

Article

Deep Reinforcement Learning-Based Approach for Video Streaming: Dynamic Adaptive Video Streaming over HTTP

Naima Souane ^{1,*} , Malika Bourenane ^{1,*} and Yassine Douga ²¹ LRIIR Laboratory, Department of Computer Science, Université Oran 1, Oran 31000, Algeria² Department of Computer Science, University of Blida 1, Blida 09000, Algeria; yassine.douga@univ-blida.dz

* Correspondence: souane.naima@edu.univ-oran1.dz (N.S.); bourenane.malika@univ-oran1.dz (M.B.)

Abstract: Dynamic adaptive video streaming over HTTP (DASH) plays a crucial role in delivering video across networks. Traditional adaptive bitrate (ABR) algorithms adjust video segment quality based on network conditions and buffer occupancy. However, these algorithms rely on fixed rules, making it challenging to achieve optimal decisions considering the overall context. In this paper, we propose a novel deep-reinforcement-learning-based approach for DASH streaming, with the primary focus of maintaining consistent perceived video quality throughout the streaming session to enhance user experience. Our approach optimizes quality of experience (QoE) by dynamically controlling the quality distance factor between consecutive video segments. We evaluate our approach through a comprehensive simulation model encompassing diverse wireless network environments and various video sequences. We also conduct a comparative analysis with state-of-the-art methods. The experimental results demonstrate significant improvements in QoE, ensuring users enjoy stable, high-quality video streaming sessions.

Keywords: DASH; video streaming; wireless networks; QoE; deep learning; ABR; reinforcement learning algorithms; deep reinforcement learning; bandwidth estimation



Citation: Souane, N.; Bourenane, M.; Douga, Y. Deep Reinforcement Learning-Based Approach for Video Streaming: Dynamic Adaptive Video Streaming over HTTP. *Appl. Sci.* **2023**, *13*, 11697. <https://doi.org/10.3390/app132111697>

Academic Editor: Cosimo Nardi

Received: 17 August 2023

Revised: 24 September 2023

Accepted: 27 September 2023

Published: 26 October 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Maximizing the quality of experience (QoE) for users of video streaming services over wireless networks is a prominent challenge for researchers and video providers. Multimedia content, particularly videos, occupies a substantial portion of internet traffic [1], making it of utmost importance to address this challenge.

The dynamic adaptive streaming over HTTP (DASH) standard plays a vital role in selecting the appropriate quality of video segments based on network conditions, even over networks with limited bandwidth. However, DASH streaming can be susceptible to quality variations and rebuffering events, especially in challenging network conditions. To enhance user QoE [2], service providers and academic researchers have proposed several approaches aimed at achieving specific objectives, such as minimizing initial delays, preventing interruptions, and ensuring consistently high quality throughout streaming sessions.

In the DASH server, videos are divided into fixed-duration chunks or segments, typically ranging from 2 to 10 s. These segments are encoded in different qualities, and their details, including URLs, are stored in a Media Presentation Description (MPD) file. The DASH client is responsible for deciding the quality of the next segment to be downloaded, taking into consideration factors such as current network conditions, buffer occupancy, and device capabilities [3].

Furthermore, during a streaming session, the segments can be classified into four states: played, playing, in buffer, and to be downloaded. The buffer can contain segments of varying qualities, including low, medium, and high.

The design of an optimal adaptive bitrate (ABR) algorithm for dynamic adaptive streaming over HTTP (DASH) video streaming in highly dynamic environments presents

a significant challenge [4]. On one hand, fixed-rule-based ABR algorithms struggle to adapt to the varying conditions encountered during streaming. To address this, many ABR schemes have integrated artificial intelligence (AI) technologies to enhance user QoE. However, ensuring high user QoE in diverse network environments and client equipment can be challenging. These ABR schemes need to improve perceived QoE, minimize quality switches, prevent interruptions, and reduce initial delays.

On the other hand, artificial intelligence technology, especially deep reinforcement learning (DRL), indeed provides robust techniques, with promising applications in various domains [5–9], including video streaming. In this subsection, we have identified several potential challenges associated with applying DRL in the context of DASH:

- **Optimizing QoE:** One significant challenge lies in how DRL effectively manages the trade-off between delivering high-quality video and ensuring a seamless streaming experience.
- **Adaptation to Dynamic Network Conditions:** DASH operates in environments characterized by constantly changing network parameters, such as bandwidth and latency. It is essential to explore how DRL models adapt to these dynamic fluctuations and make real-time decisions to maintain optimal streaming quality.
- **Segmentation and Quality Switching:** The process of segmenting video content and determining when and how to switch between different quality levels is intricate. Discussing these intricacies sheds light on the challenges involved in ensuring a smooth transition between video segments.
- **Reward Function Design:** Designing a reward function that accurately reflects the quality of the streaming experience and encourages optimal decision making is a complex task. Elaborating on this complexity can provide insights into the inner workings of DRL models.
- **Training Complexity:** Training DRL models for video streaming poses computational challenges. This includes the need for substantial datasets and considerable computational resources. Exploring these training complexities highlights the practical considerations involved in implementing DRL for video streaming applications.

Existing approaches to DASH video streaming typically rely on heuristics or machine learning models to make bitrate selection decisions. However, these approaches often have limited performance and are not able to adapt to dynamic network conditions effectively.

In this paper, we propose a novel ABR algorithm based on deep reinforcement learning for determining the quality of the next segment in DASH video streaming over wireless networks. Our approach takes into account three key parameters: network conditions, buffer state, and the distance factor between consecutive video segments. The primary objective of our proposal is to provide a high-quality user experience.

The main contributions of our work are as follows:

- **Formulation and design model:** We introduce a new deep reinforcement learning approach for DASH video streaming that focuses on controlling the quality distance between consecutive segments. By doing so, we effectively manage the perceptual quality switch. We formulate the DASH video streaming process within a learning model called the Markov Decision Process, enabling the determination of optimal solutions through reinforcement learning.
- **Optimizing QoE:** To optimize the QoE, we define the QoE function that relies on three key parameters: the quality of each segment, the perceptual change in quality, and playback interruptions.
- **Reward Function Design:** We have designed and formulated a reward function that takes into account the rate of perceived quality change, occurrences of rebuffering events, and the quality of each video segment.
- **Analysis and implementation:** We classify the available qualities (bitrates) into three classes: Q_{high} , Q_{medium} , and Q_{poor} . To evaluate our approach, we conducted experiments using the animation video sequence “Big Buck Bunny” in the DASH.js

environment over a wireless network. The experiments involved playing video sequences on different devices under various network conditions. We monitored the performance of the experiments throughout the streaming sessions, specifically observing the perceptible quality switch. Based on the observed quality switches, we classified the bitrates into three distinct classes.

- **Training process:** To tackle the training challenge of DRL and achieve enhanced performance, we leveraged a comprehensive and diverse dataset, and we trained the DRL agent using a two-step training process.
- **Simulation and comparison:** We simulated and compared our proposed approach with existing studies. The results demonstrated a significant improvement in QoE, providing highly stable video quality. Our scheme successfully minimized the distance factor, ensuring a smooth streaming session.

Our proposal enhances video QoE using deep reinforcement learning by employing the following techniques: Firstly, we categorize video quality into three distinct classes and define the distance factor between two consecutive video segments. Secondly, we treat client-side adaptation as an autonomous agent. This agent is represented by a deep neural network, which generates policies for making decisions. This network includes an LSTM neural network component responsible for determining the optimal quality for the next video segment and receiving rewards iteratively until all segments have been downloaded.

The structure of this study's remaining sections is as follows: Section 2 offers an overview of related work, analyzing previous research in the field. Section 3 provides a comprehensive description of our proposed solution, delineating the methodology and approach used in the development of our innovative ABR algorithm based on deep reinforcement learning. In Section 4 showcases the experimental results derived from our system's evaluation, engaging in a discussion of the performance and outcomes of our approach. Finally, Section 5 concludes the paper by summarizing the key findings, emphasizing our work's contributions, and exploring potential avenues for future research in this domain.

2. Related Work

Adaptive video streaming has seen significant advancements in recent years, resulting in the introduction of various approaches. These approaches can be classified into four primary categories based on the adaptation technology employed:

- **Traditional ABR-based approaches:** This category encompasses methods that rely on bandwidth measurements, buffer occupancy, or a combination of both to guide streaming decisions. These approaches typically use fixed rules for adaptation throughout the streaming process. While they have found widespread use and implementation, their efficacy may be constrained, particularly in highly dynamic and diverse network conditions.
- **Deep-learning-based approaches:** Within this category, deep learning techniques, specifically neural network models, play a pivotal role. By training these models on extensive datasets, they can discern intricate patterns and make informed decisions regarding adaptation. Deep-learning-based approaches have demonstrated enhanced performance in adapting to a range of network conditions and user preferences. However, they often necessitate substantial amounts of training data and computational resources for effective model training.
- **Reinforcement-learning-based approaches:** In this category, adaptation decisions are orchestrated by an agent within an interactive environment, refining its choices through trial and error, guided by rewards. Reinforcement learning empowers the agent to learn and optimize its decisions based on the feedback received. These approaches offer the advantage of adaptability and the capability to navigate dynamic and uncertain network conditions. However, it is worth noting that training rein-

forcement learning models can be a time-consuming process, and their performance heavily depends on the design of rewards and exploration strategies.

- Deep-reinforcement-learning-based approaches: This category harnesses the synergy of deep neural networks and reinforcement learning techniques. Deep reinforcement learning approaches utilize deep neural networks to approximate various elements of the reinforcement learning process, equipping them to handle intricate streaming environments and make sound adaptation decisions. By leveraging the representation learning capabilities of deep neural networks, these approaches have exhibited promising results in delivering high-quality video streaming experiences.

2.1. Traditional ABR-Based Approaches

Many ABR algorithms have been proposed for DASH, including BBA [10], BOLA [11], ELASTIC [12], MPC [13], ABMA+ [14], Festive [15], PANDA [16], and Pensieve [17]. These ABR algorithms are implemented on the client side of DASH and aim to optimize video streaming by selecting the highest possible bitrates while minimizing rebuffering. However, these ABR algorithms often rely on fixed rules for bitrate selection and streaming orchestration, which can be less effective in highly dynamic environments. In such conditions, accurately characterizing and predicting bandwidth and making decisions that consider the overall environment can be challenging.

2.2. Deep-Learning-Based Approaches

Luca et al. [18] introduced ERUDITE, a deep neural network approach aimed at optimizing adaptive video streaming controllers to adapt to various parameters, such as video content and bandwidth traces. ERUDITE employs a deep neural network (DNN) as a supervisor to dynamically adjust the controller parameters based on observed bandwidth traces and resulting quality of experience (QoE). In [19], the authors proposed a deep learning approach to predict the qualities of video segments that will be cached, with the goal of enhancing the performance of DASH in a software-defined networking (SDN) domain. They utilized a deep recurrent neural network (LSTM) with three hidden layers to forecast future segments and incorporated SDN technology in their approach. However, when poor bandwidth conditions arise, it can lead to difficulties in downloading segments in real time, which may affect the accuracy of predictions. In [20], the authors presented a video QoE metric for DASH utilizing a combination of a three-dimensional convolutional neural network (3D CNN) and LSTM to extract deep spatial-temporal features that represent the video content. They also incorporated network transmission factors that influence QoE, following the DASH standard. By combining the extracted features, they generated an input parameter vector and used the ridge regression method to establish a mapping between the input parameter vector and the mean opinion score (MOS) value, predicting the final QoE value.

These studies collectively showcase the application of deep learning techniques, including deep neural networks and recurrent neural networks, in optimizing various aspects of adaptive video streaming to enhance performance and improve the user experience. However, none of these studies consider the perceived change in quality, which can also negatively impact QoE.

2.3. Reinforcement-Learning-Based Approaches

Hongzi et al. [21] developed an adaptive bitrate reinforcement learning (ABRL) system for Facebook's web-based video platform. The ABRL system generates RL-based ABR policies and implements them in the production environment. They established a simulator to train the ABR agent using RL techniques, and then deployed the translated ABRL policies to the user interface. Similarly, Jun et al. [22] proposed an enhanced ABR approach known as 360SRL (sequential reinforcement learning) to reduce the decision space and improve QoE. The 360SRL method refines the ABR policy by optimizing the QoE value or reward, which they define as a weighted sum of video quality, rebuffering, and spatial and temporal

video quality variance. Throughout their training process, they used the Deep Q-Network (DQN) method.

These studies highlight the utility of reinforcement learning techniques, such as RL and DQN, in designing and enhancing ABR systems for web-based video platforms. Their primary objective is to optimize ABR policies to improve user QoE, taking into account factors like video quality, rebuffering, and temporal/spatial video quality variance. Nonetheless, it is worth noting that there may be additional factors to consider that have not been addressed in these studies.

2.4. Deep-Reinforcement-Learning-Based Approaches

In recent years, several studies have been conducted to enhance the QoE in video streaming by leveraging DRL techniques. These studies have explored a variety of neural network architectures and reinforcement learning algorithms to optimize adaptive video streaming and improve user satisfaction. This section provides a comprehensive overview of the relevant literature in this field.

Anirban et al. [23] introduced the LASH model, a deep-neural-network-based approach that combines LSTM, CNN, and reinforcement learning for adaptive video streaming. Their primary objective was to maximize QoE by considering perceived video quality, buffering events, and video smoothness. However, the specific neural network architectures and reinforcement learning algorithms used were not explicitly mentioned.

Lu et al. [24] developed an end-to-end unified deep neural network (DNN) architecture to predict both the quality of service (QoS) and QoE values in wireless video streaming sessions. Their framework featured two hidden layers and used the deep Q-network (DQN) algorithm for bitrate adjustments. The primary focus was on deriving an optimal bitrate adaptation policy to maximize rewards during video streaming.

Dong et al. [25] proposed an online and end-to-end DRL-based policy for power allocation in video streaming. They utilized the deep deterministic policy gradient (DDPG) algorithm and integrated safety layers, post-decision states, and virtual experiences to ensure quality of service and expedite convergence.

Matteo et al. [26] conceived the D-DASH framework, which leveraged a combination of multi-layer perceptron (MLP) and LSTM neural networks within a deep Q-learning environment to optimize QoE in DASH. Their reward function incorporated the structural similarity index (SSIM) and considered video quality switches and rebuffering events.

Tianchi et al. [27] proposed a video quality-aware rate control approach aimed at achieving high perceptual quality with low latency. Their method employed deep reinforcement learning to train a neural network for selecting future bitrates. To simplify the reinforcement learning model, they divided it into two models: the video quality prediction network (VQPN) and the video quality reinforcement learning (VQRL) model. The VQPN model predicted the next video quality, while VQRL trained the neural network.

Zhao et al. [28] introduced the Deeplive model, which used two neural networks to make decisions about bitrate, target buffer, and latency limits in live video streaming. The agent selected the higher reward based on a reward function that considered frame video quality, rebuffering, latency, frame skipping, and bitrate switches.

Gongwei et al. [29] presented the DeepVR approach for predictive panoramic video streaming using deep reinforcement learning techniques. Their method used LSTM to predict users' field of view (FoV) in the future seconds, and the DRL agent selected the quality level based on a reward function that incorporated FoV quality, rebuffering time, video quality change, and buffer level. They employed the Rainbow training method to train the DRL agent.

Furthermore, other studies have concentrated on optimizing specific facets of video streaming. For instance, in [17], an HTTP adaptive streaming framework integrated online reinforcement learning, where the neural network was trained using the Pensieve reinforcement learning algorithm. The primary aim was to enhance the adaptive bitrate (ABR) model when quality of experience (QoE) degradation occurred.

Ling yun Lu et al. utilized the actor–critic (A3C) algorithm for adaptive streaming based on DRL in [30]. Their approach determined the optimal bitrate based on user preferences, network throughput, and buffer occupancy. Similarly, in [31], Omar Houidi et al. adapted the actor–critic architecture for deep reinforcement learning to account for QoS and network traffic management, ultimately maximizing QoE.

ALVS [32] introduced a deep reinforcement learning framework for video live streaming, where adaptive playback speed and video quality level were selected for the next segment to enhance QoE. Additionally, GreenABR [33] proposed a deep-reinforcement-learning-based ABR scheme with the objective of optimizing energy consumption during streaming sessions without compromising QoE. The authors in [34] proposed a deep reinforcement learning approach aimed at enhancing the QoE in video applications. This was achieved by dynamically adjusting IEEE 802.11 parameters to improve network conditions and maintain a higher QoE for users. However, it is important to note that this approach employed traditional ABR with fixed rules in a dynamic environment. Addressing the need for effective bitrate adaptation in mobile video streaming, [35] introduced DQNReg, an advanced deep reinforcement learning approach that optimizes QoE. DQNReg outperforms other methods, displaying faster convergence, higher QoE scores, reduced rebuffering, and superior video quality maintenance. In [36], a proximal-policy-optimization-based DRL (PPO-ABR) approach was presented for adaptive bit rate streaming. This leverages DRL to maximize sample efficiency and elevate video QoE. The authors tackle the challenge of ensuring high QoE in video streaming within dynamic 5G and beyond 5G (B5G) networks, underscoring the limitations of existing ABR algorithms.

Several DRL-based approaches and frameworks have been presented in the literature, some of which employ intricate neural network architectures, often combining multiple neural networks. In contrast to the cited works, we proposed a novel DRL-based approach utilizing an LSTM neural network that takes into account the holistic environment when determining the next bitrate. In our DRL approach, a single deep neural network approximates the policy. We have incorporated a novel factor known as “distance” to aid in bitrate selection, all in the pursuit of enhancing the QoE. Our primary objective is to deliver optimal dynamic adaptation in highly complex and dynamic environments.

3. Materials and Methods

3.1. Problem Formulation

The deep reinforcement learning algorithm is concerned with how agents make decisions to achieve a goal or maximize a specific objective over multiple steps in a complex environment, ultimately accumulating the highest cumulative rewards. The DRL algorithm excels at making sequential decisions within defined time intervals, making it valuable for solving intricate problems. In reinforcement learning, two fundamental learning models are the Markov Decision Process and Q-learning. In this paper, we frame the challenge of streaming video DASH in the context of the Markov Decision Process (MDP) learning model to derive reinforcement learning solutions. Reinforcement learning comprises five key components: the agent, the environment, the states, the actions, and the rewards. The agent takes a series of sequential actions based on the current state of the environment. After each action, the environment moves to a new state, and in response, the agent receives a reward. The ultimate objective of the agent is to maximize the total cumulative rewards within a specific number of steps.

3.2. System Model

This section describes the system model underlying our proposed approach. In this deep reinforcement learning environment, the focus is on the DASH system. Within this DASH system, the DASH server makes available a collection of videos represented as $V \{v_1, v_2 \dots v_N\}$. Each of these videos is segmented into various bitrates or quality levels, denoted as $Q \{q_1, q_2, q_3 \dots q_M\}$, with each segment having a fixed duration. Furthermore, each video is accompanied by a media presentation description (MPD) file containing

comprehensive information about these segments. At the inception of a video streaming session, the DASH client kicks off the process by requesting and subsequently analyzing the MPD file. This analysis serves as the foundation for initializing the streaming process, aligning it with the current network conditions and the capabilities of the user's device. Within this framework, we conceptualize the client-side adaptation as an autonomous agent tasked with determining the most suitable bitrate for each segment based on an optimal policy. Upon downloading the initial segment, this agent actively monitors the environment. This environment encompasses critical parameters such as bandwidth measurements, buffer status, and the quality (bitrate) of the previously downloaded segment. The agent leverages this environmental information to make informed decisions about the optimal bitrate for the next segment. This iterative process continues as the agent takes actions and receives rewards, culminating when all segments have been successfully downloaded.

We employ the Markov Decision Process (MDP) learning model to formulate the optimal bitrate adaptation strategy, leveraging the distance factor between two consecutive segments. Our objective is to ensure a high level of user satisfaction, denoted as $S = \{s_1, s_2, s_3, \dots\}$, which is defined by the avoidance of playback interruptions and the provision of a consistently high-quality video streaming experience. In this context, the variable S represents the function of playback interruption avoidance and the maintenance of stable, high-quality streaming, denoted as follows:

$$S = f(\text{playback}_{\text{interruption}}, \text{stable_high_quality}), \quad (1)$$

3.3. Reward Function

This research endeavors to ensure consistently high perceptual quality and maximize user satisfaction while eliminating rebuffering. For that, we have formulated a reward function $R = f(r, rb, q)$ that takes into account the rate of perceived quality change (r), the occurrences of rebuffering (rb), and the quality of each video segment (q).

In this subsection, we describe the experimental study conducted to classify the various bitrates (qualities). For this experiment, we used the BigBuckBunny video sequence from the BigBuckBunny dataset [37]. Users participated in the tests by initialing their video sequences on different devices across various internet conditions, all within the context of an ongoing streaming session. Moreover, we programmed bitrate switches in the following manner: $\{(144p, 240p), (144p, 360p), (144p, 720p), \dots\}$. We meticulously observed the resulting changes in perceived quality.

Based on the empirical findings, we categorized the qualities into three distinct classes: Q_{high} , Q_{medium} and Q_{poor} . These are visually depicted in Figure 1. To define these classes, we establish the following quality sets: $Q_{\text{high}} = \{1080p, \dots, 2160p\}$ representing high quality, $Q_{\text{medium}} = \{360p, \dots, 720p\}$ representing medium quality, and $Q_{\text{poor}} = \{144p, \dots, 240p\}$ representing poor quality. The quality level qt of a given segment s_i can be classified into the quality set $Q = \{Q_{\text{high}}, Q_{\text{medium}}, Q_{\text{poor}}\}$, while the bandwidth can be categorized into the set $Bw = \{Bw_{\text{low}}, Bw_{\text{medium}}, Bw_{\text{high}}\}$.



Figure 1. Classification of quality levels.

3.3.1. Perceived Quality Change

Within this proposal, we introduce a redefined perspective on the rate of quality change. In this context, we acknowledge a perceived quality change when the user detects a switch, implying that a shift in quality within the same quality set is not regarded as a perceived quality change. However, we do recognize a change when the quality level q_t transitions from the poor-quality set to the high-quality set q_{t+1} . To quantify this transition, we introduce a distance factor denoted as Δfact , which quantifies the disparity between two quality levels, as illustrated in Table 1.

Table 1. Distance factor values between quality levels.

Quality	High Quality	Medium Quality	Poor Quality
High quality	0	−1	−2
Medium quality	1	0	−1
Poor quality	2	1	0

According to Table 1, the $\Delta\text{fact}_{q_t}^{q_{t+1}} = |1|$, where q_t corresponds to medium quality, and q_{t+1} corresponds to high quality.

3.3.2. Rebuffering

Rebuffering Rb that occurs subsequent to the download of each segment can be measured by comparing the playback time of a downloaded segment sp and the completion time of the download sd . A rebuffering event occurs when sd surpasses sp , signifying a pause or interruption in video playback. Consequently, rebuffering Rb can be defined as the cumulative count of video interruptions.

$$Rb = \sum_{k=1}^N Rb_k, \quad (2)$$

3.3.3. Segment Quality

Segment quality signifies the quality level of each downloaded segment, denoted as q_i . Consequently, the average quality (Avg_q) is computed as the summation of all segment qualities (q_k) divided by the total number of segments (N), as expressed below:

$$Avg_q = \frac{\sum_{k=1}^N q_k}{N}, \quad (3)$$

Here, N signifies the total count of segments.

3.3.4. QoE Function

In this section, we highlight the QoE function. As previously delineated, QoE relies on three key parameters: the quality of each segment, the perceptual change in quality, and rebuffering or playback interruptions. The agent receives a reward for each decision, leading to the following function:

$$r_t = f(q_t, q_{t-1}), \quad (4)$$

Consequently, the QoE can be articulated as:

$$QoE_{max} = \sum_{k=1}^K (r_k) - \sum_{k=1}^K |\mu Rb_k| - \sum_{k=1}^{K-1} |\lambda(\Delta\text{fact}_k)|, \quad (5)$$

where μ and λ represent the penalties attributed to rebuffering and changes in quality, respectively.

We normalize the QoE_{value} to align it with the state of the art [17,24,29,38], resulting in the following expression:

$$QoE = \frac{QoE_{max}}{N} * 100, \quad (6)$$

Here, N denotes the total count of video segments.

3.4. Markov Decision Process (MDP)

Client-side adaptation necessitates making sequential decisions over time denoted as t . To formalize this adaptation process, we utilize a Markov Decision Process (MDP). The MDP is characterized by a quintuple of fundamental components: agent, environment, states, actions, and rewards, as illustrated in Figure 2.

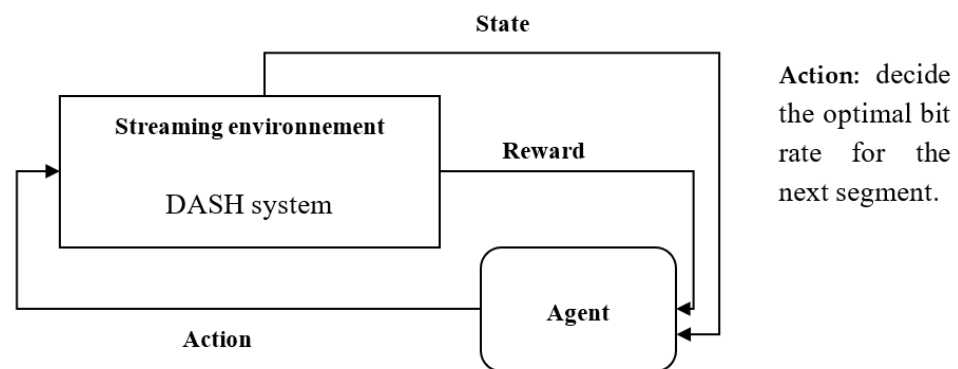


Figure 2. Reinforcement learning architecture.

The agent embodies client-side adaptation responsibilities, with its primary task being to select an action denoted as a_t . This action entails the choice of an appropriate bitrate or quality q_t for the next subsequent segment S_t awaiting download. The agent's decision is based on current bandwidth measurements Bw_t , the prevailing buffer state $buff_state_t$, and the quality q_{t-1} of the previous segment s_{i-1} . The agent's bitrate selection (adaptation) results in the receipt of rewards or penalties.

In the realm of adaptive video streaming, we define segments as a finite set of states, $S = \{s_0, s_1, \dots, s_N\}$. It is important to note that a video is divided into N segments. Simultaneously, we establish a set of actions as $\{a_t\}$, with t taking values from $\{0, 1, 2, \dots, N\}$, signifying the decision made at step t . The duration of each step, t , is determined by the download time of the segment. The action set for a specific state is denoted as $A(s) = \{A_{s1}, A_{s2}, A_{s3}\}$. Here, A_{s1} characterizes the scenario when q_t and q_{t-1} reside in the same quality set, as defined previously. A_{s2} pertains to cases where q_t and q_{t-1} do not belong to the same quality set but are the closest in quality levels. A_{s3} represents instances where q_t falls within a lower quality set. State transitions at each step are defined as $s_t = \{q_{t-1}, q_t, buff_state_t, Bw_t\}$, encompassing variables such as q_{t-1} denoting the quality of the preceding segment, q_t signifying the quality of the downloaded segment, Bw_t representing the current bandwidth, and $buff_state_t$ indicating the current buffer state. To ascertain the quality of the subsequent segment, the agent utilizes a policy π as an approximation strategy. In this work, we incorporate a deep neural network (DNN) to approximate the policy $\pi(a|s; \theta)$, responsible for mapping the current state S to the next state S' . The agent is encapsulated within a deep neural network, which generates the policy for decision making, while θ denotes the weights within the deep neural network.

The state transition at time t is denoted as $S = (Bw, buff_state_t, q_t)$ after implementing the policy $\pi(a|s; \theta)$. The selected action is $a_t = A_{s1}$. The resulting state is:

$$S' = (Bw', buff_state'_t, q'_t), \quad (7)$$

where the following are true:

$$q'_t = \Delta q_t \text{ if } (q_t, q_{t-1}) \text{ are in the same quality set.}$$

$$Bw_t' = \Delta Bw_t$$

$$buff_state_t' = \Delta buff_state_t$$

A reward function $R_i(S_t)$ is received after downloading each segment S_t . The reward is as follows:

$$r_t = R(S_t = S) = f(q_t, q_{t-1}), \quad (8)$$

The following Table 2 illustrates the rewards received after each action taken by the agent:

Table 2. Rewards defined for each state.

$S_t = S$	$R(S_t = S)$
$(q_t, q_{t-1}) \in Q$ // q_t and q_{t-1} are in the same quality set	1
$(q_t) \in Q_{high}$ // q_t is in the high-quality set	0.75
$(q_t) \in Q_{medium}$ // q_t is in the medium-quality set	0.50
$(q_t) \in Q_{poor}$ // q_t is in the poor-quality set	−1
$R_{bt} = 0$ // No rebuffering	1
$R_{bt} > 0$ // rebuffering event occurs	−1
$(q_t, q_{t-1}) \notin Q$ // q_t and q_{t-1} are not in the same quality set	−1

3.5. Deep Neural Network Architecture

A typical deep neural network (DNN) consists of an input layer, multiple hidden layers, and an output layer. Each layer consists of neurons that are activated based on input from the previous layer using an activation function. In this paper, we use the long-short-term memory network (LSTM) [39]. LSTM stands out as a powerful variant of recurrent neural networks (RNNs) that is widely recognized for its capabilities in processing and predicting sequence data, with a specific emphasis on time-series data. The LSTM unit encompasses four essential components, as visually depicted in Figure 3. These components include a specialized cell that is designed to store and retain information over extended periods. Furthermore, the cell incorporates three important gates: the input gate, the output gate, and the forget gate. These gates serve as vital mechanisms for controlling the flow of information into and out of the cell. In our specific context, we train the LSTM to intelligently preserve quality information from previous segments while disregarding historical details such as bandwidth and buffer state information.

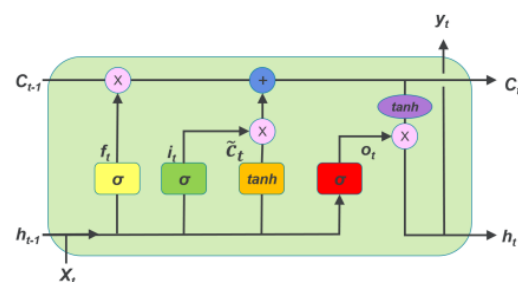


Figure 3. LSTM cell architecture [40].

3.5.1. Agent Design

The agent is constructed using an LSTM neural network, which makes decisions based on the current bandwidth, buffer state, and the quality of the previously downloaded segment. The LSTM takes the state $S_t = (q_t, Bw_t, Buft)$ as input and selects an action that corresponds to the next state $S_{t+1} = (q_{t+1}, Bw_t, Buft)$. Subsequently, the agent receives a reward or penalty based on its chosen action. If the agent receives a positive reward, it proceeds to the next state and continues making decisions. However, if the agent receives a negative reward, it sends an alert message to the secondary agent, allowing for potential corrective measures. The secondary agent then observes the environment to determine if a correction can be made without adversely impacting the QoE. Figure 4 illustrates the

architecture of the agent, where the output at time step $t - 1$ serves as the input for the subsequent decision at time step t .

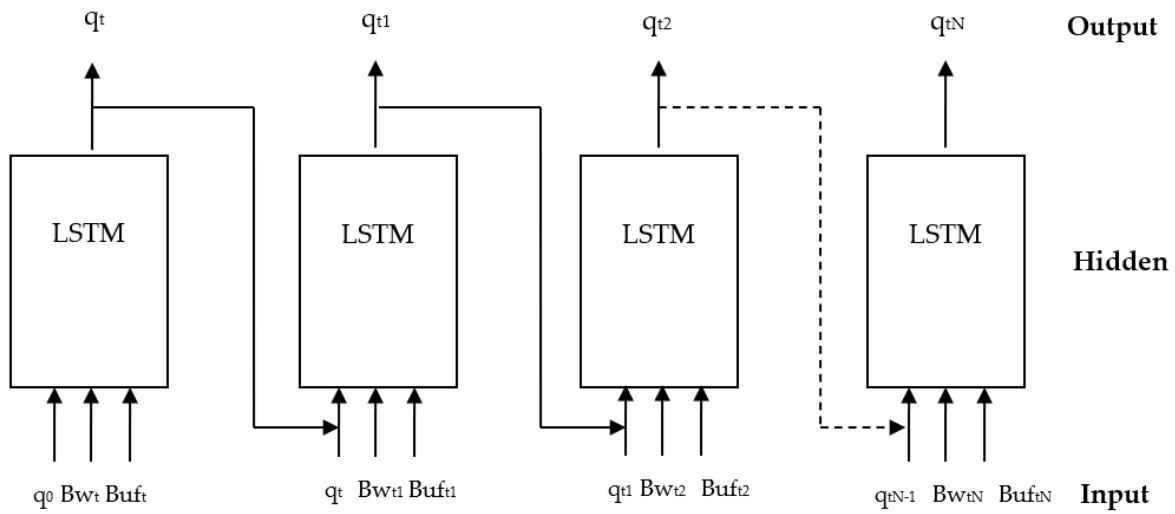


Figure 4. Agent architecture.

3.5.2. Agent Training

This section provides an in-depth explanation of the training process for our deep reinforcement learning approach. The agent is trained in diverse environments that encompass bandwidth statistics and videos encoded at different quality levels, with different numbers of segments.

During the offline phase, we start by individually training two identical agents: the main agent and the secondary agent. We use the rewards and penalties specified in Table 2 to train these agents. The primary goal is to minimize the factor distance between consecutive segments, prevent rebuffering events, and select high-quality segments for downloading.

In the subsequent phase of training, we deploy the two agents to stream content in different environments. While a streaming session is underway, the main agent makes decisions and receives rewards or penalties, while the secondary agent remains in a passive mode. The secondary agent only reacts when it receives an alert message from the main agent, indicating a need to correct a decision.

In this paper, we use the policy gradient algorithm REINFORCE [41] to optimize the agent's policy (6) and update the policy parameter θ .

$$\pi_{\theta}(s, a) = P[a|s; \theta] \quad (9)$$

Equation (9) defines the policy function $\pi_{\theta}(s, a)$, a fundamental concept in reinforcement learning used to represent the probability of taking action 'a' in state 's'. In this equation, note the following:

- $\pi_{\theta}(s, a)$ represents the policy function, where π_{θ} is the policy, 's' is the state, and 'a' is the action. This function calculates the probability of taking action 'a' in state 's' according to the policy parameterized by θ .
- $P[a|s; \theta]$ indicates the probability of taking action 'a' given the state 's' under the policy π_{θ} .

The main objective is to maximize the accumulated reward. The cumulative discounted reward, denoted as R_t , is computed as the sum of discounted rewards over time, with γ as the discount factor and t as the time step:

$$R_t = \sum_t \gamma^t * r_t \quad (10)$$

Equation (10) calculates the cumulative discounted reward R_t by summing up the rewards obtained at each time step 't', with each reward being discounted by γ^t . This allows the agent to consider the long-term consequences of its actions, giving more weight to immediate rewards and gradually reducing the impact of future rewards as 't' increases.

The gradient of the discounted reward with respect to the parameter θ , denoted as ∇_{θ} , can be expressed as:

$$R_t = \nabla_{\theta} E \left[\sum_t \gamma^t * r_t \right] = E \left[\sum_t \gamma^t * r_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right] \quad (11)$$

Equation (11) describes the gradient of the discounted reward with respect to the policy parameter θ in the context of reinforcement learning. This equation is essential in policy gradient methods, which are used to optimize the agent's policy to maximize the expected rewards.

After each episode, the policy parameter θ is updated using the learning rate α as follows:

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_{\theta} \log (\pi_{\theta}(s_t, a_t)) (\sum_t r_t) \quad (12)$$

Equation (12) describes how the policy parameter θ is updated after each episode in a reinforcement learning setting. This equation represents the update rule used in the policy gradient algorithm REINFORCE.

3.6. Overall Functionality Block Diagram of the Proposed Method

Figure 5 illustrates the system model, providing an overview of the proposed method's overall functionality. The key components of the system are:

- **Agent:** The reinforcement learning agent, which uses the current state (bandwidth, buffer state, and quality of the previously downloaded segment) to select the next action to take (next video segment to download).
- **Environment:** The environment in which the agent operates, which includes the video stream and the network conditions.
- **Reward:** The reward that the agent receives for taking a particular action.

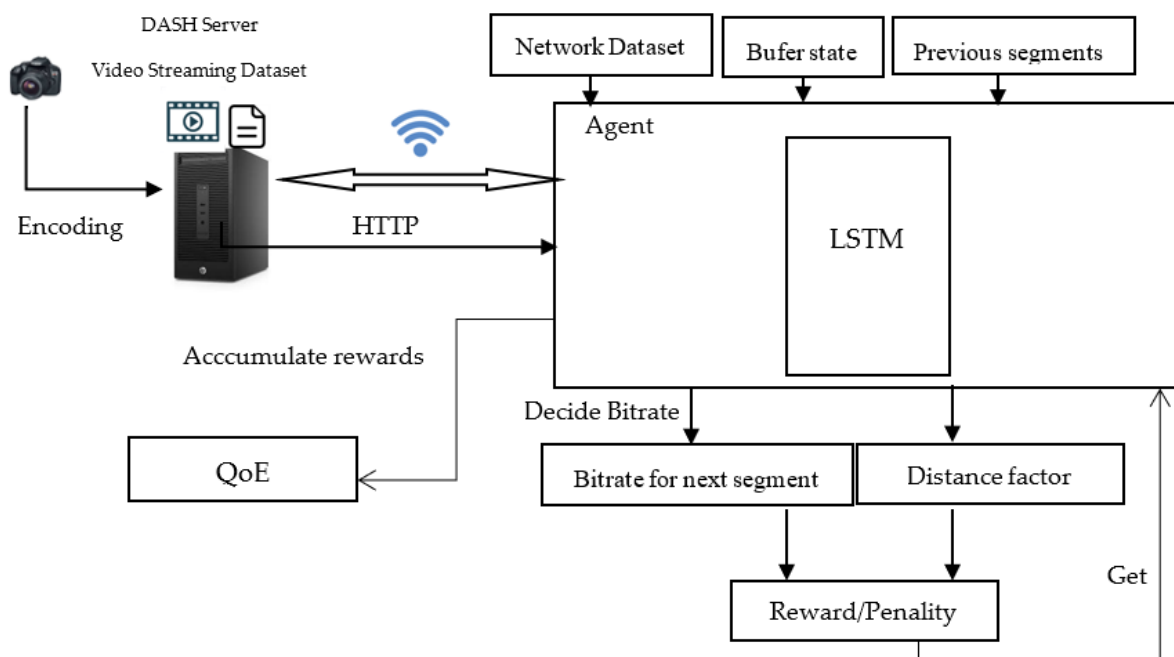


Figure 5. System model of the proposed method.

The agent uses the current state to select an action, which is then executed in the environment. The environment then returns a reward to the agent, which the agent uses to update its policy.

4. Performance Evaluation

In this section, we discuss the implementation environment, provide comprehensive details about the LSTM neural network, describe the experiment setup in detail, and outline the datasets used for both training and testing our model. Subsequently, we present the results of our approach and compare them to the state-of-the-art methods.

Implementation Details: Our simulation is implemented in python using the Keras API [42]. We adopted the neural network architecture for our agent, as shown in Figure 4. This architecture consists of a single hidden layer. We employed the TanH activation function, which maps the output values to the interval $[-1, 1]$. The number of iterations corresponds to the total number of video segments. In the initial phase, the model takes as inputs the quality of the first downloaded segment (determined based on network parameters), the current bandwidth, and the buffer state. It then computes and generates the quality of the next segment based on these inputs. This process continues iteratively, with the previous quality influencing the decision-making process until the last video segment is reached. The simulation parameter values used are illustrated in Table 3.

Table 3. Simulation parameter values.

Simulation Parameter	Description	Value
γ	Discount factor	0.99
λ	Penalty of the rebuffering	−1
μ	Penalties of the change of quality	−1

Datasets: In our simulation, we use two distinct datasets for different purposes:

- **Network Dataset:** This dataset is used to simulate network conditions, such as bandwidth, latency, and other network-related parameters. It contains data about network performance.
- **Video Streaming Dataset:** This dataset is used in a server-side component for video streaming, where video content is requested and streamed to users. It contains data about video files, user requests, and streaming performance.

Experiment setup: The proposed approach is validated using two crucial datasets: the SJTU HDR video sequences dataset [43], and the Network Dataset Norway [44]. Here are the specifics of these datasets and the experimental setup:

- The SJTU HDR video sequences dataset consists of 15 ultra-high-definition (UHD) video sequences with a resolution of 4K (2160 p). Each video sequence is divided into segments of 2 s each. A buffer size of 30 s is maintained during playback. The video sequences are encoded using the high-efficiency video coding (HEVC) format and have a frame rate of 30 frames per second (fps).
- Network Dataset Norway contains network traces collected from Telenor's 3G/HSDPA mobile wireless network in Norway. It accurately replicates real-world network conditions, capturing variations and characteristics commonly found in mobile networks.

Agent Training Steps:

The agent undergoes a two-step training process:

- **Step 1—Individual Agent Training (Offline Mode):** In this initial phase, the agent is trained alone using deep reinforcement learning (DRL). For this training phase, the datasets are randomly divided, with 70% of the video sequences from the video streaming dataset/Network Dataset Norway used for agent training, while the re-

maintaining 30% are reserved for testing the model's performance. This phase equips the agent with the ability to learn and make decisions based on the training data.

- **Step 2—Collaborative Agent Training (With a slave Agent):** In the second step, the agent collaborates with a secondary agent, which plays a crucial role in enhancing the quality of streaming sessions. The slave agent works in collaboration with the main agent and corrects any erroneous decisions made by the latter. This collaborative approach leads to improved adaptation and decision-making processes.

Through these experiments, we aim to evaluate the effectiveness of our proposed approach in handling real-world video streaming scenarios and optimizing the streamed content quality.

Experimental Results and Discussion: In this section, we present the experimental results and discuss our findings in detail. We begin by presenting the results of training the LSTM neural network, providing insights into its performance and learning progress during the training phase. Our evaluation encompasses critical metrics such as network convergence and loss, which allow us to assess its effectiveness in optimizing streaming quality. Next, we conduct a comparative analysis of our proposed model with the state-of-the-art approaches. These comparative experiments take place in a simulated environment with a diverse range of video sequences and different network traces. Our goal is to evaluate the robustness and generalization capabilities of our approach. We meticulously examine key performance indicators, such as video quality, buffer occupancy, rebuffering events, and user experience, providing a comprehensive evaluation of the effectiveness of our approach.

For a comprehensive evaluation, we first trained the neural agent network using 70% of the video sequences from the SJTU HDR video sequences dataset. Subsequently, we used the remaining 30% of the dataset videos for the testing process. This deliberate split allows us to effectively appraise the model's performance on unseen data, providing insights into its generalization capabilities beyond the training set.

The results presented here represent the average outcomes of all conducted experiments. This approach ensures both reliability and representative assessment of our approach's performance. Our intention in presenting these results is to demonstrate the effectiveness and potential advantages inherent in our proposed approach in optimizing video streaming quality and elevating user experience.

Figure 6 illustrates the degree of performance improvement achieved when using 70% of the video sequences from the dataset and how it converged with the amount of data. The graph showcases the progress of performance improvement as the training process advances, with the *x*-axis representing the percentage of data used for training. As depicted in the graph, performance improvement consistently increases as more of the dataset is used for training. However, it becomes evident that the rate of improvement starts to converge most clearly by about 60% of the dataset being used for training.

This graph provides insights into the effectiveness of our approach in improving performance. Moreover, it helps identify the saturation point, where additional training data may not significantly improve performance. This means that the agent is well trained and ready for deployment (streaming).

Figure 7 shows the degree of performance improvement achieved over time and when it converges. The graph showcases the evolution of performance improvement as the training process advances, with the *x*-axis representing the time in hours. This experiment was conducted using 70% of video sequences and Network Dataset Norway in real time. As shown in Figure 7, performance improvement consistently increases throughout the experiment and starts to converge after about 40 h. Therefore, we stopped the experiment in 60 h when the desired amount of data was achieved. Additionally, more training time may not significantly improve performance.

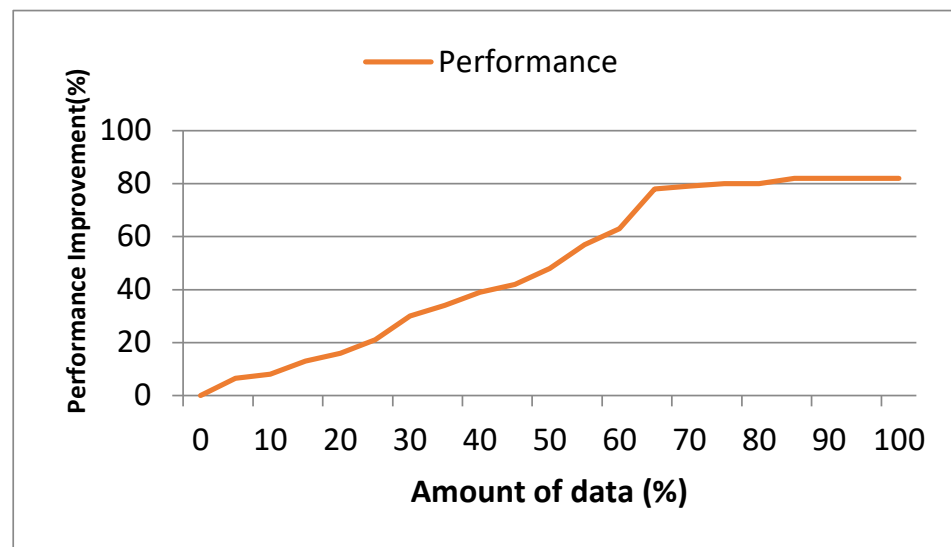


Figure 6. Degree of performance improvement with 70% of video sequences dataset.

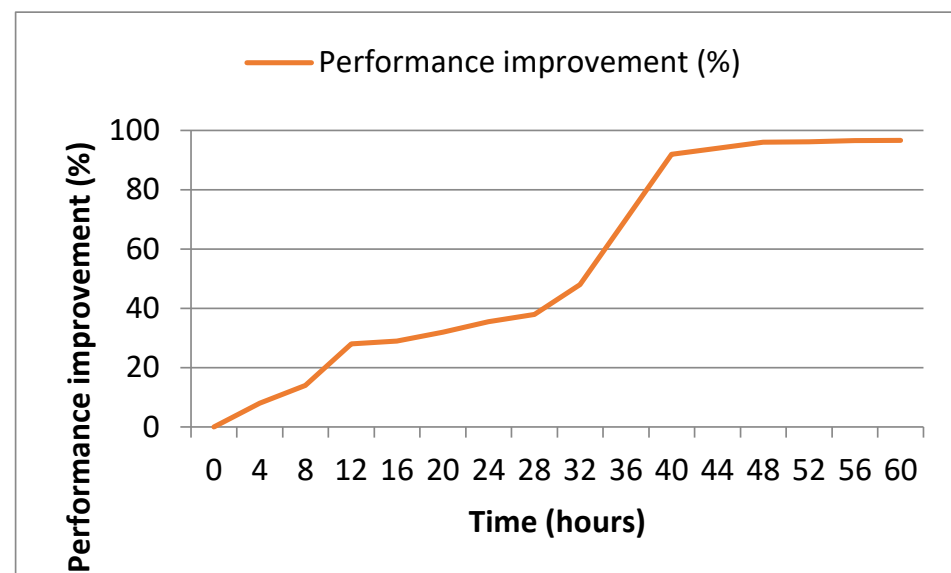


Figure 7. Degree of performance improvement over time.

Figure 8 shows valuable insights into the effectiveness of utilizing a secondary agent and its impact of performance improvement. It demonstrates the importance of collaborative approaches in enhancing the overall system's decision-making capabilities and optimizing performance.

In this experimental scenario, we conducted training for the main neural network agent while introducing a secondary agent into the training process to evaluate its impact on performance improvement. Throughout the duration of the experiment, the overall system continuously monitored the demand for intervention and promptly dispatched alert messages to the secondary agent in cases involving penalties or erroneous decisions. As a result, we can infer that the inclusion of the secondary agent led to increase performance improvement and reduced penalties. This collaborative interaction between the two agents proved to be very effective. During the experiment, a distinct trend emerged: there was a significant decline in the number of alert messages transmitted by the overall system after approximately 55% of the video sequences had been incorporated into the training process. This decline indicates that the collaboration between the main agent and the secondary agent had positively influenced the decision-making process. The end result was

a marked improvement in overall system performance, coupled with a reduced frequency of penalties or erroneous decisions.

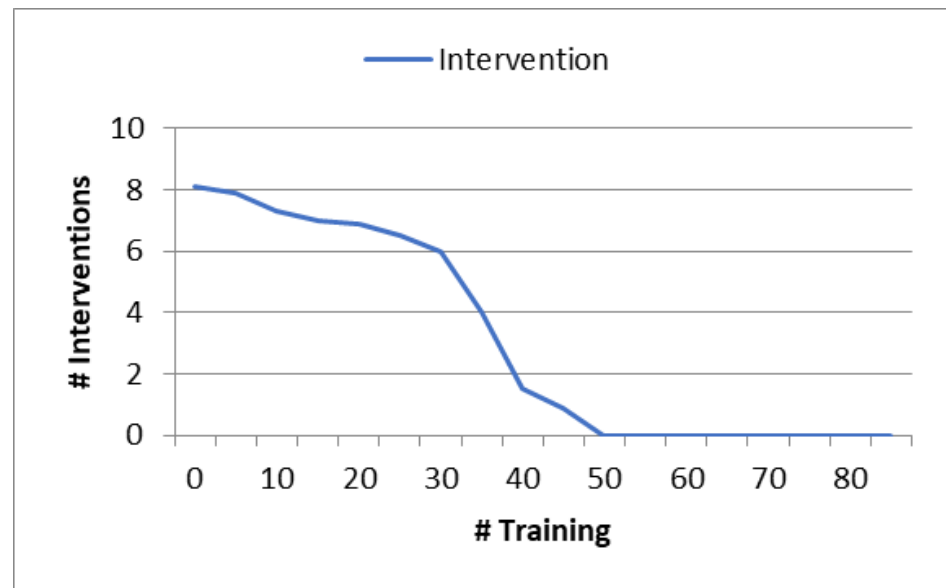


Figure 8. Degree of performance improvement with a secondary agent.

Figure 9 illustrates the results of a comprehensive assessment of various QoE metrics, including rebuffering events, average segment quality, and the overall QoE value. These metrics are pivotal indicators for evaluating the performance of our proposed approach.

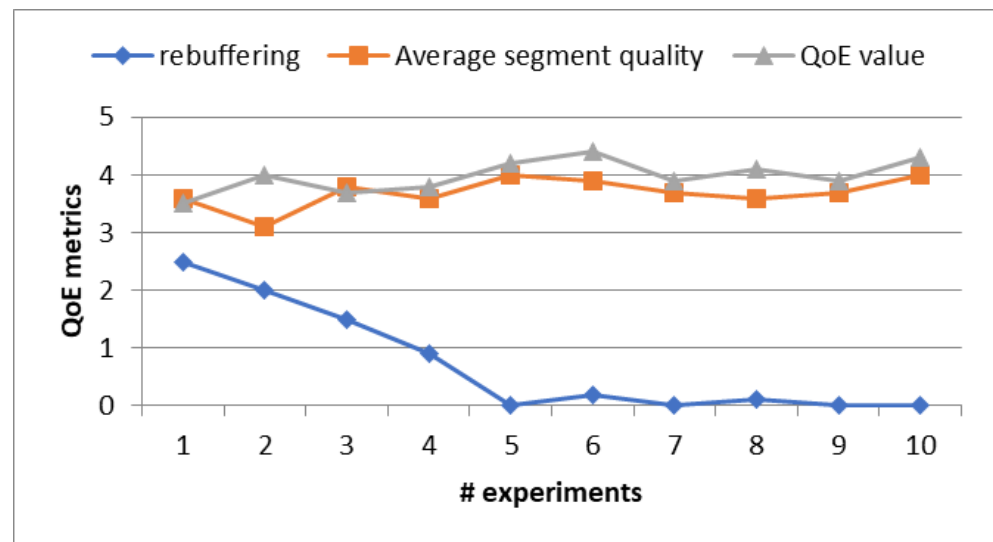


Figure 9. Results of QoE metrics.

This experiment was conducted using the remaining 30% of video sequences and Network Norway Datasets. As a result, it can be confirmed that the number of rebuffering events decreases linearly according to experiments. Also, we observed that there were no rebuffering events after five experiments. The average segment quality and QoE value are high across the experimental timeline. From the experimental results, it can be confirmed that the proposed approach has improved the overall QoE values using the video sequences dataset with a very wide range of wireless network conditions.

QoE assessment can be conducted through both subjective and objective methodologies. Subjective methods entail the collection of evaluations from individuals who assess

the service based on their personal satisfaction. In our endeavor to appraise QoE, we have embraced the mean opinion score (MOS) [45] as subjective evaluation method.

In this experiment, we invited 20 users to actively participate in the testing phase. Each user played back video sequences from a designated video dataset.

Figure 10 shows that the proposed solution achieved an impressive average MOS score of 4.5, which fall within the “good” range. This average MOS score signifies a high level of user satisfaction with the video streaming experience. The positive reception of our approach is a testament to its effectiveness in enhancing QoE for users.

In this experiment, we conducted a comprehensive comparison of our proposed approach with several existing schemes: Pensieve [17], the DASH framework with online reinforcement learning [38], LBC [24], and Rainbow [29]. Based on the comparative results, it is clear that our proposed approach substantially improves the QoE, with improvements exceeding 96% compared to the existing schemes. Moreover, the evaluation of QoE metrics showed that our approach significantly reduces average rebuffering time below 1.8 s and minimizes average quality switching to less than 0.21%. Furthermore, the results obtained from the subjective MOS method validate the effectiveness of our approach and highlight its potential to enhance the overall streaming experience. This collective evidence solidifies the superiority of our approach in delivering an enhanced user experience.

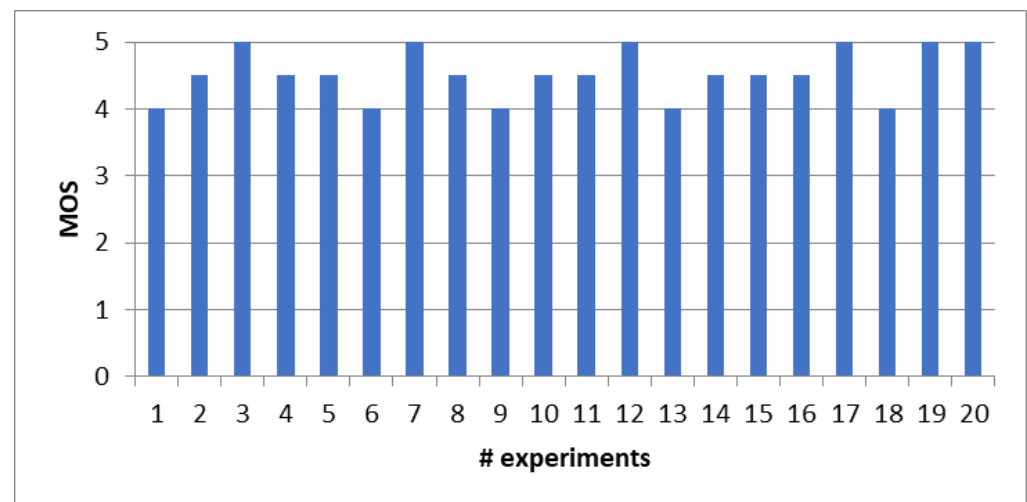
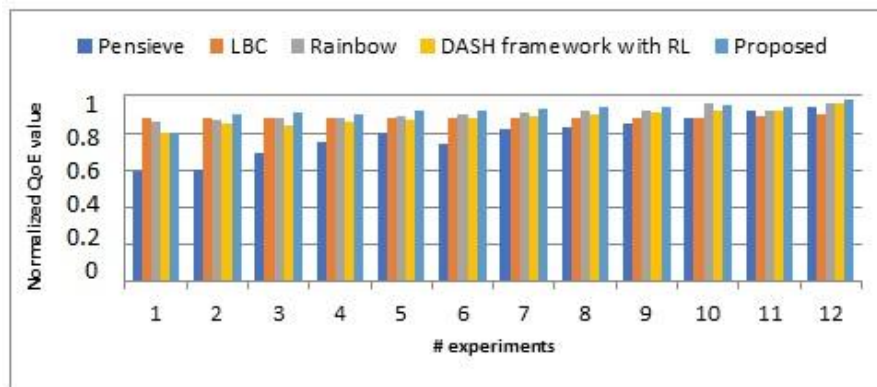


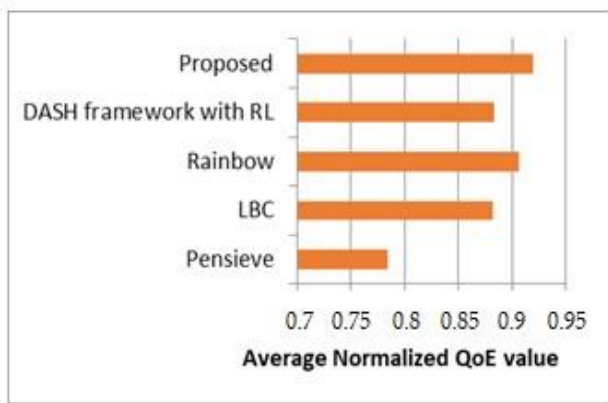
Figure 10. Subjective QoE test using the MOS method.

Figure 11 illustrates that our approach has the highest values in terms of normalized QoE value, average normalized QoE value, average switching quality, and average rebuffering time compared to the existing schemes, particularly, in low and fluctuating bandwidth conditions, the proposed approach and existing schemes showed high performance, but in medium and low bandwidth conditions, the proposed approach provided high and stable QoE.

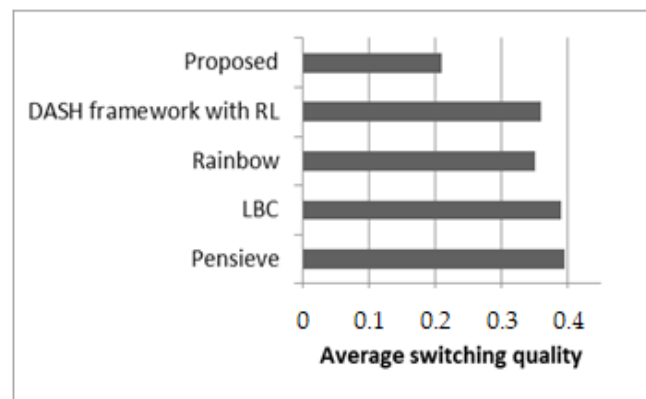
To summarize, the results unequivocally affirm that the proposed approach establishes stable and high-quality DASH streaming session. Its noteworthy performance improvements, collaborative agent interaction, and superior QoE metrics collectively emphasize its effectiveness. These findings represent a meaningful contribution to advancing DASH streaming research. Additionally, the more the agent streams videos in real-time with fluctuating bandwidth, the more it updates its neural network model and consequently improves the quality of experience in very highly dynamic environments.



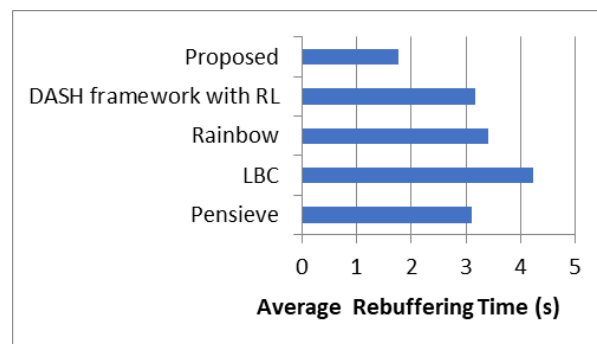
(a)



(b)



(c)



(d)

Figure 11. (a) Comparison of our approach vs. existing schemes; (b) average normalized QoE; (c) average switching quality; (d) average rebuffering time (s).

5. Conclusions

This study aimed to enhance the stable quality of video streaming within the DASH system by developing a novel deep-reinforcement-learning-based approach. The key contributions and findings of this research are as follows: First, we established a classification of qualities (bitrates) into three distinct classes: Q_{high} , Q_{medium} , and Q_{poor} . Additionally, we introduced a distance factor, Δ_{fact} , to quantify the difference between two qualities and defined a reward function that encapsulates the key aspects of quality change, rebuffering events, and segment quality. Second, we formulated the problem of DASH video streaming as a Markov Decision Process (MDP) learning model. In this model, an agent represents client-side adaptation, making bitrate selection decisions based on the current bandwidth measurements, Bw_t , buffer states, $buff_state_t$, and the quality, q_{t-1} , of the previous segment,

s_{i-1} . Our approach leverages an LSTM neural network for the agent and involves a two-phase training process: first, training the agent alone using the SJTU HDR video sequences dataset under various network conditions (utilizing the Norway Network Dataset); second, training the agent in collaboration with a secondary agent. The secondary agent intervenes only when alerted by the main agent's need for correction. Simulation results demonstrated that our approach outperforms state-of-the-art methods. Notably, it achieved over 98% in average segment quality and QoE value, successfully reducing rebuffering time to less than 1.8 s and limiting switching quality to below 0.21. These outcomes affirm the effectiveness of our approach in delivering stable and high-quality DASH streaming sessions. In future work, we intend to extend our approach to support live video streaming scenarios, further advancing the capabilities and performance of our proposed method. This research represents a significant step toward enhancing the quality of video streaming experiences and contributes to the ongoing evolution of DASH streaming technologies.

Author Contributions: Conceptualization, N.S.; Methodology, N.S.; Software, N.S.; Validation, N.S.; Formal analysis, N.S.; Investigation, N.S.; Resources, N.S.; Writing—original draft, N.S.; Visualization, M.B.; Supervision, M.B. and Y.D.; Project administration, M.B. and Y.D. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by The General Directorate of Scientific Research and Technological Development, Algeria (DGRSDT).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

Symbol	Meaning
V	Video set
Q	Set of different bitrates or qualities of a given video
S	Represent the user satisfaction
R	Reward function
R _t	Reward value obtained after a decision
R _b	Rebuffering
q	The quality of a segment
Q _{High}	High qualities set: Set of high qualities
Q _{Medium}	Medium qualities set: Set of medium qualities
Q _{Poor}	Poor qualities set: Set of Poor qualities
B _w	Bandwidth set
q _t , S _i	Quality t of segment i
$\Delta \text{fact}_{q_t}^{q_{t+1}}$	The factor distance between q _t and q _{t+1}
S _d	Download time of segment
S _p	Playback time of segment
Avg _q	Average quality of total video segments
N	Total number of segments in a given video
QoE _{max}	The estimated QoE during a streaming session
QoE	Represents the normalized QoE _{max}
Buff _{state_t}	Buffer state at time t
π	Policy
μ	Penalty of rebuffering
λ	Penalty of quality change

References

1. CISCO. Cisco Annual Internet Report (2018–2023). White Paper. 2020. Available online: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf> (accessed on 15 January 2022).
2. ITU-T SG12; Definition of Quality of Experience (QoE); COM12–LS 62–E, TD 109rev2 (PLEN/12). ITU: Geneva, Switzerland, 2007.
3. Slaney, M. Precision-Recall is Wrong for Multimedia. *IEEE MultiMedia* **2011**, *18*, 4–7. [\[CrossRef\]](#)
4. Petrangeli, S.; Hooft, J.V.D.; Wauters, T.; Turck, F.D. Quality of experience-centric management of adaptive video streaming services: Status and challenges. *ACM Trans. Multimed. Comput. Commun. Appl.* **2018**, *14*, 1–29. [\[CrossRef\]](#)
5. Akay, M.F.; Zayid, E.I.M.; Aktürk, E.; George, J.D. Artificial neural network-based model for predicting VO2max from a submaximal exercise test. *Expert Syst. Appl.* **2011**, *38*, 2007–2010. [\[CrossRef\]](#)
6. Zhang, Y. Applications of Artificial Neural Networks (ANNs) in Several Different Materials Research Fields. Ph.D. Thesis, Queen Mary University of London, London, UK, 2010.
7. Kerdvibulvech, C.; Saito, H. Vision-based detection of guitar players' fingertips without markers. In Proceedings of the Computer Graphics, Imaging and Visualisation, Bangkok, Thailand, 14–17 August 2007; pp. 419–428. [\[CrossRef\]](#)
8. Jung, E.; Kim, J.; Kim, M.; Jung, D.H.; Rhee, H.; Shin, J.-M.; Choi, K.; Kang, S.-K.; Kim, M.-K.; Yun, C.-H.; et al. Artificial neural network models for prediction of intestinal permeability of oligopeptides. *BMC Bioinform.* **2007**, *8*, 245. [\[CrossRef\]](#)
9. Liu, C.-C.; Lin, C.-C.; Li, K.-C.; Chen, W.-S.E.; Chen, J.-C.; Yang, M.-T.; Yang, P.-C.; Chang, P.-C.; Chen, J.J. Genome-wide identification of specific oligonucleotides using artificial neural network and computational genomic analysis. *BMC Bioinform.* **2007**, *8*, 164. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Huang, T.Y.; Johari, R.; McKeown, N.; Trunnell, M.; Watson, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In Proceedings of the 2014 ACM Conference on SIGCOMM, Chicago, IL, USA, 17–22 August 2014; pp. 187–198. [\[CrossRef\]](#)
11. Spiteri, K.; Urgaonkar, R.; Sitaraman, R.K. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Trans. Netw.* **2020**, *28*, 1698–1711. [\[CrossRef\]](#)
12. De Cicco, L.; Caldaro, V.; Palmisano, V.; Mascolo, S. ELASTIC: A client-side controller for dynamic adaptive streaming over HTTP (DASH). In Proceedings of the 2013 20th International Packet Video Workshop, San Jose, CA, USA, 12–13 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–8. [\[CrossRef\]](#)
13. Yin, X.; Jindal, A.; Sekar, V.; Sinopoli, B. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, London, UK, 17–21 August 2015; pp. 325–338. [\[CrossRef\]](#)
14. Beben, A.; Wiśniewski, P.; Batalla, J.M.; Krawiec, P. ABMA+ is a lightweight and efficient algorithm for HTTP adaptive streaming. In Proceedings of the 7th International Conference on Multimedia Systems, Klagenfurt, Austria, 10–13 May 2016; pp. 1–11.
15. Jiang, J.; Sekar, V.; Zhang, H. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive. In Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies, Paris, France, 10–13 December 2012; pp. 97–108.
16. Li, Z.; Zhu, X.; Gahm, J.; Pan, R.; Hu, H.; Begen, A.C.; Oran, D. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 719–733. [\[CrossRef\]](#)
17. Mao, H.; Netravali, R.; Alizadeh, M. Neural Adaptive Video Streaming with Pensieve. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM 17), Los Angeles, CA, USA, 21–25 August 2017; pp. 197–210. [\[CrossRef\]](#)
18. De Cicco, L.; Cilli, G.; Mascolo, S. Erudite: A deep neural network for optimal tuning of adaptive video streaming controllers. In Proceedings of the 10th ACM Multimedia Systems Conference, Amherst, MA, USA, 18–21 June 2019; pp. 13–24.
19. Kheibari, B.; Sayit, M. Quality estimation for DASH clients by using Deep Recurrent Neural Networks. In Proceedings of the 2020 16th International Conference on Network and Service Management (CNSM), Virtual, 2–6 November 2020; pp. 1–8.
20. Du, L.; Zhuo, L.; Li, J.; Zhang, J.; Li, X.; Zhang, H. Video Quality of Experience Metric for Dynamic Adaptive Streaming Services Using DASH Standard and Deep Spatial-Temporal Representation of Video. *Appl. Sci.* **2020**, *10*, 1793. [\[CrossRef\]](#)
21. Mao, H.; Chen, S.; Dimmery, D.; Singh, S.; Blaisdell, D.; Tian, Y.; Alizadeh, M.; Bakshy, E. Real-World Video Adaptation with Reinforcement Learning. *arXiv* **2020**, arXiv:2008.12858.
22. Fu, J.; Chen, X.; Zhang, Z.; Wu, S.; Chen, Z. 360SRL: A sequential reinforcement learning approach for ABR tile-based 360 video streaming. In Proceedings of the 2019 IEEE International Conference on Multimedia and Expo (ICME), Shanghai, China, 8–12 July 2019; pp. 290–295. [\[CrossRef\]](#)
23. Lekharu, A.; Moulii, K.Y.; Sur, A.; Sarkar, A. Deep learning-based prediction model for adaptive video streaming. In Proceedings of the 2020 International Conference on COMMunication Systems & Networks (COMSNETS), Bengaluru, India, 7–11 January 2020; pp. 152–159.
24. Liu, L.; Hu, H.; Luo, Y.; Wen, Y. When Wireless Video Streaming Meets AI: A Deep Learning Approach. *IEEE Wirel. Commun.* **2019**, *27*, 127–133. [\[CrossRef\]](#)
25. Liu, D.; Zhao, J.; Yang, C.; Hanzo, L. Accelerating Deep Reinforcement Learning With the Aid of Partial Model: Energy-Efficient Predictive Video Streaming. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 3734–3748. [\[CrossRef\]](#)
26. Gadaleta, M.; Chiariotti, F.; Rossi, M.; Zanella, A. D-DASH: A Deep Q-Learning Framework for DASH Video Streaming. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 703–718. [\[CrossRef\]](#)

27. Huang, T.; Zhang, R.X.; Zhou, C.; Sun, L. QARC: Video quality aware rate control for real-time video streaming based on deep reinforcement learning. In Proceedings of the 26th ACM International Conference on Multimedia, Seoul, Republic of Korea, 22–26 October 2018; pp. 1208–1216.
28. Tian, Z.; Zhao, L.; Nie, L.; Chen, P.; Chen, S. Deeplive: QoE optimization for live video streaming through deep reinforcement learning. In Proceedings of the 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 4–6 December 2019; pp. 827–831.
29. Xiao, G.; Wu, M.; Shi, Q.; Zhou, Z.; Chen, X. DeepVR: Deep Reinforcement Learning for Predictive Panoramic Video Streaming. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 1167–1177. [\[CrossRef\]](#)
30. Lu, L.; Xiao, J.; Ni, W.; Du, H.; Zhang, D. Deep-Reinforcement-Learning-based User-Preference-Aware Rate Adaptation for Video Streaming. In Proceedings of the 2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Belfast, UK, 14–17 June 2022; pp. 416–424. [\[CrossRef\]](#)
31. Houidi, O.; Zeghlache, D.; Perrier, V.; Quang, P.T.A.; Huin, N.; Leguay, J.; Medagliani, P. Constrained Deep Reinforcement Learning for Smart Load Balancing. In Proceedings of the 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Virtual, 8–11 January 2022; pp. 207–215. [\[CrossRef\]](#)
32. Ozcelik, I.M.; Ersoy, C. ALVS: Adaptive Live Video Streaming using deep reinforcement learning. *J. Netw. Comput. Appl.* **2022**, *205*, 103451. [\[CrossRef\]](#)
33. Turkkan, B.O.; Dai, T.; Raman, A.; Kosar, T.; Chen, C.; Bulut, M.F.; Zola, J.; Sow, D. GreenABR: Energy-aware adaptive bitrate streaming with deep reinforcement learning. In Proceedings of the 13th ACM Multimedia Systems Conference, Athlone, Ireland, 14–17 June 2022; pp. 150–163. [\[CrossRef\]](#)
34. Henrique, M.; Júnia, P.; Daniel, S.; Daniel, M.; Marcos, A.M.V. Improved Video Qoe in Wireless Networks Using Deep Reinforcement Learning. Available online: <https://ssrn.com/abstract=4356698> (accessed on 10 April 2023).
35. Hafez, N.A.; Hassan, M.S.; Landolsi, T. Reinforcement learning-based rate adaptation in dynamic video streaming. *Telecommun. Syst.* **2023**, *83*, 395–407. [\[CrossRef\]](#)
36. Naresh, M.; Saxena, P.; Gupta, M. PPO-ABR: Proximal Policy Optimization based Deep Reinforcement Learning for Adaptive BitRate streaming. *arXiv* **2023**, arXiv:2305.08114.
37. Big Buck Bunny Movie. Available online: <http://www.bigbuckbunny.org> (accessed on 2 May 2021).
38. Kang, J.; Chung, K. HTTP Adaptive Streaming Framework with Online Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 7423. [\[CrossRef\]](#)
39. Ismail, A.A.; Wood, T.; Bravo, H.C. *Improving Long-Horizon Forecasts with Expectation-Biased LSTM Networks*; Cornell University Library: Ithaca, NY, USA, 2018.
40. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [\[CrossRef\]](#)
41. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Adv. Neural Inf. Process. Syst.* **1992**, *12*, 2–5.
42. Keras Documentation. Available online: <https://keras.io> (accessed on 26 December 2018).
43. SJTU HDR Video Sequences. Available online: <https://medialab.sjtu.edu.cn/files/SJTU%20HDR%20Video%20Sequences/> (accessed on 3 July 2023).
44. Riiser, H.; Vigmostad, P.; Griwodz, C.; Halvorsen, P. Commute path bandwidth traces from 3G networks: Analysis and applications. In Proceedings of the 4th ACM Multimedia Systems Conference, Oslo, Norway, 28 February–1 March 2013; pp. 114–118.
45. Recommendation ITU-T P.800.1. *Series P: Terminals and Subjective and Objective Assessment Methods: Methods for Objective and Subjective Assessment of Speech and Video Quality, Mean Opinion Score (MOS) Terminology*; ITU: Geneva, Switzerland, 2016.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.