Article

Not peer-reviewed version

Deep Reinforcement Learning-Based Approach for Video Streaming DASH

Naima Souane *, Malika Bourenane *, Yassine DOUGA

Posted Date: 21 August 2023

doi: 10.20944/preprints202308.1429.v1

Keywords: DASH; video streaming; Wireless networks; QoE; Deep learning; Reinforcement Learning algorithms; Deep reinforcement learning; Bandwidth estimation

Article

Deep Reinforcement Learning-Based Approach for Video Streaming DASH

Naima Souane [1,*], Malika Bourenane [1,*] and Yassine Douga [2]

[1] LRIIR laboratory, Department of Computer Science, UNIVERSITE ORAN1, ORAN, ALGERIA.
[2] Department of Computer Science, UNIVERSITY OF BLIDA 1, BLIDA, ALGERIA.
* 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 video transmission across networks. Traditional adaptive bitrate (ABR) algorithms adjust the quality of video segments based on network conditions and buffer occupancy. However, these algorithms rely on fixed rules within a complex environment, making it challenging to achieve optimal decisions considering the overall context. In this paper, we propose a novel Deep Reinforcement Learning-based approach for streaming DASH, focusing on maintaining consistent perceived video quality throughout the streaming session to enhance user experience. Our approach optimizes the Quality of Experience (QoE) by dynamically controlling the quality distance factor between consecutive video segments. We evaluate this approach through a simulation model that encompasses diverse wireless network environments and various video sequences. Additionally, we compare our proposed approach 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; reinforcement learning algorithms; deep reinforcement learning; bandwidth estimation

1. Introduction

Maximizing the Quality of Experience (QoE) for users of video streaming services over wireless networks has become a prominent challenge for researchers and video providers. With multimedia content, particularly videos, occupying a substantial portion of Internet traffic [1], addressing this challenge is of utmost importance.

The Dynamic Adaptive Streaming over HTTP (DASH) standard plays a vital role in selecting the appropriate quality of video segments based on network conditions. In order to enhance user QoE, service providers and academic researchers have put forth several approaches aimed at achieving specific objectives, such as minimizing initial delays, preventing interruptions, and ensuring a consistently high quality throughout streaming sessions.

Within the DASH server, videos are divided into fixed-duration chunks or segments, typically ranging from 2 to 10 seconds. 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... [2].

Furthermore, during a streaming session, the segments can be classified into four states: played, in playing, in buffer, and to be downloaded. The buffer has the potential to 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. 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 Quality of Experience (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.

In this paper, we propose a novel ABR algorithm based on the Deep Reinforcement Learning (DRL) method 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. The primary objective of our proposal is to provide a highly satisfactory 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, focusing 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 Markov Decision Process, enabling the determination of optimal solutions through reinforcement learning.
- Analysis and implementation: We classify the available qualities (bitrates) into three classes: Qhigh, Qmedium, and Qpoor. 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 the video sequence on different devices under various network conditions. We monitored the performance of the experiments throughout the streaming session, specifically observing the perceptible quality switch. Based on the observed quality switches, we classified the bitrates into three distinct classes.
- 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.

The remaining sections of this study are structured as follows: Section 2 presents an overview of related work, examining previous research in the field. Section 3 provides a detailed description of our proposed solution, outlining the methodology and approach employed in developing the novel ABR algorithm based on Deep Reinforcement Learning. In Section 4, we present the experimental results obtained from evaluating our system, discussing the performance and outcomes of our approach. Finally, in Section 5, we conclude the paper by summarizing the key findings, highlighting the contributions of our work, and discussing potential avenues for future research in this area.

2. Related Work

In recent years, significant progress has been made in the field of adaptive video streaming, leading to the proposal of various approaches. These approaches can be categorized into four main categories based on the adaptation technology utilized:

- Traditional ABR-based approaches: This category encompasses approaches that rely on bandwidth measurement, buffer occupancy, or a combination of both to make streaming decisions. These approaches typically employ fixed rules for adaptation during the streaming process. While they have been widely used and implemented, their effectiveness can be limited in highly dynamic and diverse network conditions.
- Deep learning-based approaches: Deep learning techniques, specifically neural network models, are employed in this category. By training these models on extensive datasets, they can learn complex patterns and make informed decisions for adaptation. Deep learning-based approaches have demonstrated improved performance in adapting to diverse network conditions and user preferences. However, they often require large amounts of training data and computational resources for effective model training.
- Reinforcement learning-based approaches: In this category, adaptation decisions are made by an agent in an interactive environment through trial and error, guided by rewards. Reinforcement learning enables the agent to learn and optimize its decisions based on the received rewards. These approaches have the advantage of adaptability and the ability to handle dynamic and uncertain network conditions. However, training reinforcement learning models can be time-consuming, and the performance heavily relies on the reward design and exploration strategy.

- Deep reinforcement learning-based approaches: This category combines the power of deep neural networks with reinforcement learning techniques. Deep reinforcement learning approaches use deep neural networks to approximate various components of the reinforcement learning process, enabling them to handle complex streaming environments and make effective adaptation decisions. By leveraging the representation learning capabilities of deep neural networks, these approaches have shown promising results in achieving high-quality video streaming experiences.

2.1. Traditional ABR-based approaches

Many ABR algorithms [2] have been proposed for DASH, including BBA [3], BOLA [4], ELASTIC [5], MPC [6], ABMA+ [7], Festive [8], PANDA [9], and Pensieve [10]. These ABR algorithms are implemented on the client-side of DASH and aim to stream the video at the highest possible bitrates while minimizing rebuffing. However, these ABR algorithms rely on fixed rules for bitrate selection and streaming orchestration in highly dynamic environments. In such conditions, accurately characterizing and predicting bandwidth and making decisions based on the overall environment becomes challenging.

2.2. Deep learning-based approaches

Luca et al. [11] introduced ERUDITE, a deep neural network approach for 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 the observed bandwidth traces and the resulting Quality of Experience (QoE). The experimental results demonstrated that ERUDITE achieves near-optimal performance. In [12], the authors proposed a deep learning approach to predict the qualities of video segments that will be cached, aiming to enhance the performance of DASH in an SDN domain. They utilized a deep recurrent neural network (LSTM) with three hidden layers to forecast future segments. Additionally, they leveraged SDN technology in their approach. The simulation results showed that their solution effectively reduces under runs on the client side. In [13], the authors presented a video QoE metric for DASH utilizing a combination of a 3D CNN and LSTM to extract deep spatial-temporal features representing 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 utilized 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.

Overall, these studies demonstrate the application of deep learning techniques, including deep neural networks and recurrent neural networks, in optimizing various aspects of adaptive video streaming for improved performance and enhanced user experience.

2.3. Reinforcement Learning-based approaches

Hongzi et al. [14] developed an Adaptive Bitrate Reinforcement Learning (ABRL) system for Facebook's web-based video platform. The ABRL system generates RL-based ABR policies and deploys them in the production environment. They created a simulator to train the ABR agent using RL techniques, and the translated ABRL policies are then deployed to the user front end. Jun et al. [15] proposed an improved ABR approach called 360SRL (Sequential Reinforcement Learning) to reduce the decision space and enhance the Quality of Experience (QoE). The 360SRL method learns the ABR policy by optimizing the QoE value or reward. They defined the reward function as a weighted sum of video quality, rebuffing, and spatial and temporal video quality variance. In their training process, they utilized the Deep Q-Network (DQN) method.

These studies highlight the application of reinforcement learning techniques, such as RL and DQN, in designing and improving ABR systems for web-based video platforms. The focus is on optimizing ABR policies to enhance the user's QoE by considering factors such as video quality, rebuffing, and temporal/spatial video quality variance.

2.4. Deep Reinforcement Learning-based approaches

In recent years, several studies have been conducted to improve the Quality of Experience (QoE) in video streaming through the application of Deep Reinforcement Learning (DRL) techniques. These works have explored various neural network architectures and reinforcement learning algorithms to optimize adaptive video streaming and enhance user satisfaction. This section presents a comprehensive overview of the relevant literature in this field.

Anirban et al. [16] proposed the LASH model, a deep neural network-based approach that combines LSTM, CNN, and reinforcement learning for adaptive video streaming. Their objective was to maximize QoE by considering perceived video quality, buffering events, and the smoothness of the video. However, the specific neural network architectures and reinforcement learning algorithms used were not explicitly mentioned.

Lu et al. [17] developed an end-to-end unified Deep Neural Network (DNN) architecture to predict the Quality of Service (QoS) and QoE values in wireless video streaming sessions. Their framework employed two hidden layers and utilized the Deep Q-Network (DQN) algorithm for bit rate adjustment. The primary focus was on deriving an optimal rate adaptation policy to maximize rewards during video streaming.

Dong et al. [18] proposed an online and end-to-end DRL-based policy for power allocation in video streaming. They employed the Deep Deterministic Policy Gradient (DDPG) algorithm and integrated safety layer, post-decision state, and virtual experiences to ensure quality of service and enhance convergence speed.

Matteo et al. [19] designed 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 the Dynamic Adaptive Streaming over HTTP (DASH) QoE. Their reward function incorporated the Structural Similarity Index (SSIM) and considered video quality switches and rebuffering events.

Tianchi et al. [20] proposed a video quality-aware rate control approach aiming to achieve high perceptual quality with low latency. Their method employed deep reinforcement learning to train a neural network for selecting future bitrates. The VQPN model was used for predicting the next video quality, and VQRL was utilized to train the neural network.

Zhao et al. [21] introduced the Deeplive model, which utilized two neural networks to make decisions on bit rate, target buffer, and latency limit 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 bit rate switches.

Gongwei et al. [22] presented the DeepVR approach for predictive panoramic video streaming using DRL techniques. Their method employed LSTM to predict the users' Field of View (FoV) in future seconds, and the DRL agent selected the quality level based on a reward function incorporating FoV quality, rebuffering time, video quality change, and buffer. The Rainbow training method was utilized to train the DRL agent.

Furthermore, other studies have focused on optimizing specific aspects of video streaming. For instance, an HTTP adaptive streaming framework with online reinforcement learning was proposed in [23], where the neural network was trained with the Pensieve reinforcement learning algorithm. The main objective was to upgrade the Adaptive Bitrate (ABR) model when QoE degradation occurs.

Ling yun Lu et al. employed the Actor-Critic (A3C) algorithm for adaptive streaming based on DRL [24]. Their approach selected the optimal bitrate based on user preferences, network throughput, and buffer occupancy. Similarly, Omar Houidi et al. [25] adjusted the actor-critic architecture for DRL to consider Quality of Service (QoS) and balance network traffic, thereby maximizing QoE.

ALVS [26] proposed a DRL framework for video live streaming, where adaptive playback speed and video quality level were decided for the next segment to improve QoE. Additionally, GreenABR [27] introduced a DRL-based ABR scheme that aimed to optimize energy consumption during a streaming session without degrading QoE.

Several DRL-based approaches and frameworks are proposed in the literature, and certain of them use complicated neural network architectures (combined two or more neural networks). In contrast of cited work, we proposed a new DRL-based approach with LSTM neural network that considers the overall environment for deciding the next bitrate. The DRL used deep neural networks to approximate the policy. To improve the QoE, we introduced the factor distance to decide the next bitrate. The main goal is to provide an optimal dynamic adaptation in very complex environments.

3. Materials and methods

3.1. Problem formulation

The deep reinforcement learning algorithm focuses on how agents take actions to achieve a goal or maximize a specific dimension over multiple steps in a complex environment and gain maximum cumulative rewards. The DRL algorithm makes a sequential decision in a time interval that plays a crucial role in problem-solving. There are two essential learning models in reinforcement learning: the Markov decision process and Q-learning. In this paper, we formulate the streaming video DASH in the learning model: Markov Decision Process (MDP) for solution determination in reinforcement learning. RL has five components: the agent, the environment, the states, the actions, and the rewards. The agent takes sequential actions based on the current state of the environment. After every action, the environment moves to another state, and the agent receives a reward. The agent aims to maximize the total of received rewards in specific steps.

The deep reinforcement learning algorithm focuses on how agents take actions to achieve a goal or maximize a specific dimension over multiple steps in a complex environment and gain maximum cumulative rewards. The DRL algorithm makes sequential decisions in time intervals that play a crucial role in problem-solving. There are two essential learning models in reinforcement learning: the Markov decision process and Q-learning. In this paper, we formulate the streaming video DASH learning model as a Markov Decision Process (MDP) for solution determination in reinforcement learning. RL has five components: the agent, the environment, the states, the actions, and the rewards. The agent takes sequential actions based on the current state of the environment. After every action, the environment transitions to another state, and the agent receives a reward. The agent aims to maximize the total of received rewards in specific steps.

3.2. System model

This section details the system model of the proposed approach. The Deep Reinforcement Learning environment refers to the DASH system. The DASH server provides a set of videos V {v_1, v_2 ...v_N}; each video is encoded in segments of different bitrates or quality Q {q_1, q_2, q_3 ...q_M} of fixed duration. Additionally, each video is associated with an MPD description file containing this information. At the start of a video streaming session, The DASH client requests the MPD file, analyzes the information within it, and then initiates the streaming based on the current network and device conditions. We consider the client-side adaptation as an agent that determines the appropriate bitrate for segments using an optimal policy. After downloading the first segment, the agent observes the environment, including network parameters (bandwidth measurement), buffer state, and the quality (bitrate) of the previous segment, to decide the optimal bitrate for the next segment. The agent continues to take actions and receive rewards until all segments have been downloaded.

We adopt the learning model: MDP to determine the optimal bitrate adaptation strategy based on distance factor between two consecutive segments. Our goal is to achieve high user satisfaction S = {s_1, s_2, s_3...} in terms of avoiding playback interruptions and guaranteeing a stable high-quality video streaming. Here, S represents the function of playback interruption and stable high quality, denoted as.

$$S = f(\text{playback}_{\text{interruption}}, \text{stable\_high\_quality}), \tag{1}$$

3.3. Reward function

This work aims to provide highly stable perceptual quality and achieve high user satisfaction without rebuffering. Therefore, we define a reward function $R = f(r, rb, q)$ that considers the rate of perceived quality change (r), rebuffering (rb), and the quality of each segment (q).

In this subsection, we present the experimental study we have performed to classify the bitrates (qualities). For the experiment, we used the BigBuckBunny video sequence from the DASH dataset [28]. All users participated in the tests starting to play their video sequences on different devices under various internet conditions. The experiment was conducted during the streaming session. Furthermore, we programmed bitrate switches as follows: {(144p, 240p), (144p, 360p), (144p, 720p)…} and observed the perceived quality change.

According to the experimental results, we classified the qualities into three classes: Q_{high}, Q_{medium} and Q_{poor}. As shown in Figure 1. To define these classes, we establish the following quality sets: Q_{high} = {1080p... 2160p} for high-quality, Q_{medium}= {360p... 720p} for medium quality, and Q_{poor}= {144p... 240p} for poor quality. The quality *qt* of a segment s_i can be categorized into the quality set Q = {Q_{high}, Q_{medium}, Q_{poor}}, while the bandwidth can be classified into the set Bw = {Bw_{low}, Bw_{medium}, Bw_{high}}.

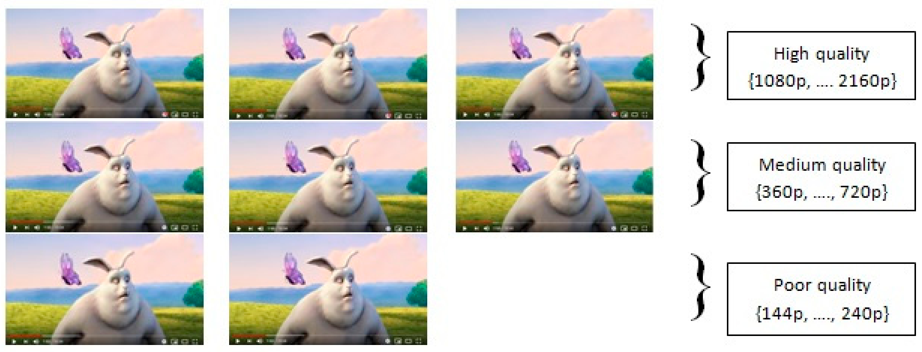


Figure 1. Classification of qualities.

3.3.1. Perceived quality change

In this proposal, we redefine the rate of quality change. Therefore, we consider a perceived quality change when the user notices a switch, meaning that a change of quality within the same set is not considered a change. However, we do consider a change when the quality q_t is in the poor-quality set, and q_{t+1} is in the high-quality set. Thus, we introduce a distance factor $\Delta fact$ that measures the difference between two qualities, as illustrated in Table 1.

Table 1. The distance factor values between qualities.

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 $\Delta fact_{qt}^{qt+1}$ =|1| value is |1|, given that q_t corresponds to Medium quality and q_{t+1} corresponds to high quality.

3.3.2. Rebuffering

The rebuffering *Rb* occurring after each downloaded segment can be measured by comparing the playback time of a downloaded segment *sp* and the time of the end of download *sd*. A rebuffering event occurs when *sd* is greater than *sp*, indicating a pause or interruption in video playback. Therefore, the rebuffering *Rb* can be defined as the total number of video interruptions.

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

3.3.3. Segment quality

The segment quality represents the quality of each downloaded segment, denoted as q_i. Therefore, the average quality (Avg_q) is defined as the sum of all segment qualities (q_k) divided by the total number of segments (N), as shown below:

$$Avg\_q = \frac{\sum_{k=1}^{N} q_k}{N}, \tag{3}$$

Here, N represents the total number of segments.

3.3.4. QoE function

In this section, we highlight the QoE function. As described above, the QoE is based on three parameters: the quality of each segment, the perceptual change of quality and the rebuffering or playback. As we know, the agent receives a reward for every decision of quality; in this way, we obtain the following function:

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

Thus, the QoE can be expressed as:

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

Where μ and λ represent the penalties for rebuffering and the change in quality, respectively.

We normalize the QoE_{value} to match it with the state-of-the-art. Thus, we obtain the following expression:

$$QoE = \frac{QoE_{max}}{N} * 100, \tag{6}$$

Where N is the total number of video segments.

3.4. Markov Decision Process (MDP)

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

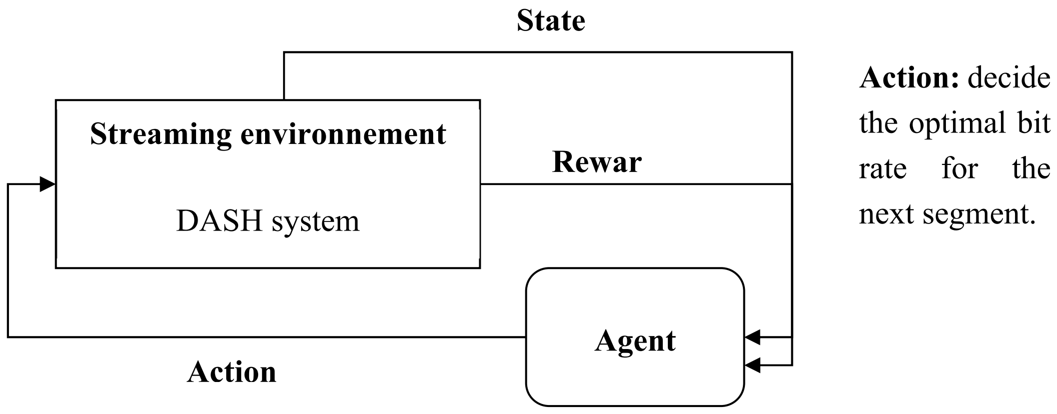


Figure 2. Reinforcement learning architecture.

The agent represents the client-side adaptation and is responsible for selecting an action at. This action involves choosing an appropriate bitrate or quality q_t for the next segment S_i to be downloaded. The agent's decision is based on current bandwidth measurements Bw_t, the current buffer state *buff_state*$_t$ and the quality q_{t-1} of the previous segment s_{i-1}. The agent receives rewards or penalties based on its bitrate selection (adaptation).

In adaptive video streaming, we define the segments as a finite set of states, S = {s_0,s_1,… s_N}. It should be noted that a video is divided into N segments. We define a set of actions as {$\mathbf{a_t}$}, where t = {0, 1, 2 …N}, indicating the decision made at step **t.** The duration of each step, **t,** depends on the download time of a segment. The action set for a given state is denoted as A(s) = {A_{s1}, A_{s2}, A_{s3}}. Here, A_{s1} represents the case when q_t and q_{t-1} are in the same quality set defined previously. A_s the two indicate, q_t and q_{t-1} are not in the same quality set but are the closest to each other. A_{s3} signifies that q_t is in a lower quality set. The state transition at step is defined as s_t = {q_{t-1}, q_t, buff_state$_t$, Bw_t}, where q_{t-1} represents the quality of the previous segment, q_t represents the quality of the downloaded segment, Bw_t represents the current bandwidth, and buff_state$_t$ represents the current buffer state. To determine the quality of the next segment, the agent utilizes a policy π as an approximation strategy. In this work, we integrate a deep neural network (DNN) to approximate the policy $\pi(a|s;\theta)$, which maps from the current state S to the next state S′. The agent is represented by a deep neural network that generates the policy for making decisions, and θ represents the weights in the deep neural network.

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

$$S' = (Bw', \text{buff\_state}_t', q_t'), \tag{7}$$

Where:

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

$Bw_t' = \Delta Bw_t$

buff_state$_t'$ = Δ buff_state$_t$

A reward function $R_i(S_t)$ is received after downloading each segment S_t:

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

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

3.4. Deep neural network architecture

A typical deep neural network (DNN) consists of an input layer, multiple hidden layers, and an output layer. Each layer is composed of neurons that are activated based on the input from the previous layer using an activation function. In this paper, we utilize the Long-Short Term Memory network (LSTM) [29].

LSTM is a powerful variant of recurrent neural networks (RNNs) widely used for processing and predicting sequence data, with a particular focus on time-series data. The LSTM unit comprises four essential components, as visually depicted in Figure 3. These components include a specialized cell designed to store and retain information over extended periods. Additionally, the cell incorporates three pivotal gates: the input gate, output gate, and 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 the quality information from previous segments, while disregarding historical details such as bandwidth and buffer state information.

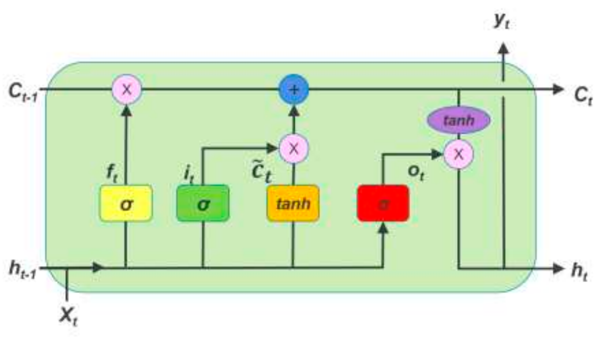


Figure 3. LSTM cell architecture [30].

3.4.1. Agent design

The agent is composed of an LSTM neural network that makes decisions based on the current bandwidth, buffer state, and the quality of the previously downloaded segment. The LSTM takes the state St = (qt, Bwt, Buft) as input and selects an action that corresponds to the next state St+1= (qt+1, Bwt, 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 slave agent, allowing for potential corrective measures. The slave agent then observes the environment to determine if a correction can be made without adversely impacting the Quality of Experience (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.

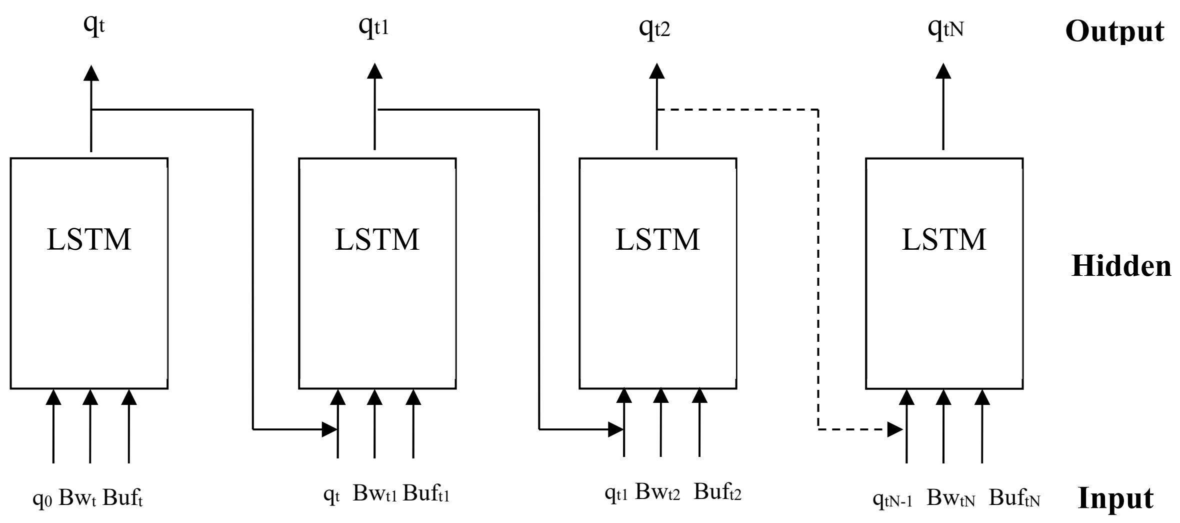

q_t
q_{t1}
q_{t2}
q_{tN}
Output
LSTM
LSTM
LSTM
LSTM
Hidden
q_0 Bw_t Buf_t
q_t Bw_{t1} Buf_{t1}
q_{t1} Bw_{t2} Buf_{t2}
q_{tN-1} Bw_{tN} Buf_{tN}
Input

Figure 4. Agent Architecture.

3.4.2. Agent Training

In this section, we provide a detailed explanation of the training process for our DRL approach. The agent is trained in various environments using bandwidth statistics and videos encoded at different quality levels, as well as different numbers of segments.

In the offline phase, we first start by the training of two twin agents alone: the main and the slave agent. We initiate the training by separately training two twin agents: the main agent and the slave agent. We utilize the rewards and penalties specified in Table 2 to train these agents. The objective is to minimize the (factor distance) difference between consecutive segments, prevent rebuffering events, and select high-quality segments for downloading.

Table 2. Rewards defined for each state.

St = S		R (St = S)
(qt, qt-1) ϵ Q	// qt and qt-1 are in the same quality set	1
(qt) ϵ Qhigh	// qt is in the high-quality set	0.75
(qt) ϵ Qmeduim	// qt is in the medium-quality set	0.50
(qt) ϵ Qpoor	// qt is in the poor-quality set	-1
Rbt= 0	// there is no rebuffering event	1
Rbt> 0	// there is a rebuffering event	-1
(qt, qt-1) ∉ Q	// qt and qt-1 are not in the same quality set	-1

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

In this paper, we adopts the policy gradient algorithm REINFORCE [31] to optimize the agent's policy (6) and update the policy parameter θ.

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

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:

$$R_t = \sum_t \gamma^t * r_t \tag{10}$$

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

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

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

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

4. Performance evaluation

In this section, we will discuss the implementation environment, provide details about the LSTM neural network, describe the experiment setup, and outline the datasets used for training and testing the model. Subsequently, we will present the results of our approach compared to the state of the art.

Implementation Details: The simulation is implemented in python using the Keras API [32]. We utilize the neural network agent architecture demonstrated in Figure 4, which consists of a single hidden layer. The activation function used is **TanH,** which maps the output values to the interval [-1, 1]. The number of iterations corresponds to the total number of video segments. During the initial phase, the model takes inputs such as the quality of the first download segment (determined according to 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.

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

Experiment setup: to evaluate the performance of our proposed approach, we conducted experiments using a dataset obtained from [33]. This Dataset consists of 15 ultra-high definition (UHD) video sequences with a resolution of 4K. Each video sequence is divided into segments, with each segment lasting 2 seconds. During playback, a buffer size of 30 seconds is maintained. The video sequences are encoded using the High-Efficiency Video Coding (HEVC) format and have a frame rate of 30 frames per second (fps).

For network conditions, we utilized the Norway dataset from [34], which contains network traces collected from Telenor's 3G/HSDPA mobile wireless network in Norway. These traces provide realistic network conditions for our experiments, capturing the variations and characteristics of real-world mobile networks.

In this experiment, the agent is trained in two steps. First, the agent is trained alone using Deep Reinforcement Learning (DRL) in offline mode. For this training phase, 70% of the video sequences from the dataset are used to train the agent, while the remaining 30% are held back for testing the model's performance. This initial training enables the agent to learn and make decisions based on the training data.

In the second step, the agent is trained with the assistance of a slave agent, which helps in providing high-quality over-streaming sessions. The slave agent works in collaboration with the primary agent and corrects any incorrect decisions made by the primary agent, leading to improved adaptation and decision-making processes.

By conducting experiments with this setup, we aim to evaluate the effectiveness of our proposed approach in handling real-world video streaming scenarios and optimizing the quality of the streamed content.

Results: Firstly, we present the results of training the LSTM neural network, showcasing the performance and learning achieved during the training process. We analyze the network's convergence, loss, and other relevant metrics to assess its effectiveness in optimizing the streaming quality.

Next, we compare the performance of our model with the state-of-the-art approaches. By conducting experiments in a simulation environment with various video sequences and different network traces, we evaluate the robustness and generalization capability of our proposed approach. We consider key performance indicators such as video quality, buffer occupancy, rebuffering events, and user experience to assess the effectiveness of our approach.

To ensure comprehensive evaluation, we trained the neural agent network using 70% of the video sequences from the dataset. Subsequently, the remaining 30% of the dataset videos were used to perform the testing phase. This division allows us to gauge the model's performance on unseen data and assess its ability to generalize beyond the training set.

The presented results are the average of all experiments conducted, ensuring a reliable and representative assessment of our approach's performance. By providing these results, we aim to demonstrate the effectiveness and potential advantages of our proposed approach in optimizing video streaming quality and enhancing user experience.

Figure 5 illustrates the degree of performance improvement achieved using 70% of the video sequences from the dataset. The graph showcases the progress of performance improvement as the training process advances, with the x-axis representing the percentage of data utilized for training.

Based on the graph, it can be observed that the performance improvement steadily increases as more training data is utilized. However, the rate of improvement starts to converge after approximately 60% of the data has been incorporated into the training process.

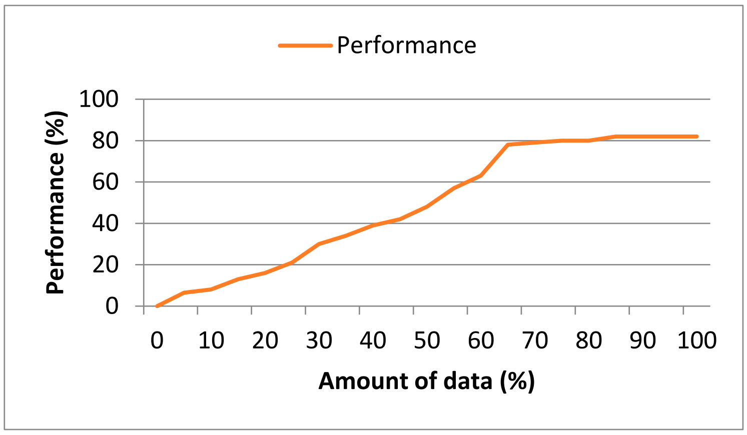


Figure 5. Degree of performance improvement with 70% of Dataset.

This graph provides insights into the effectiveness of our approach in optimizing performance and highlights the saturation point where further training data might not significantly contribute to additional improvements. It serves as an important reference for determining the optimal utilization of training data for achieving the desired performance levels.

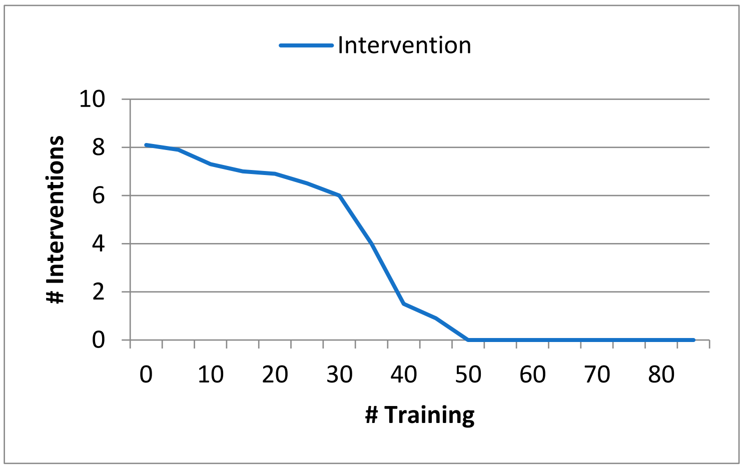


Figure 6. Degree of performance improvement with a Slave agent.

In this experiment, we trained the primary neural network agent in conjunction with a slave agent. The aim of this experiment was to evaluate the impact of the slave agent's intervention on performance improvement. Throughout the experiment, the overall system monitored the demand for intervention and sent alert messages to the slave agent in case of penalties or incorrect decisions.

Figure 7 showcases the results of the performance improvement achieved with the inclusion of the slave agent. The graph represents the progression of performance improvement over the training process, with the x-axis indicating the percentage of video sequences utilized for training.

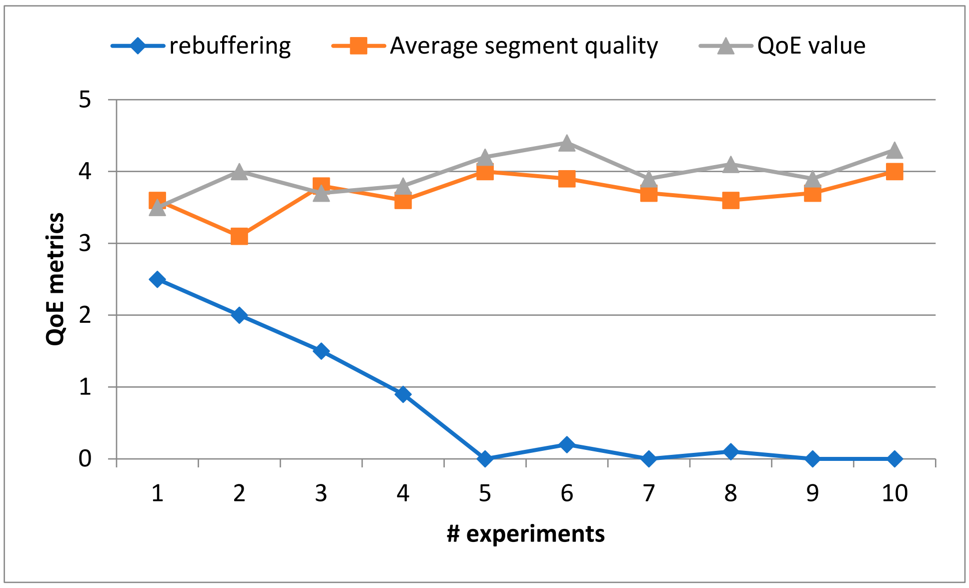


Figure 7. Results of QoE metrics.

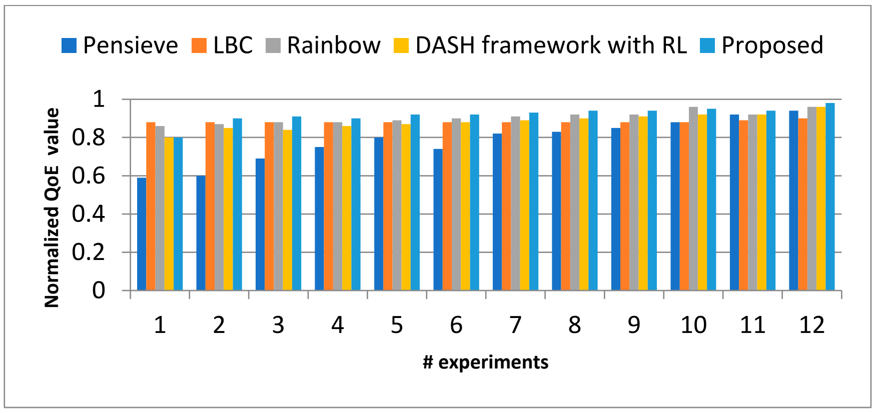
During the experiment utilizing 70% of the dataset, we observed a notable reduction in the number of alert messages sent by the overall system after approximately 55% of the video sequences were incorporated into the training process. This indicates that the collaboration between the primary agent and the slave agent positively influenced the decision-making process, resulting in improved performance and a decreased occurrence of penalties or incorrect decisions.

Figure 6. provides valuable insights into the effectiveness of utilizing a slave agent and highlights the impact of its intervention on performance improvement. It demonstrates the importance of collaborative approaches in enhancing the overall system's decision-making capabilities and ultimately optimizing performance.

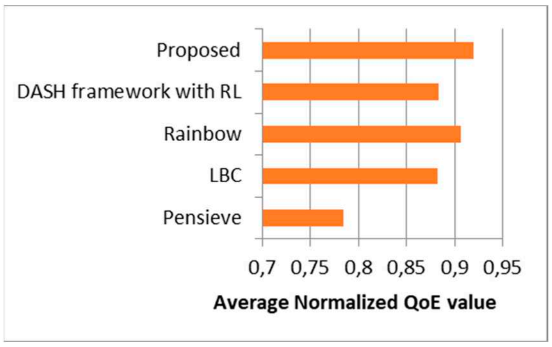
Figure 7 presents the results of various Quality of Experience (QoE) metrics, including rebuffering, average segment quality, and the overall QoE value. These metrics were utilized to evaluate the performance of our proposed approach.

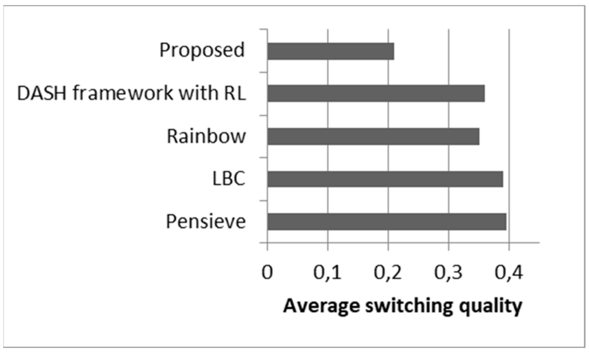
Throughout the experiment, we placed particular emphasis on these metrics as they directly reflect the user's streaming experience. By analyzing rebuffering events, which represent interruptions in the video playback, we can assess the system's ability to maintain a smooth streaming experience. The average segment quality metric allows us to gauge the overall video quality delivered to the user, while the QoE value provides an aggregated measure of the user's satisfaction with the streaming service. Figure 7 provides a visual representation of the performance of our approach based on these QoE metrics. The graph showcases the progression of these metrics over the course of the experiment, allowing us to analyze and compare the performance at different stages. By evaluating these QoE metrics, we can gain insights into the effectiveness of our approach in enhancing the user experience during video streaming. These metrics provide a comprehensive assessment of the quality and reliability of the streaming service, enabling us to make informed judgments about the performance and effectiveness of our proposed approach.

In this experiment, we compare the performance of our proposed approach with several existing schemes, namely Pensieve [10], DASH framework with online Reinforcement Learning [36], LBC [17], and Rainbow [22]. The results presented in Figure 8 represent the average outcomes obtained from conducting over ten experiments.

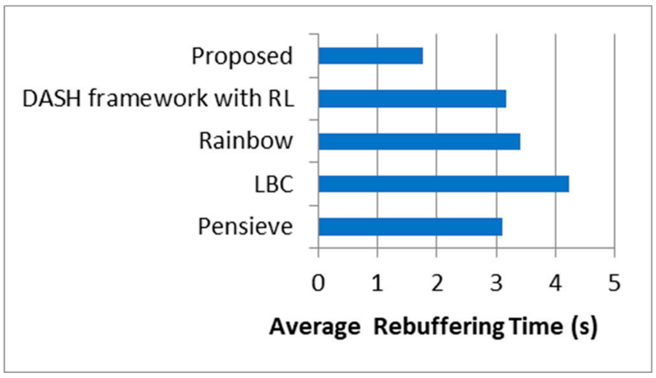


(a)





(b) (c)



(d)

Figure 8. (a) Comparison of Our Approach vs. Existing Schemes; (b) Average normalized QoE ; (c) Average switching quality; (d) Average rebuffering Time (s).

The results depicted in Figure 8 clearly illustrate that our proposed approach consistently outperforms the existing schemes in terms of the normalized QoE value. This signifies that our approach offers a superior solution for delivering stable and high-quality DASH streaming sessions.

Based on these comparative results, we can conclude that our proposed approach demonstrates significant performance improvements when compared to the existing schemes. The averaged results obtained from the experiments further validate the effectiveness of our approach and highlight its potential to enhance the streaming experience compared to state-of-the-art solutions.

5. Discussion

The discussion of the experimental results highlights several key points. Firstly, the model converged after approximately 60% of the dataset, indicating that the training process reached a stable state. Secondly, the inclusion of the slave agent in our approach led to increased performance improvement and reduced penalties. This collaborative interaction between agents proved effective.

Moreover, the evaluation of QoE metrics showed that our proposed approach achieved high results in terms of average segment quality and QoE value, while successfully reducing rebuffering. These outcomes indicate that our approach offers a stable and high-quality DASH streaming session.

Additionally, when compared to existing schemes, our proposed approach consistently outperformed them in terms of normalized QoE value. This reinforces the superiority of our approach in delivering an enhanced user experience.

Overall, the discussion confirms that our proposed approach provides a stable and high-quality DASH streaming session, and its performance improvements, collaborative agent interaction, and superior QoE metrics demonstrate its effectiveness. These findings contribute to the advancement of DASH streaming research.

6. Conclusions

The purpose of this study was to enhance the stable quality of video streaming in the DASH system. To achieve this, we introduced a novel approach based on Deep Reinforcement Learning. We formulated the DASH video streaming as a Markov Decision Process (MDP) learning model and incorporated a distance factor (DF) to capture the difference between segment qualities. The LSTM neural network was trained using ABR algorithms and a dataset of video sequences under various network environments.

Through simulations and comparisons with existing studies, we demonstrated that our proposed approach significantly improved the overall DASH streaming system and provided users with a highly stable quality during their streaming sessions.

In future work, we aim to extend our approach to support live video streaming, further advancing the capabilities and performance of our proposed method.

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

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

Appendix A

Table A1 summarizes the main symbols used throughout this paper.

Table A1. Symbols table.

Symbol	Meaning
V	Video set
Q	Set of different bitrates or qualities of a given video
S	Represent the user satisfaction
R	Reward function
Rt	Reward value obtained after a decision
Rb	Rebuffering
q	The quality of a segment
QHigh	High qualities set: Set of high qualities
QMedium	Medium qualities set: Set of medium qualities
QPoor	Poor qualities set: Set of Poor qualities
Bw	Bandwidth set
q_t, S_i	Quality t of segment i
$\Delta fact_{qt}^{qt+1}$	The factor distance between q_t and q_{t+1}
Sd	Download time of segment
Sp	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 rebuferring
λ	Penalty of quality change

References

1. CISCO. "Cisco Annual Internet Report (2018-2023)." White Paper. 2020. [Online] Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf
2. Sodagar, I. The MPEG-dash standard for multimedia streaming over the internet. IEEE Multimedia, 18(4), 62–67. https://doi.org/10.1109/MMUL.2011.50
3. 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, pp. 187–198. https://doi.org/10.1145/2619239.2626311
4. Spiteri, K.; Urgaonkar, R.; Sitaraman, R. K. BOLA: Near-optimal bitrate adaptation for online videos. IEEE/ACM Transactions on Networking, 28(4), 1698-1711.
5. De Cicco, L.; Caldaralo, V.; Palmisano, V.; Mascolo, S. ELASTIC: A client-side controller for dynamic adaptive streaming over HTTP (DASH). In 2013 20th International Packet Video Workshop, pp. 1-8. IEEE.
6. 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, pp. 325-338. https://doi.org/10.1145/2785956.2787485
7. 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, pp. 1-11.

8. 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, pp. 97-108.
9. 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 Journal on Selected Areas in Communications, 32(4), 719-733.
10. 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), Association for Computing Machinery, New York, NY, USA, pp. 197–210. https://doi.org/10.1145/3098822.3098843
11. 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, pp. 13-24.
12. Kheibari, B.; Sayıt, M. Quality estimation for DASH clients by using Deep Recurrent Neural Networks. In 2020 16th International Conference on Network and Service Management (CNSM), pp. 1-8. IEEE.
13. 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 video representation. Applied Sciences, 10(5), 1793.
14. Mao, H.; Chen, S.; Dimmery, D.; Singh, S.; Blaisdell, D.; Tian, Y.; Alizadeh, M.; Bakshy, E. Real-World Video Adaptation with Reinforcement Learning. Preprints 2020, 2020080584 (doi: 10.20944/preprints202008.0584.v1).
15. Fu, J.; Chen, X.; Zhang, Z.; Wu, S.; Chen, Z. 360SRL: A sequential reinforcement learning approach for ABR tile-based 360 video streaming. In 2019 IEEE International Conference on Multimedia and Expo (ICME), pp. 290–295. IEEE.
16. Lekharu, A.; Moulii, K. Y.; Sur, A.; Sarkar, A. Deep learning-based prediction model for adaptive video streaming. In 2020 International Conference on COMmunication Systems & Networks (COMSNETS), pp. 152-159. IEEE.
17. Liu, L.; Hu, H.; Luo, Y.; Wen, Y. When wireless video streaming meets AI: a deep learning approach. IEEE Wireless Communications, 27(2), 127-133.
18. Liu, D.; Zhao, J.; Yang, C.; Hanzo, L. Accelerating deep reinforcement learning with the partial model: Energy-efficient predictive video streaming. IEEE Transactions on Wireless Communications, 20(6), 3734–3748.
19. Gadaleta, M.; Chiariotti, F.; Rossi, M.; Zanella, A. D-DASH: A deep Q-learning framework for DASH video streaming. IEEE Transactions on Cognitive Communications and Networking, 3(4), 703-718.
20. 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, pp. 1208–1216.
21. Tian, Z.; Zhao, L.; Nie, L.; Chen, P.; Chen, S. Deeplive: QoE optimization for live video streaming through deep reinforcement learning. In 2019 IEEE 25th international conference on parallel and distributed systems (ICPADS), pp. 827–831. IEEE.
22. Xiao, G.; Wu, M.; Shi, Q.; Zhou, Z.; Chen, X. DeepVR: Deep reinforcement learning for predictive panoramic video streaming. IEEE Transactions on Cognitive Communications and Networking, 5(4), 1167–1177.
23. Mao, H.; Netravali, R.; Alizadeh, M. Neural adaptive video streaming with pensieve. In Proceedings of the ACM special interest group conference on data communication, pp. 197-210.
24. Lu, L.; Xiao, J.; Ni, W.; Du, H.; Zhang, D. Deep-Reinforcement-Learning-based User-Preference-Aware Rate Adaptation for Video Streaming. In 2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 416-424. IEEE.
25. O. Houidi et al. Constrained Deep Reinforcement Learning for Smart Load Balancing. 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, pp. 207-215. [Online] Available: https://doi.org/10.1109/CCNC49033.2022.9700657
26. Ozcelik, I. M.; Ersoy, C. ALVS: Adaptive Live Video Streaming using deep reinforcement learning. Journal of Network and Computer Applications, 205, 103451.
27. Turkkan, B. O.; Dai, T.; Raman, A.; Kosar, T.; Chen, C.; Bulut, M. F.; Sow, D. GreenABR: energy-aware adaptive bitrate streaming with deep reinforcement learning. In Proceedings of the 13th ACM Multimedia Systems Conference, pp. 150-163. https://doi.org/10.1145/3524273.3528188
28. Big Buck Bunny Movie. Available online: http://www.bigbuckbunny.org (accessed on 6 January 2023).

29. Improving Long-Horizon Forecasts with Expectation-Biased LSTM Networks. (2021). [Online] Available: https://doi.org/10.475/123_4
30. Hochreiter, S.; Schmidhuber, J. Long short-term memory. Neural Computation, 9(8), 1735-1780.
31. Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour AT&T Labs - Research, 180 Park Avenue, Florham Park, NJ 07932. Policy Gradient Methods for Reinforcement Learning with Function Approximation.
32. Keras: Deep Learning for humans. Available online: https://keras.io/ (accessed on April 2023).
33. SJTU HDR Video Sequences. Available online: https://medialab.sjtu.edu.cn/files/SJTU%20HDR%20Video%20Sequences/
34. Riiser, H.; Vigmostad, P.; Griwodz, C.; Halvorsen, P. Commute path bandwidth traces from 3G networks: analysis and applications. Proceedings of the 4th ACM Multimedia Systems Conference, pp. 114-118.
35. Pensieve: 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, pp. 197-210.
36. Kang, J.; Chung, K. HTTP Adaptive Streaming Framework with Online Reinforcement Learning. Appl. Sci., 2022, 12, 7423. [Online] Available: https://doi.org/10.3390/app12157423