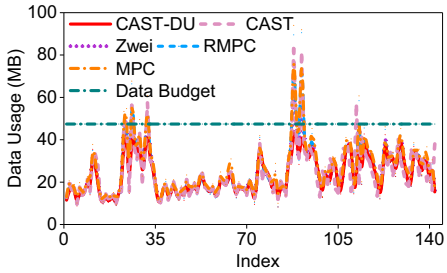


Fig. 19: Data usage for different ABR algorithms over the HSDPA traces.

Recalling the details provided in Section 3.2, there exist some data-capped methods to constrain data usage. Here, we also apply the data-capped approach to vanilla CAST, MPC and Zwei, and subsequently compare their results to those of CAST-DU. As depicted in Figure 20, the analysis reveals that while the data usage of MPC-Capped, Zwei-Capped, and CAST-Capped stays within the prescribed data budget, these strategies tend to exhibit suboptimal bandwidth utilization. In contrast, CAST-DU efficiently narrows the disparity between data usage and the target budget. Furthermore, we calculate and normalize the gap between data usage and the target budget for each trace, using CAST-DU as the reference. The results tabulated in Table 2 indicate that the gap observed in MPC-Capped, Zwei-Capped and CAST-Capped is larger compared with CAST-DU, thus validating the bandwidth efficiency superiority of CAST-DU over these capped alternatives.

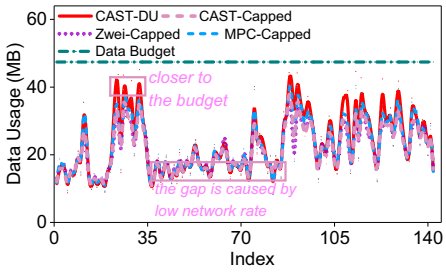


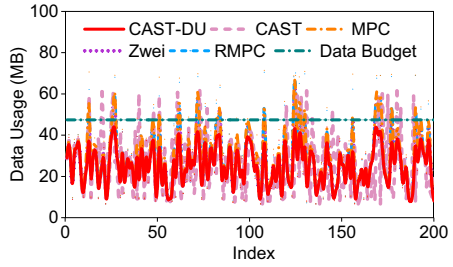
Fig. 20: Data usage for different capped ABR algorithms over the HSDPA traces.

TABLE 2: Discrepancy in data usage among various adaptation approaches.

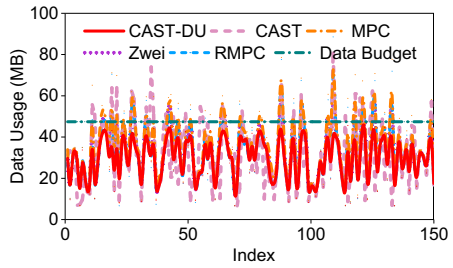
Scheme	Data Usage Gap
CAST-DU	**1**
CAST-Capped	1.07
MPC-Capped	1.03
Zwei-Capped	1.06

5.8.1 Diverse Scenarios: Different Network Environments

We vary the network environment to assess its influence on the data budget, utilizing both the FCC dataset [52] and the Puffer 2022 dataset [13]. In Figure 21, we present the data usage comparison across different ABR algorithms. Notably, CAST-DU consistently keeps the data usage below the specified data budget. Even when exposed to the unfamiliar Puffer 2022 trace, CAST-DU exhibits robustness in effectively managing data usage.



(a) FCC.

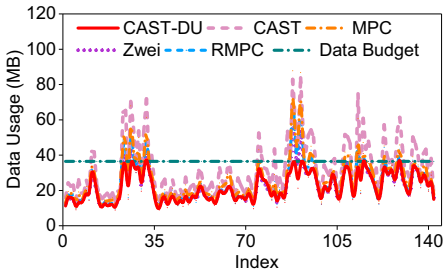


(b) Puffer 2022.

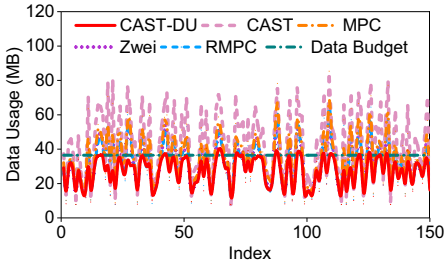
Fig. 21: Data usage for different ABR algorithms over different traces.

5.8.2 Diverse Scenarios: Different Videos and Data Usages

To enhance the effectiveness and robustness of CAST-DU, we conduct training on multiple video datasets [66]. Specifically, we select six bitrates: {375, 750, 1050, 1750, 2350, 4300} kbps, with each chunk encoded at a duration of 4 seconds. The 2350 kbps bitrate serves as the reference track, and the data budget for each video is determined as 70% of the cumulative size of all chunks within the reference track. The remaining settings align with CAST. We randomly allocate 80% of the video traces for training and reserve the remaining 20% for testing. Figure 22 depicts the performance of CAST-DU across the Puffer 2022 traces, showcasing results on a randomly selected test video from the music category. We observe that on both network traces, when the network bandwidth is sufficient, the data usage of our CAST-DU is always close to the predefined data usage. The low data usage of all algorithms indicates a low network bandwidth.



(a) HSDPA.



(b) Puffer 2022.

Fig. 22: Data usage for different ABR algorithms over different traces with the music video.

5.8.3 Diverse Scenarios: Comparison with PSWA

Furthermore, we applied PSWA to MPC and conducted a performance comparison with CAST-DU using the Puffer 2022 traces. Figure 23 illustrates the data usage of MPC-PSWA and CAST-DU. Our observation indicates that CAST-DU consistently maintains the actual data usage close to and below the designated data budget, whereas MPC-PSWA does not achieve the same level of control, resulting in data usage either falling too low or rising too high.

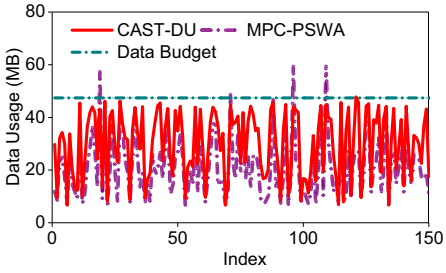


Fig. 23: Data usage comparison of CAST-DU and MPC-PSWA on the Puffer 2022 traces.

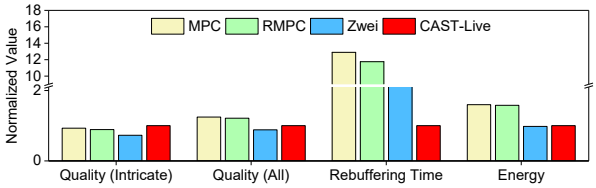
5.9 CAST-Live vs. Exsiting ABR Schemes

For CAST-Live, the target data usage value is set to 70% of the accumulated size of the reference track [18], which we have chosen as 3079 Kbps. Figure 24 illustrates the

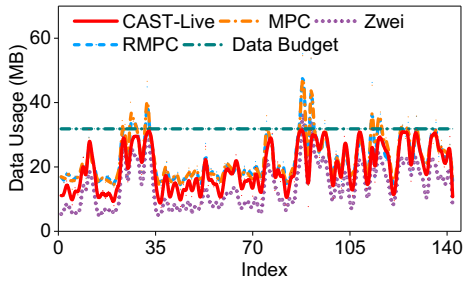
TABLE 3: Subjective tests from 20 real users via watching an online video with different ABR schemes.

Questions	Disagree	Neutral	Agree
CAST provides you the optimal viewing experience.	2	2	16
CAST delivers a good overall playback quality.	1	2	17
CAST offers you a satisfactory quality for chunks with intricate scenes.	0	1	19
CAST incurs few playback freezing events.	1	4	15

performance of various ABR algorithms in the live streaming scenario. As depicted in Figure 24(a), presenting the average value of each metric normalized to the results of CAST-Live, we observe that CAST-Live achieves the highest playback quality compared to the benchmarks. Moreover, CAST-Live sustains commendable overall playback quality and demonstrates low rebuffering time, showcasing its ability to maintain a good playback buffer level even under limited buffer settings. Through evaluation, we note that our method keeps the average buffer occupancy around 3 seconds, which is smaller than that of the compared baselines (all around 5 seconds), resulting in reduced latency. Additionally, the energy cost of CAST-Live is significantly lower than that of MPC and RMPC. Zwei also achieves a low energy cost, but this comes at the expense of lower playback quality. Figure 24(b) illustrates the data usage of different methods, revealing that CAST-Live effectively constrains data usage close to the designated data budget, avoiding additional costs for users' mobile data in practical scenarios.



(a) Average Normalized Value.



(b) Data Usage.

Fig. 24: Performance evaluation of ABR algorithms in live streaming scenarios using HSDPA traces.

5.10 Subjective Tests from Real Users

To assess the practical performance of CAST, we establish a testbed configuration comprising a client player utilizing `Dash.js` [63], an ABR server, and a content server responsible for hosting video content. To alleviate the computational load on the video client [10], CAST is implemented on the local ABR server. To evaluate the real-world effectiveness of various ABR schemes, we enlist the participation of 20 volunteers to watch an online video within an HSDPA network environment.

Given the complexity of capturing viewers' subjective experiences, we employ a questionnaire-based approach, asking participants to rate their responses rather than assigning quantitative scores to overall viewing quality [21], [65]. The questionnaire outcomes are summarized in Table 3. The subjective evaluation data analysis demonstrates that CAST contributes to an enjoyable viewing experience, as endorsed by 16 out of 20 volunteers. Furthermore, 19 out of 20 volunteers agree with CAST's capability to deliver satisfactory quality for intricate video chunks. These findings affirm the practical effectiveness of CAST.

6 Future Work

Similar to many existing ABR algorithms such as [9], [20], [21], [27], [67], [68], our paper focuses on application-layer bitrate selection. In practical application scenarios, the model can be deployed at the client side, usually with only one player per client. In this way, each client can independently make decisions based on the model according to its perceived network conditions and player states. Thus, our approach supports multi-user scenarios. However, ensuring fairness in QoE among multiple clients is also an important area in the video streaming domain [69], [70]. Addressing the challenges of QoE fairness across diverse clients is a key direction for our future research. We aim to enhance the adaptability of our algorithm in various multi-client scenarios by integrating sub-layer information. Specifically, we plan to utilize data from the transport layer to inform bitrate selection for clients sharing a common bottleneck link, ultimately achieving a high level of fairness in QoE.

7 Conclusions

This paper introduces CAST, a novel self-play reinforcement learning-based bitrate adaptation solution that considers scene complexity. Additionally, we present CAST-DU, a variant that incorporates data budget constraints into the adaptation process. Moreover, we introduce CAST-Live to enable our method to attain good performance under live streaming scenarios with tight buffer lengths, considering energy costs. Through explicit goal-oriented training, our methods effectively meet specific system requirements. Experimental evaluations highlight CAST's superiority over existing ABR methods in terms of quality for intricate chunks across diverse network conditions. Furthermore, CAST maintains comparable average quality for all chunks as prior methods while minimizing rebuffering incidents. Meanwhile, CAST-DU successfully constrains data usage within budget limitations while maintaining satisfactory QoE. Moreover, CAST-Live maintains robustness under tight buffer sizes while achieving low energy costs.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 62132022 and 62302524; the Key Research and Development Program of Hunan under Grant 2022WK2005; the Natural Science Foundation of Hunan Province, China, under Grant 2021JJ30867; the Science and Technology Innovation Program of Hunan Province, China, under Grant 2024JJ6531; and by the computing resources at the High Performance Computing Center of Central South University.

REFERENCES

[1] W. Li, J. Huang, Y. Liang, J. Liu, W. Zhang, W. Lyu, and J. Wang, "CAST: An Intricate-Scene Aware Adaptive Bitrate Approach for Video Streaming via Parallel Training," in Proceedings of Springer ICA3PP, 2023.

[2] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62-67, 2011.

[3] Y. Qin, S. Hao, K.R. Pattipati, F. Qian, S. Sen, B. Wang and C. Yue, "ABR Streaming of VBR-encoded Videos: Characterization, Challenges, and Solutions," in *Proc. ACM CoNEXT*, 2018, pp. 366-378.

[4] Y. Qin, S. Hao, K.R. Pattipati, and F. Qian, "Quality-aware Strategies for Optimizing ABR Video Streaming QoE and Reducing Data Usage," in *Proc. ACM MMSys*, 2019, pp. 189-200.

[5] W. Li, J. Huang, S. Wang, C. Wu, S. Liu and J. Wang, "An Apprenticeship Learning Approach for Adaptive Video Streaming Based on Chunk Quality and User Preference," *IEEE Trans. on Multimedia*, vol. 25, pp. 2488-2502, 2023.

[6] X. Yin, A. Jindal, V. Sekar and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proc. ACM SIGCOMM*, 2015, pp. 325-338.

[7] P.K. Yadav, A. Shafiei, and W.T. Ooi, "QUETRA: A Queuing Theory Approach to DASH Rate Adaptation," in Proceedings of ACM MM, 2017.

[8] Y. Qin, R. Jin, S. Hao, K.R. Pattipati, and F. Qian, "A Control Theoretic Approach to ABR Video Streaming: A Fresh Look at PID-based Rate Adaptation," in Proceedings of IEEE INFOCOM, 2017.

[9] B. Wang, F. Ren, J. Yang and C. Zhou, "Improving the Performance of Online Bitrate Adaptation with Multi-Step Prediction Over Cellular Networks," IEEE Transactions on Mobile Computing, vol. 20, no. 1, pp. 174-187, 2021.

[10] H. Mao, R. Netravail, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," in *Proc. ACM SIGCOMM*, 2017, pp. 197-210.

[11] T. Huang et al., "Quality-Aware Neural Adaptive Video Streaming With Lifelong Imitation Learning," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2324-2342, 2020.

[12] N. Kan, C. Li, C. Yang, W. Dai, J. Zou, and H. Xiong, "Uncertainty-aware Robust Adaptive Video Streaming with Bayesian Neural Network and Model Predictive Control," in *Proc. ACM NOSSDAV*, 2021, pp. 17-24.

[13] F.Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a Randomized Experiment in Video Streaming," in Proceedings of USENIX NSDI, 2020.

[14] D. Yuan, Y. Zhang, W. Zhang, X. Liu, H. Du, and Q. Zheng, "PRIOR: Deep Reinforced Adaptive Video Streaming with Attention-Based Throughput Prediction," in Proceedings of ACM NOSSDAV, 2022.

[15] X. Zuo, J. Yang, M. Wang and Y. Cui, "Adaptive Bitrate with User-level QoE Preference for Video Streaming," in *Proc. IEEE INFOCOM*, 2022, pp. 1279-1288.

[16] T. Huang, R. Zhang and L. Sun, "Zwei: A Self-play Reinforcement Learning Framework for Video Transmission Services," *IEEE Trans. on Multimedia*, vol. 24, no. 1, pp. 1350-1365, 2022.

[17] T. Huang, X. Yao, C. Wu, RX. Zhang, Z. Pang, and L. Sun, "Tiyuntsong: A Self-Play Reinforcement Learning Approach for ABR Video Streaming," in *Proc. IEEE ICME*, 2019, pp. 1678-1683.

[18] Y. Qin, C. Shende, C. Park, S. Sen, and B. Wang, "DataPlanner: Data-budget Driven Approach to Resource-efficient ABR Streaming," in Proceedings of ACM MMSys, 2021.

[19] T. Huang, R. Zhang, and L. Sun, "Self-play Reinforcement Learning for Video Transmission," in Proceedings of ACM NOSSDAV, 2020.

[20] T. Huang, C. Zhou, RX. Zhang, C. Wu, X. Yao and L. Sun, "Stick: A Harmonious Fusion of Buffer-based and Learning-based Approach for Adaptive Streaming," in *Proc. IEEE INFOCOM*, 2020, pp. 1967-1976.

[21] W. Li, J. Huang, W. Lyu, B. Guo, W. Jiang and J. Wang, "RAV: Learning-Based Adaptive Streaming to Coordinate the Audio and Video Bitrate Selections," *IEEE Trans. on Multimedia*, 2022, doi: 10.1109/TMM.2022.3198013.

[22] W. Li, J. Huang, Y. Liang, J. Liu, and F. Gao, "Synthesizing Audio and Video Bitrate Selections via Learning from Actual Requirements," in Proceedings of IEEE ICME, 2022.

[23] W. Li, J. Huang, J. Liu, W. Jiang and J. Wang, "Learning Audio and Video Bitrate Selection Strategies Via Explicit Requirements," IEEE Transactions on Mobile Computing, vol. 23, no. 4, pp. 2849-2863, 2024.

[24] A. Bentaleb, M. N. Akcay, M. Lim, A. C. Begen and R. Zimmermann, "Catching the Moment With LoL$^+$ in Twitch-Like Low-Latency Live Streaming Platforms," IEEE Transactions on Multimedia, vol. 24, pp. 2300-2314, 2022.

[25] Y. Li, X. Zhang, C. Cui, S. Wang and S. Ma, "Fleet: Improving Quality of Experience for Low-Latency Live Video Streaming," IEEE Transactions on Circuits and Systems for Video Technology, vol. 33, no. 9, pp. 5242-5256, 2023.

[26] T. Huang, C. Zhou, R. Zhang, C. Wu and L. Sun, "Learning Tailored Adaptive Bitrate Algorithms to Heterogeneous Network Conditions: A Domain-Specific Priors and Meta-Reinforcement Learning Approach," IEEE Journal on Selected Areas in Communications, vol. 40, no. 8, pp. 2485-2503, 2022.

[27] C. Qiao, G. Li, Q. Ma, J. Wang, and Y. Liu, "Trace-Driven Optimization on Bitrate Adaptation for Mobile Video Streaming," IEEE Transactions on Mobile Computing, vol. 21, no. 6, pp. 2243-2256, 2022.

[28] Xiph.Org, "Xiph.org Video Test Media," https://media.xiph.org/video/derf/, 2016.

[29] C. Liu, I. Bouazizi and M. Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," in *Proc. ACM MMSys*, 2011, pp. 169-174.

[30] ITU-T P. 910, "Subjective Video Quality Assessment Methods for Multimedia Applications," 2008.

[31] T. Huang, R. Zhang, C. Zhou, and L. Sun, "QARC: Video Quality Aware Rate Control for Real-Time Video Streaming based on Deep Reinforcement Learning," in *Proc. ACM MM*, 2018, pp. 1208-1216.

[32] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "HotDASH: Hotspot Aware Adaptive Video Streaming using Deep Reinforcement Learning," in Proceedings of IEEE ICNP, 2018.

[33] J. Huang, Q. Su, W. Li, Z. Liu, T. Zhang, S. Liu, P. Zhong, W. Jiang, and J. Wang, "Opportunistic Transmission for Video Streaming over Wild Internet," ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 18, no. 140, pp. 1-22, 2023.

[34] Y. Guan, Y. Zhang, B. Wang, K. Bian, X. Xiong, and L. Song, "PERM: Neural Adaptive Video Streaming with Multi-path Transmission," in Proceedings of IEEE INFOCOM, 2020.

[35] J. Jiang, V. Sekar and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE," in Proceedings of ACM CoNEXT, 2012.

[36] T.Y. Huang, R. Johari, N. McKeown, M. Trunnell and M. Watson, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *Proc. ACM SIGCOMM*, 2014, pp. 187-198.

[37] K. Spiteri, R. Urgaonkar and R. K. Sitaraman, "BOLA:Near-Optimal Bitrate Adaptation for Online Videos," in *Proc. IEEE INFOCOM*, 2016, pp. 1-9.

[38] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward A Practical Perceptual Video Quality Metric," http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html.

[39] T. Huang, C. Zhou, R. Zhang, C. Wu, and L. Sun, "Buffer Awareness Neural Adaptive Video Streaming for Avoiding Extra Buffer Consumption," in Proceedings of IEEE INFOCOM, 2023, pp. 1-10.

[40] G. Zhang, K. Liu, H. Hu, V. Aggarwal and J. Y. B. Lee, "Post-Streaming Wastage Analysis – A Data Wastage Aware Framework in Mobile Video Streaming," IEEE Transactions on Mobile Computing, vol. 22, no. 1, pp. 389-401, 2023.

[41] O.F. Aladag, D. Ugur, M.N. Akcay, and A.C. Begen, "Content-Aware Playback Speed Control for Low-Latency Live Streaming of Sports," in Proceedings of ACM MMSys, 2021.

[42] C. Qian, D. Liu and H. Jiang, "Harmonizing Energy Efficiency and QoE for Brightness Scaling-based Mobile Video Streaming," in Proceedings of IEEE/ACM IWQoS, 2022.

[43] L. Sun, R. K. Sheshadri, W. Zheng and D. Koutsonikolas, "Modeling WiFi Active Power/Energy Consumption in Smartphones," in Proceedings of IEEE ICDCS, 2014.

[44] C. Yue, S. Sen, B. Wang, Y. Qin, and F. Qian, "Energy Considerations for ABR Video Streaming to Smartphones: Measurements, Models and Insights," in Proceedings of ACM MMSys, 2020.

[45] T. Huang, C. Ekanadham, A.J. Berglund, and Z. Li, "Hindsight: Evaluate Video Bitrate Adaptation at Scale," in Proceedings of ACM MMSys, 2019.

[46] T-Mobile, "T-Mobile Binge On," https://goo.gl/Q9fbw6, 2018.
[47] Y. Bai, C. Jin, and T. Yu. "Near-optimal Reinforcement Learning with Self-play," in *Proc. NeurIPS*, 2020, pp. 1-12.
[48] D. Ye, *et al.*, "Mastering intricate Control in MOBA Games with Deep Reinforcement Learning," arXiv preprint arXiv:1912.09729, 2019.
[49] M. Abadi, P. Barham, J. Chen, *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. USENIX OSDI*, 2016, pp. 265-283.
[50] T. Yuan, "TF.Learn: TensorFlow's High-level Module for Distributed Machine Learning," http://tflearn.org/, 2017.
[51] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications," in *Proc. ACM MMSys*, 2013, pp. 114-118.
[52] Federal Communications Commission, Raw Data - Measuring Broadband America, https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016, 2016.
[53] R. Coulom, "Whole-History Rating: A Bayesian Rating System for Players of Time-Varying Strength," in Proceedings of Springer International Conference on Computers and Games, 2008.
[54] "Elo Rating System," https://en.wikipedia.org/wiki/Elo_rating_system.
[55] W. Li, J. Huang, S. Wang, S. Liu and J. Wang, "DAVS: Dynamic-Chunk Quality Aware Adaptive Video Streaming using Apprenticeship Learning," in *Proc. IEEE GLOBECOM*, 2020, pp. 1-6.
[56] L. Jia, T. Huang and L. Sun, "ZiXia: A Reinforcement Learning Approach via Adjusted Ranking Reward for Internet Congestion Control," in Proceedings of IEEE ICC, 2022, pp. 365-370.
[57] D. Zha, et al., "Douzero: Mastering Doudizhu with Self-Play Deep Reinforcement Learning," in Proceedings of PMLR, 2021, pp. 1-12.
[58] "LTE/WiFi Dataset," http://www.aitrans.online/competition_detail/competition_id=2, 2018.
[59] Z. Akhtar, Y.S. Nam, R. Govindan, S. Rao, J. Chen, E.K. Bassett, J. Zhan, and H. Zhang, "Oboe: Auto-tuning Video ABR Algorithms to Network Conditions," in Proceedings of ACM SIGCOMM, 2018, pp. 44-58.
[60] B. Wang, M. Xu, F. Ren, C. Zhou, and J. Wu, "Cratus: A Lightweight and Robust Approach for Mobile Live Streaming," IEEE Transactions on Mobile Computing, vol. 21, no. 8, pp. 2761-2775, 2022.
[61] Z. Li, C. Bampis, J. Novak, A. Aaron, K. Swanson, A. Moorthy, and J. De Cock, "VMAF: The Journey Continues," https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12, 2018.
[62] J. Ozer, "Finding the Just Noticeable Difference with Netflix VMAF," https://goo.gl/TGWCGV, 2017.
[63] "Dash.js," https://github.com/Dash-Industry-Forum/dash.js, 2017.
[64] M. Dong and L. Zhong, "Power Modeling and Optimization for OLED Displays," IEEE Transactions on Mobile Computing, vol. 11, no. 9, pp. 1587-1599, 2012.
[65] C. Qiao, J. Wang, Y. Liu, "Beyond QoE: Diversity Adaption in Video Streaming at the Edge," IEEE/ACM Transactions on Networking, vol. 29, no. 1, pp. 289-302, 2021.
[66] T. Huang, "Comyco-Video-Description-Dataset," https://github.com/godka/Comyco-Video-Description-Dataset/tree/master.
[67] T. Huang, R. Zhang, C. Wu, and L. Sun, "Optimizing Adaptive Video Streaming with Human Feedback," in Proceedings of ACM MM, 2023.
[68] X. Xiao, W. Wang, T. Chen, Y. Cao, T. Jiang and Q. Zhang, "Sensor-Augmented Neural Adaptive Bitrate Video Streaming on UAVs," IEEE Transactions on Multimedia, vol. 22, no. 6, pp. 1567-1576, 2020.
[69] Y. Yuan et al., "VSiM: Improving QoE Fairness for Video Streaming in Mobile Environments," in Proceedings of IEEE INFOCOM, 2022.
[70] V. Nathan, V. Sivaraman, R. Addanki, M. Khani, P. Goyal, and M. Alizadeh, "End-to-end Transport for Video QoE Fairness," in Proceedings of ACM SIGCOMM, 2019.

Weihe Li received his Masters's degree from Central South University in 2021. Currently, he is pursuing a Ph.D. degree at the University of Edinburgh. His research interests include network measurement, video streaming and data center networks.

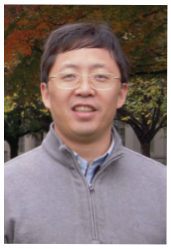
Jiawei Huang obtained his PhD (2008) and Masters degrees (2004) from the School of Computer Science and Engineering at Central South University. He also received his Bachelor's (1999) degree from the School of Computer Science at Hunan University. He is now a professor in the School of Computer Science and Engineering at Central South University, China. His research interests include cloud computing, data center networks, wireless networks, Internet video, web performance, and programmable switching architectures.

Yu Liang received his Masters's degree from the University of Leeds in 2022. Currently, he is pursuing a Ph.D. degree at Lancaster University. His research interest includes video streaming.

Qichen Su received the B.S degree and M.S. degree from Central South University, China, in 2017 and 2021 respectively. His research interests include video streaming and network transmission.

Jingling Liu is currently a lecturer at School of Computer Science and Engineering, Central South University. Before that, she received the B.E. and Ph.D degrees from Central South University, China, in 2016 and 2022, respectively. She worked as a Visiting Scholar with Singapore University of Technology and Design, Singapore, from 2021 to 2022. Her current research interests are in the area of data center networks, Internet video, and distributed machine learning.

Wenjun Lyu received the B.E. and M.E. degrees in computer science and technology from Central South University in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree with Rutgers University. His research interests include data center networks and cyber-physical systems.



Jianxin Wang received the BEng and MEng degrees in computer engineering from Central South University, China, in 1992 and 1996, respectively, and the PhD degree in computer science from Central South University, China, in 2001. He is the chair of and a professor in the School of Computer Science and Engineering, Central South University, Changsha, Hunan, P.R. China. His current research interests include algorithm analysis and optimization, parameterized algorithm, bioinformatics, and computer network. He is a senior member of IEEE.