TCLiVi: Transmission Control in Live Video Streaming Based on Deep Reinforcement Learning

Laizhong Cui, *Senior Member, IEEE*, Dongyuan Su, Shu Yang, Zhi Wang, and Zhong Ming

***Abstract*—Currently, video content accounts for the majority of network traffic. With increased live streaming, rigorous requirements have been introduced for better Quality of Experience (QoE). It is challenging to meet satisfactory QoE in live streaming, where the aim is to achieve a balance between 1) enhancing the video quality and stability and 2) reducing the rebuffering time and end-to-end delay, under different scenarios with various network conditions and user preferences, where the fluctuation in the network throughput degrades the QoE severely. In this paper, we propose an approach to improve the QoE for live video streaming based on Deep Reinforcement Learning (DRL). The new approach jointly adjusts the streaming parameters, including the video bitrate and target buffer size. With the basic DRL framework, TCLiVi can automatically generate the inference model based on the playback information, to achieve the joint optimization of the video quality, stability, rebuffering time and latency parameters. We evaluate our framework on real-world data in different live streaming broadcast scenarios, such as a talent show and a sports competition under different network conditions. We compare TCLiVi with other algorithms, such as the Double DQN, MPC and Buffer-based algorithms. The simulation results show that TCLiVi significantly improves the video quality and decreases the rebuffering time, consequently increasing the QoE score by 40.84% in average. We also show that TCLiVi is self-adaptive in different scenarios.**

***Index Terms*—Live video streaming, reinforcement learning, joint optimization, adaptive transmission control.**

I. Introduction

NOWADAYS, video streaming contributes the largest proportion to network traffic, and the volume of HTTP-based video streaming traffic constitutes almost 58% of the total downstream traffic [1], [2]. Many previous works have attempted to enable high-quality video delivery due to the demand for more bandwidth and higher video resolution [3], [4]. Among the video streaming services, live video streaming has becomes increasingly popular for many reasons, such as the explosive growth of live broadcast platforms, which has high entertainment value and intensive social interaction capabilities in real-time [5].

In live streaming systems, video quality, stability, rebuffering time, and end-to-end latency play important roles in the QoE optimization. Buffering ratio has a significant impact on user engagement (in terms of the total play time and number of videos viewed) in live streaming, and the average bitrate affects live video content more greatly than Video-on-Demand (VoD) content [6]. The quality of live video has a great impact on user behaviors and influences the revenues of the service providers. Thus, QoE optimization is important in live streaming video.

However, it faces several great challenges: 1) heterogeneous user devices present increased difficulties in managing live video transmissions, e.g., some broadcasters may use professional cameras whereas others only common smartphones; and some viewers may use PCs connected to fiber optic networks whereas others only smartphones with poor wireless connections; 2) Unlike VoD, live streaming applications produce videos in real-time, which is more dynamic; 3) The interactions between the broadcasters and the viewers demand higher QoE requirements, especially the end-to-end latency.

Currently, live video platforms use static default bitrate without considering the network and transmission conditions. For example, Betta and Yingke, which are the most popular live streaming applications in China, use the default high video bitrate. This is inconvenient to the users as they have to decrease the video bitrate manually when suffering from video playback stall, and increase the video bitrate when the transmission conditions improve. To improve the QoE, we need a solution that can manage the transmissions adaptively according to the network conditions and user device information, to provide a personalized live video stream.

Several previous works have attempted to improve the QoE performance. For example, the most direct method to improve the QoE is by improving the network infrastructure, such as by installing more Content Delivery Network (CDN) nodes [7]. However, the infrastructure can not be upgraded overnight and more CDN nodes need more investments. Many rule-based algorithms use only the buffer or throughput prediction information to formulate decisions, which are mainly used in VoD scenarios and are not self-adaptive when being applied to other scenarios. Some other algorithms for live streaming scenarios use

Manuscript received October 16, 2019; revised February 16, 2020; accepted March 28, 2020. Date of publication April 6, 2020; date of current version January 29, 2021. This work was supported in part by the National Key R&D Program of China under Grants 2018YFB1800302 and 2018YFB1800805, in part by the National Natural Science Foundation of China under Grants 61772345 and 61902258, and in part by the Major Fundamental Research Project in the Science and Technology Plan of Shenzhen under Grant JCYJ20190808142207420. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zixiang Xiong. *(Corresponding author: Shu Yang.)*

Laizhong Cui, Dongyuan Su, Shu Yang, and Zhong Ming are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Guangdong Laboratory of Artificial-Intelligence and Cyber-Economics(SZ), Shenzhen University, Shenzhen 518060, China (e-mail: cuilz@szu.edu.cn; 1900271016@email.szu.edu.cn; yang.shu@szu.edu.cn; mingz@szu.edu.cn).

Zhi Wang is with Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: wangzhi@sz.tsinghua.edu.cn).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2020.2985631

complex encoding mechanisms or require the cooperation between the server and the user, which may potentially increase the computational complexity.

Markov Decision Process (MDP) is a popular model for adaptive control in the Dynamic Adaptive Streaming over HTTP (DASH) system in time-varying channel conditions [8], [9]. DRL based on the MDP model, is widely used in different scenarios, such as gaming [10], [11], dialogue generation [12], and navigation [13]. A representative recent advanced system, Pensieve [14] generates Adaptive Bitrate (ABR) algorithms using DRL in VoD scenarios. Inspired by the previous works, we leverage DRL to cope with the challenge presented by live streaming transmission scenarios.

In our DRL-based solution (TCLiVi), we utilize the frame-level information to finely capture the transmission dynamics, and create a chunk-level control at the Group of Picture (GOP) boundary. GOP is an inseparable compression unit that consists of many video frames. Specifically, we use the information of the latest 48 video frames (download time interval, video frame size, rebuffering time, buffer size and end delay) as the input for the neural network. GOP can capture the dynamic of the transmission conditions and the underlying transmission pattern, and generate a joint decision after considering both the bitrate level and the target buffer level. By combining the deep neural network with a state-of-the-art RL algorithm Actor-Critic (AC) algorithm as well as other techniques, such as cross entropy and mini-batch training, our model can produce a balanced result in terms of the video quality, playback stall frequency, and latency between the broadcasters and viewers.

We select important observations and suitable representation to construct the state. We then construct two neural networks as the policy and the state-value estimator respectively, and use the AC framework [15] to train the neural networks. The trained policy network can serve as an inference model that can make the suitable transmission control decisions in a real-time streaming system. We further improve the approach by implementing many enhancements. For example, we adaptively decrease the value of the temperature parameter in the soft-max layer and add an entropy entry to the policy gradient formula to maintain a balance between exploration and exploitation. Moreover, we utilize the multi-step return concept [16] to reduce the training time, and also use a fixed double Q-network [17] to improve the training process of the critic network. We then conduct extensive experiments to demonstrate the performance of TCLiVi under various settings including different scenarios, learning rate, step size, etc. We also implement different algorithms in different live streaming scenarios to prove the generalization ability of TCLiVi. The compared algorithms include Double DQN, MPC, and Buffer-based algorithms.

Our study makes the following contributions: 1) we construct an accurate representation of the state, action, reward, and neural network and design a DRL-based training process to generate an inference model; 2) we add several enhancements to the basic approach, such as adaptively decreasing the temperature parameter value in the soft-max layer, and utilizing entropy, multi-step return and fixed Double Q-network; 3) we conduct extensive experiments to tune the hyperparameters and compare the performance of the different algorithms.

The rest of our paper is organized as follows. We review related works in Section II. In Section III, we introduce the live streaming transmission scenarios and define the relevant problems. In Section IV, we represent our proposed approach, including the related theoretic preliminary, the basic solution, and the enhancement of the solution. In Section V, we introduce experiment results, including the description of dataset, the hyperparameter tunning process, the sensitivity analysis, and performance comparision among different algorithms. Finally, in Section VI, we conclude our work.

II. Related Works

Recently, many works are proposed to improve bitrate adaptation methods, that consider playback buffer occupancy, network bandwidth, or a combination of the two. Buffer occupancy information is used for bitrate adaptation in [18], [19]. Algorithms introduced in [20]–[23] use the predicted future network bandwidth to adjust video bitrate. To avoid the fine-tuning of parameters, the Adaptive Forgetting Factor method is used for throughput estimation [24]. Both buffer occupancy and predicted future bandwidth are used for bitrate adaptation in [25], [26].

However, the network and transmission conditions are dynamic, which make the parameter configurations difficult. Estimating client-side bandwidth is hard, leading to variable and low-quality video, which has a great impact on algorithms based on accurate bandwidth estimation [27]. A two-phase algorithm is proposed in [28], which utilizes a rate-based and a buffer-based approach respectively and switches between the two phases.

Besides VoD, live video streaming makes the problem harder due to its liveness and interactions between broadcasters and users. A cooperative server-client DASH system is constructed in [29], where the server utilizes the bandwidth information collected from users to determine the encoding bitrate, and the clients select a segment to increase the bandwidth utilization. However, the solution is not scalable in large streaming systems where the bandwidth information varies, and the centralized server could become the hot spot to deal with enormous information. Adjusting the bitrates of Scalable Video Coding (SVC), the bitrate can be adjusted more frequently to adapt to severely fluctuated network conditions [30].

However, the abovementioned rule-based algorithms could not meet the demands of all users, because of different preferences and different network conditions. A broadcast-based rate adaption for DASH is formulated as an MDP-based optimization problem in [8], which can create a unique policy for each player and can be updated in real time. Another example for utilizing the MDP model for rate adaptation for DASH is [9].

DRL, a technology based on the MDP model, has becoming increasingly popular recently, and has attracted much attention in different fields such as games and robotics [31]. With the combination of deep learning [32] and RL [33], DRL can effectively capture the complex and stochastic environments and is a promising approach for solving complex problems such as intelligent live video broadcasts. Pensieve [14] adaptively adjusts the bitrate based on DRL in a VoD scenario. The adaptive bitrate algorithm based on DRL is deployed on the edge node to improve QoE in [34]. D-DASH [35] combines feed-forward and recurrent

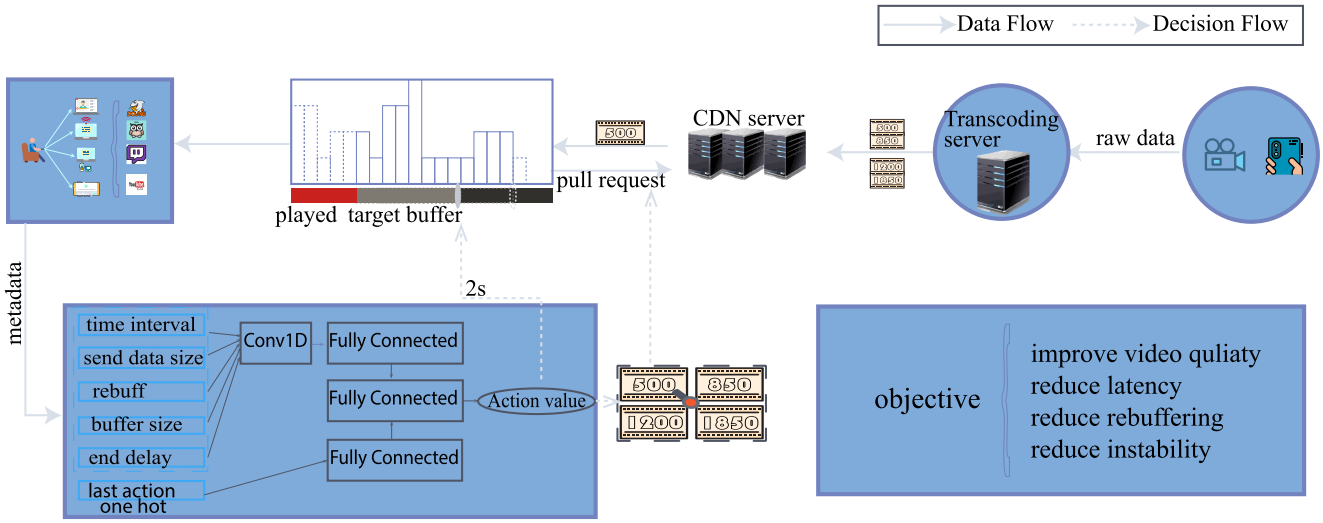


Fig. 1. Live Streaming Scenario Introduction.

TABLE I
MAJOR NOTATIONS USED IN THE PAPER

Notation	Meaning
λ, μ, ν	The weights of rebuff, latency and instability
Z, F	The bitrate and target buffer
K	the overall time step of a video session
γ	The discounted rate in the calculation of return
θ, ω	The parameter vectors of actor network and critic network
T	The temperature parameter of soft-max function
$H(s)$	The entropy entry in state s
η	The learning rate

neural network with the advanced strategies to optimize the QoE of DASH. 360SRL [36] employs sequential RL to overcome the limitations of directly applying RL in the tile-based 360 video streaming. The indroduced sequential ABR decision structure reduces the decision space, making it feasible to be employed with RL. DRL360 framework [37] employs DRL-model to allocate rates for the tiles of the future video frames. To schedule the large-scale dynamic viewers among different CDN providers, a DRL-based scheduling approach is proposed to adapt to the variation of both viewer traffic and CDN performance [38].

However, different from VoD where video content is stored in the CDN, live video streaming viewers watch the video with end-to-end communications between them and the broadcasters. Thus, the optimization in live video needs more global information than VoD. In this study, we integrate DRL into live video streaming, to capture high dimensional streaming metadata for improving the QoE. AC [15] algorithm is widely used for solving the multimedia transmission control problem. It parameterizes policies according to soft-max in action preferences. Combining entropy and temperature parameter, it is naturally to balance exploration and exploitation during training procedure. With continuous policy parameterization and Policy Gradient Theorem, a stronger convergence guarantee can be achieved. Previous works such as [34], [37], [38] leverage AC algorithm for video transmisson control. In this paper, we also use AC as the fundamental algorithm for the QoE optimization problem. By impelemting enhancements such as adaptively decreasing the temperature parameter of soft-max layer, utilizing entropy, multi-step return and fixed Double Q-network, our method significantly improves the performance. Our solution jointly adjusts the bitrate and minimum playback buffer size, and can effectively adapt to different scenarios, and thus is easy to deploy in real applications.

III. LIVE VIDEO TRANSMISSION CONTROL DESCRIPTION

A. Transmission Procedure and System Model

In this section, we describe the live video streaming scenarios and show the challenges of designing an intelligent control mechanism. We conclude the major notations through this paper in Table I. Fig. 1 shows the main components of the live streaming and the data transmission procedure. During a live video streaming broadcast, the broadcaster records real-time video frames with his smartphone or camera, and then uploads the captured raw video frames to the transcoding server. The transcoding server converts the raw video chunks (comprising several video frames) into different formats and resolutions of the videos, for example, smooth, low definition, high definition, and ultra-clear, and transports them to the CDN, which holds all the versions of the video chunks. Finally, the user equipment initiates a pull request for a specific version of the video frames, which is decided by our transmission control module.

The transmission control module aims to improve the video quality and stability, as well as avoiding the video playback stall and reduce the latency between the viewers and the broadcaster. Low latency is important due to the need for interactions between the viewers and the broadcaster in live video streaming scenarios, which in turn require a sufficient low buffer occupancy. Our challenges include the huge volume of the video content, complex heterogeneous networks, an unstable network topology, the low buffer occupancy requirement, dynamic video sources as well as various types of user equipments. It motivates us to design a DRL-based algorithm to keep a balance among the different factors.

We consider a frame-level broadcast-and-play system, in which the user equipment repeatedly sends the pull requests to the nearest CDN. Upon receiving the request, the CDN responds with a video frame or an empty signal (implicit frozen). After each interaction, the client-side equipment stores the transmission metadata, such as the download time interval, video frame size, and buffer size, for future decision making. With these metadata, the client can control the transmission by jointly switching the video bitrate and target buffer level when permitted, e.g., receiving the decision flag at the boundary of GOP. The video bitrate corresponds to the quality of the video frame, and the target buffer is the buffer level that the video players attempt to maintain, which determines two aspects of buffer occupancy. First, the target buffer is the minimum value of the playback video size required for the player to recover from a stall. Second, it is associated with the slow and fast play thresholds, which slows down the video player to delay the arrival of video playback stall and speeds up the video player to reduce the latency respectively. For example, if the slow play threshold, the target buffer, and the fast play threshold are 1 s, 2 s and 3 s, respectively, the player will trigger the slow play if the buffer size is less than 1 second, and will trigger the fast play if the buffer size is more than 3 seconds. In addition to the slow play and the fast play mechanisms, there is a frame skipping mechanism which skips some old video frames when the buffer size is more than 7 seconds, to reduce the latency between the broadcaster and the viewer [39].

B. Problem Definition and Formulation

During a live streaming process, the most common influence factors of the QoE include the video bitrate, video playback stall, end-to-end latency and its instability which is reflected by the bitrate switching frequency. Specifically, as the playback stall frequency, the end-to-end latency, and the instability decrease, and video bitrate increases, the QoE will increase. We use QoE model as defined in Equations (1) and (2), which defines our objective as the weighted sum of these four sub-objectives in the live stream broadcast procedure, in which the video bitrate has a positive weight while instability, latency and stall have negative weights. We aim to bring a balance among these conflicting sub-objectives while considering the preference of the users captured by the weight parameters. Every video frame has a QoE of frame, and at the boundary of each video group, there is an additional QoE of group, which represents the video instability.

$$\begin{aligned}\mathrm{QoE_t(Frame)} = {} & \mathrm{play\_time\_duration} * \mathrm{bitrate} \\ & - \lambda * \mathrm{rebuff} - \mu * \mathrm{latency} \end{aligned} \tag{1}$$

$$\mathrm{QoE_t(Group)} = -\nu * \mathrm{switching} \tag{2}$$

where λ, μ, ν represent the weights of rebuffering, latency, instability respectively, and their configurations will be elaborated in Section V.

Our work aims at leveraging the history information to generate proper transmission control decisions. The decisions are represented by a sequence of bitrates and target buffer levels. Our ultimate objective is to gain a high long-term QoE, i.e., optimize the sum of the QoE during a live streaming session to improve the video quliaty and decrease the rebuffering, instability and latency.

With this objective, we can formulate the problem as a QoE maximization problem. Let K denote the number of total time steps, Z denote the selected bitrate, F denote the target buffer level, and H the history information, and then the QoE optimization problem can be represented as:

Problem: At time step t, given H_t, find a decision pair $\{Z_t, F_t\}$, so that $\sum_{t=1}^{K} \mathrm{QoE_t}$ is maximized when a video session finishes. Here,

$$\mathrm{QoE_t} = \mathrm{QoE_t(Frame)} + \mathrm{QoE_t(Group)}\mathbb{1}(\mathrm{At\_Boundary})$$

and the indicator function $\mathbb{1}(\mathrm{At\_Boundary})$ is equal to 1 when the frame is at the boundary of a GoP, otherwise 0.

IV. Methodology Based on DRL

A. DRL Preliminary

In RL paradigm [33], the interactions between an agent and the environment are modeled as the MDP [40], [41], in which the current state encapsulates all the information for future decision making without considering the history states. The entire MDP is represented by a sequence of states, actions and rewards, $\{S_t, A_t, R_{t+1} | t \in [0, T)\}$. State is the observation of the environment, which captures the most relevant information of the environment. Action is taken by the agent to influence the environment, which leads to a state transition. The reward is the instant feedback of the environment after it receives the action.

Greedily choosing the best action to maximize the reward may not reach the global optimum. Return, G_t, as defined in Equation (3), is the long-term future cumulative discounted reward from time step t, γ is the discounted rate. Value function, $\nu_\pi(s)$, as defined in Equation (4), is the expected return at a specific state under policy π, which can represent the goodness of a state.

$$G_t \doteq \sum_{k=t}^{\infty} \gamma^{k-t} R_{k+1} \tag{3}$$

$$\nu_\pi(s) \doteq E_\pi[G_t | S_t = s], \quad \forall s \in \mathcal{S} \tag{4}$$

With Bellman Equation [41], we can represent the state-value of the current state, $\nu_\pi(s)$, as the combination of the instant reward, R_{t+1}, and the value of possible successor state, $\nu_\pi(S_{t+1})$, with discount factor γ. It can be represented as:

$$\nu_\pi(s) = E_\pi[R_{t+1} + \gamma \nu_\pi(S_{t+1}) | S_t = s], \quad \forall s \in \mathcal{S} \tag{5}$$

In RL, we use the difference of the target value and the current state-value to train the model by traditional supervised learning methods, such as gradient descent of Mean Square Error [42].

The policy function, $\pi(a|s)$, generates a probability distribution of different actions at each state. With Policy Gradient Theorem [43], we can derive a gradient vector to update the policy function in the desired direction and at suitable step size, which serves as the approximation of the gradient of the performance measure, $\nabla_\theta J(\theta)$. Specifically, the derived gradient vector is the combination among the gradient of the logarithm of the policy, $\nabla_\theta ln\pi$, and the error between Q-value and baseline, which can

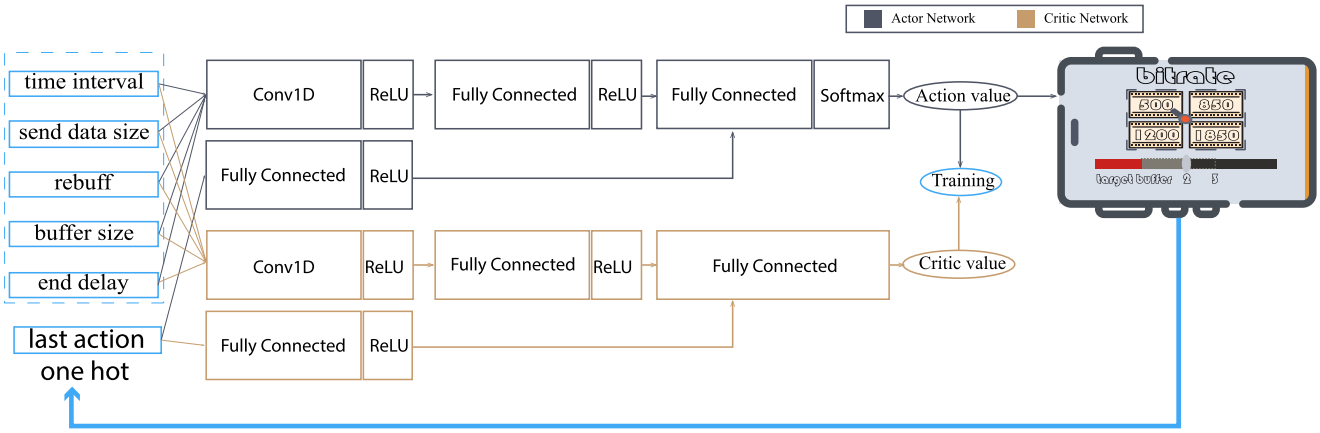


Fig. 2. Actor Critic Network.

be expressed as:

$$\nabla_\theta J(\theta) \propto E_\pi[(q_\pi(S_t, A_t) - b(S_t))\nabla \ln \pi(A_t|S_t, \theta)] \quad (6)$$

In the traditional RL, the value function and policy function are calculated and stored with table. However, this kind of one-to-one mapping is infeasible in our problem, due to the tremendous scale of the state space and the action space. Deep Neural Network is suitable to remedy this problem, since it uses limited parameters to approximate the mapping function [44]. Specifically, the number of parameters is relatively stable without depending on the size of state space and action space, which makes it possible to solve a continuous problem.

B. Solution in Detail

State space: we stack the transmission information of the last 48 video frames into the state. The transmission information includes: video frame download time length, frame size, playback stall time, buffer occupancy, end delay, and bitrate level, which is normalized into the same scale fluctuating near zero. Specifically, download time length and frame size encode the throughput information of the network, and frame size and video bitrate contain the video source information. The playback stall time and buffer size tell us how confident we are to increase the video quality without causing a video playback stall. For example, if the buffer size is large enough, the algorithm can confidently increase the video quality. The end delay and bitrate information help the algorithm to reduce latency between broadcasters and viewers and reduce the video instability.

Action space: the action space consists of bitrate Z and target buffer F. Specifically, we assume that there are four versions (500 kbps, 850 kbps, 1200 kbps, and 1850 kbps) of video frames, and two target buffer levels (2 s and 3 s) and the associated slow play and fast play thresholds are $\{1, 3\}$ and $\{2, 4\}$ second respectively.

Reward function: to optimize the overall QoE, the system reward at time step t, R_t, expressed as Equation (7), is calculated based on the QoE of each frame defined in Equations (1) and (2). With this reward definition, we aim to balance four sub-objectives according to the weights to achieve a long-term high QoE.

$$R_t = \text{play\_time\_duration} * \text{bitrate} - \lambda * \text{rebuff} - \mu * \text{latency} - \nu * \text{switching} * \mathbb{1}(\text{At\_Boundary}) \quad (7)$$

To extract the inner patterns of the transmission conditions, we design two neural networks to serve as the actor and the critic role respectively, which are depicted by Fig. 2. The two networks take the system state, such as the sequence of history buffer size, frame size and end delays, as input. For the first layer, all input data is processed by convolutional neural networks (CNN) [45]. Then the processed data further goes through linear transformations and nonlinear transformations which are represented by fully connected networks and ReLU functions [46], $\mathrm{ReLU(x)}$, defined as follows:

$$\mathrm{ReLU(x)} = \max\{0, \mathrm{x}\} \quad (8)$$

where x is the output value of the previous layer in the network.

To approximate the current state-value, the critic network was ended by a fully connected network with one neuron, which represents the expected long-term cumulative reward from current state $\nu_\pi(s)$. We use the actor network to map the state into a policy, which is the probability distribution among 8 combinations of birtates Z and target buffers F. To generate the probability distribution, the actor network was ended by a soft-max layer with eight neurons, which represents the parameterized policy, $\pi(a|s)$, as follows:

$$\pi(a|s) = \frac{\exp(h(s,a))}{\sum_b \exp(h(s,b))} \quad (9)$$

where $h(s, a)$ represents the action preference of action a at state s, which is represented by the actor network before the soft-max layer. The decision making procedure in our live streaming system is depicted as the bottom-left box of Fig. 1.

We employ the state-of-the-art RL framework, AC algorithm [15], to train the neural networks. For the critic network, the loss function is the Mean Square Error (MSE) between the updated target and the current state-value, which is expressed as:

$$\mathcal{L}(\nu) = [r + \gamma\nu(s') - \nu(s)]^2 \quad (10)$$

where r is the instantaneous reward calculated as Equation (7), $r + \gamma\nu(s')$ is the updated target according to the Bellman Equation as expressed in Equation (5). At time step t, given $\gamma = 1$, the output of this network with the input S_t, i.e., $\nu(S_t)$, is an estimation of $\sum_{i=t}^{K} \mathrm{QoE_i}$, and $R_{t+1} + \gamma\nu(S_{t+1})$ is the new estimation of $\sum_{i=t}^{K} \mathrm{QoE_i}$ under the current policy. With this loss function, the critic network can be trained to output the future long-term cumulative discounted QoE.

For the actor network, the loss function is derived by the Policy Gradient Theorem, as expressed in Equation (6). The gradient vector of the actor network can be represented as:

$$\nabla J(\theta) = [r + \gamma\nu(s') - \nu(s)]\nabla \ln \pi(a|s) \tag{11}$$

To illustrate the intuition of this equation, we can rewrite $\nabla \ln \pi(a|s)$ as $\frac{\nabla\pi(a|s)}{\pi(a|s)}$, where the numerator, i.e., the gradient of the actor network, serves as the basic updated direction, and the denominator represents the occurrence frequency of the state-action pair. The sign and magnitude of the advantaged value, $r + \gamma\nu(s') - \nu(s)$, controls the final updated direction and the step size respectively. The positive sign of advantaged value means that the long-term QoE, $\sum_{i=t}^{K} \mathrm{QoE_i}$, is improved, so that the updated direction is the same as the gradient, and the probability of action a can be improved.

We use the backpropagation algorithm to calculate the gradient with respect to all the network parameters, and use RmsProp optimizer [47] to update the two neural networks. For the actor network, the optimizer can be espressed as:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \tag{12}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{13}$$

where g_t is the gradient vector at time step t, $E[g^2]_t$ is the exponential moving square average gradient vector, θ denotes the parameter vector of the actor network, while η and ϵ are two hyperparameters, representing the learning rate and an infinitely small value respectively. To update the critic network, the parameter vector of the actor network, θ, is replaced by the parameter vector of the critic network, ω.

During the training procedure, we use a mini-batch training mechanism and combine the entropy with the loss function. By the mini-batch training, we collect and store a small sequence of state transitions as experience, and then train the network with the average loss at one time, which accelerates the convergence. The cross entropy, $-\sum_a \pi(a|s) \ln \pi(a|s)$, represents the randomness of selecting an action. By giving high weight to the cross entropy entry at the beginning of the training process, we effectively increase the exploration of the policy, which helps the optimization algorithm escape local minimum.

C. Enhancement of the Solution

In RL, it is important to strike a balance between exploration and exploitation. With extensive exploration, the algorithm can avoid falling into local optimum while the random attempts may harm the performance. Inspired by Pensieve [14], we utilize the entropy entry to adjust the degree of exploration, and the calculation of the entropy is shown as Equation (14). In addition, we exponentially decrease the temperature parameter of the soft-max layers in the actor network, i.e. exponentially decrease T in Equation (15).

TABLE II
TRAINING PARAMETERS INFORMATION

parameter name	value
actor learning rate	1e-2
critic learning rate	1e-3
discount factor	0.99
return step size	4
entropy weight	0.9 to 0.05
soft-max temperature	20 to 1
batch size	200
conv1d {outchannel,kernel,stride}	{32, 12, 6}
fc1 out neuron	128
action_fc out neuron	32

$$H(s) = -\sum_a \pi(s,a) \ln \pi(s,a) \tag{14}$$

$$\pi(s,a) = \frac{\exp(h(s,a)/T)}{\sum_b \exp(h(s,b)/T)} \tag{15}$$

One step Temporal Difference (TD) learning and Monte Carlo (MC) learning are two extremes for the calculation of the return. The former is biased whereas the latter has high variance. To enhance the proposed approach, we use a multi-step TD learning instead. In this case, the updated target, i.e. the n-step return, is calculated as:

$$G_t = \left(\sum_{k=0}^{n-1} \gamma R_{t+k+1}\right) + \gamma^n \nu(S_{t+n}) \tag{16}$$

We use a double critic network to keep a relatively stable state-value and hence a relatively stable updated target. We denote the parameter vectors of the actor network, the critic network, and the double critic network as θ, ω, and ω^-, respectively. We update the critic network ω every time step, and periodically update the double critic network, i.e. $\omega = \omega^-$, every several epochs. The updated rules are represented as:

$$\theta_{t+1} = \theta_t + [G_t - \nu(S_t;\omega_t)]\nabla_{\theta_t} \ln \pi(S_t, A_t; \theta_t) + \nabla_{\theta_t}\left[-\sum_a \pi(S_t,a) ln\pi(S_t,a)\right] \tag{17}$$

$$G_t = R_{t+1} + \gamma\nu(S_{t+1};\omega_t) \tag{18}$$

$$\omega_{t+1}^- = \omega_t^- - \nabla_{\omega^-}[G_t^- - \nu(S_t;\omega_t^-)]^2 \tag{19}$$

$$G_t^- = \left(\sum_{k=0}^{n-1} \gamma^k R_{t+k+1}\right) + \gamma^n \nu(S_{t+n};\omega_t^-) \tag{20}$$

Combining all the mentioned elements, our final training procedure is shown as Algorithm 1. In the algorithm, the action is an eight-dimensional vector encoding the four possible choices of bitrate levels and the two possible choices of target buffer levels. The reward is the average frame-level reward in one video chunk. Some important hyperparameters are summarized in Table II.

Algorithm 1: Tranmission Control in Live Streaming

while $epoch < num\_epoch$ **do**
  initialize state s
  select action a according to $\pi(s, \cdot; \theta)$
  while s is not terminal state **do**
    take action a for the following video frames until reaching the chunk boundary
    update next state s' and reward r (average inner and inter group reward)
    store transition $\{s, a, r, s'\}$ into training buffer
    if $size(training\_buffer) \geq batch\_size$ **then**
      for $t = 0 \rightarrow (batch\_size - n)$ **do**
        $G_t^- \leftarrow (\sum_{k=0}^{n-1} \gamma^k R_{t+k+1}) + \gamma^n \nu(S_{t+n}; \omega_t^-)$ ▷ calculate n-step return
        $G_t \leftarrow R_{t+1} + \gamma \nu(S_{t+1}; \omega_t)$ ▷ calculate one-step return
        $\theta_{t+1} \leftarrow \theta_t + [G_t - \nu(S_t; \omega_t)] \nabla_{\theta_t} \ln \pi(S_t, A_t; \theta_t) + \nabla_{\theta_t} [-\sum_a \pi(S_t, a) ln \pi(S_t, a)]$ ▷ update actor network parameters
        $\omega_{t+1}^- \leftarrow \omega_t^- - \nabla_{\omega^-} [G_t^- - \nu(S_t; \omega_t^-)]^2$ ▷ update double critic network parameters
      end for
    end if
  end while
  $\omega \leftarrow \omega^-$ ▷ transfer double critic network parameters to critic network parameters
end while

D. Double DQN for Comparison

To further compare TCLiVi with other advanced DRL-based algorithms, we implement the Double Deep Q-Learning [17]. We use the same representation of state, action, reward and the same training procedure with the same neural network structure in our implementation of the Double Deep Q-Learning algorithm. In Double Deep Q-Learning, there are two Q-networks used to represent state-action value function. The second Q-network is used to avoid the overoptimism due to estimated errors of the max operation in Q-learning. Equations (21) and (22) are the updated rules used to train the neural networks. The second equation shows how to decouple the selection and the evaluation of an action.

$$\theta_{t+1} = \theta_t + \alpha(Y_t^{\mathrm{DoubleDQN}} - Q(S_t, A_t; \theta_t)) \nabla_{\theta t} Q(S_t, A_t; \theta_t) \quad (21)$$

$$Y_t^{\mathrm{DoubleDQN}} = R_{t+1} + \gamma Q(S_{t+1}, \mathrm{argmax_a Q(S_{t+1}, a; \theta_t), \theta_t^-}) \quad (22)$$

V. Simulation Result and Analysis

A. Experiment Setup

We evaluated our algorithm through trace-driven simulations. The simulation data-set consists of two parts: the network traces and the video traces. The video traces record the video frame size information in different scenarios, which can emulate the dynamic of video source. The network traces include network bandwidth information sampled in different network conditions for 49 minutes with the interval of 0.5 seconds.

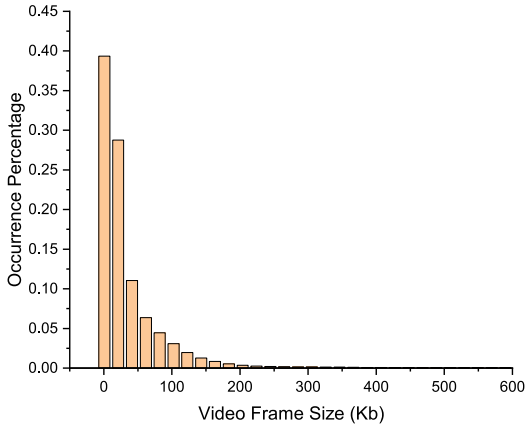


Fig. 3. Overall Video Size Distribution.

The video traces consist of the timestamp when the video frame reaches the CDNs, as well as the frame size, and the decision flag, which are sampled from the transcoding servers and CDNs. They also include the video frame size information at four bitrates: 500 kbps, 850 kbps, 1200 kbps and 1850 kbps, and include game, sport, and indoor live streaming scenarios such as the Asian Cup match between China and Uzbekistan, etc. The mean and the standard deviation of all the tested scenarios are: 20.048 ± 46.148 kb, 34.079 ± 75.435 kb, 48.109 ± 103.465 kb and 74.189 ± 162.600 kb. The probability distribution of the overall video traces is showed in Fig. 3. More detailed statistic information is given in Table III. It includes the statistic data of four levels of video bitrate, and the statistic data include average frame size (bit), confident interval, quantile, etc.

The network traces consist of the timestamps and the network bandwidths, which are sampled from Wi-Fi and LTE to emulate strong, medium, and poor network conditions. The network bandwidth of the data-set is 1.59 Mbps in average with a deviation of 1.22 Mbps, ranging from 0.2 Mbps to 11.68 Mbps, and it is lower than 1.22 Mbps for half of the network traces. The probability distribution of the overall network traces is shown in Fig. 4. Table IV describes the detailed statistic information, such as mean throughput (Mb), confident interval, quartile, etc.

The QoE setting can be adapted to different people with varying preferences by tunning the weights of the sub-objectives. Extensive literature is available on QoE design to improve user engagement [48] and on how different objectives impact user engagement. For the live video content, buffering ratio and average bitrate have a significant impact on the user engagement [6]. In our simulation, we adopt a combination of {1.5, 0.005, 0.02} for weights {λ, μ, ν} respectively [49], which means that we concentrate more on the video quality and rebuffering time. The detailed parameter configurations are listed in Table V. The performance has been further validated in Section V-C2, which shows that this QoE setting can significantly improve the average bitrate and rebuffering time.

TABLE III
VIDEO TRACES STATISTIC DATA

index name	500kbps	850kbps	1200kbps	1850kbps
trace number	1.64354e6	1.64354e6	1.64354e6	1.64354e6
mean	20048.86272	34079.0906	48108.93056	74189.25769
standard deviation	46147.57312	75435.44491	103465.90494	162600.74774
SE of mean	35.99637	58.84171	80.70624	126.83304
Lower 95% CI of Mean	19978.31107	33963.76288	47950.74911	73940.66932
Upper 95% CI of Mean	20119.41436	34194.41833	48267.112	74437.84606
Minimum	120	144	168	272
1st Quartile(Q1)	3192	5664	8288	13696
Median	7640	13024	18744	30744
3rd Quartile(Q3)	20680	36312	52544	79832
Maximum	1.03672e6	1.6064e6	2.16816E6	3.93783E6

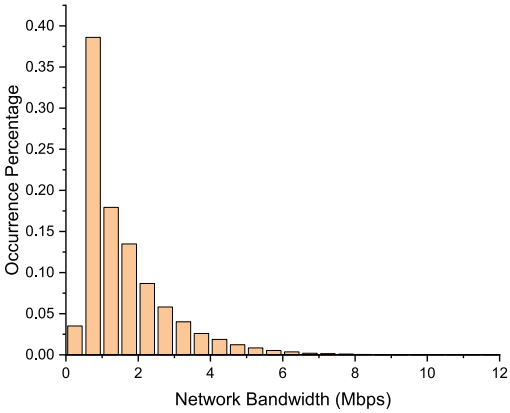


Fig. 4. Overall Network Bandwidth Distribution.

TABLE IV
NETWORK TRACES STATISTIC DATA

index name	network statistic value
trace number	117600
mean	1.59466
standard deviation	1.21561
SE of mean	0.00354
Lower 95% CI of Mean	1.58771
Upper 95% CI of Mean	1.60161
Minimum	0.2
1st Quartile(Q1)	0.69113
Median	1.20619
3rd Quartile(Q3)	2.07476
Maximum	11.67891

B. Parameters and Training Procedure Tunning

To finely tune the hyperparameters, such as the step size of the n-step return, discount factor gamma, and learning rates of actor and critic network, we conducted extensive experiments whose results as shown in Fig. 5. The left two subfigures show the full training process, whereas the right two subfigures enlarge the critical part of the left figure, i.e., after ignoring some initial epochs and hyperparameter combinations that have bad QoE. From Fig. 5(a), we can see that only the hyperparameter combination of {2, 0.99, 1e-3, 1e-4} diverges, while the other combinations, such as {2, 0.8, 1e-2, 1e-3}, converge. From Fig. 5(b), we can further discover that the combination of 1e-2 and 1e-3 for the learning rate of the actor network and the critic network has the best performance, Hence, we can conclude that proper learning rates play a more important role than the step size and the gamma parameters. In Fig. 5(c) and (d), we can see that the most suitable candidate for gamma is 0.99, when the learning rates of the actor network and critic network are set as 1e-2 and 1e-3 respectively.

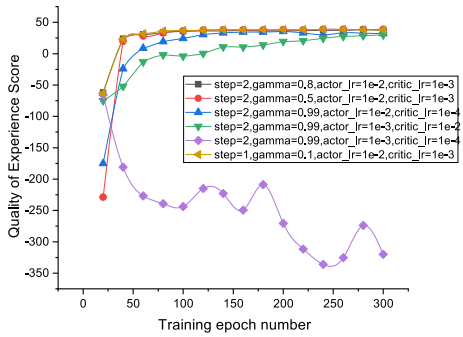
TABLE V
DETAILED WEIGHTS AND PLAYER SETTINGS

parameter name	value
λ(rebuff)	1.5
μ(latency)	0.005
ν(switching)	0.02
step size	4
frame rate	25 fps
frame length	0.04 s
bitrate type	{500, 850, 1200, 1850} kbps
{slow play, target buffer, fast play}	{1, 2, 3} s, {1.5, 3, 4} s

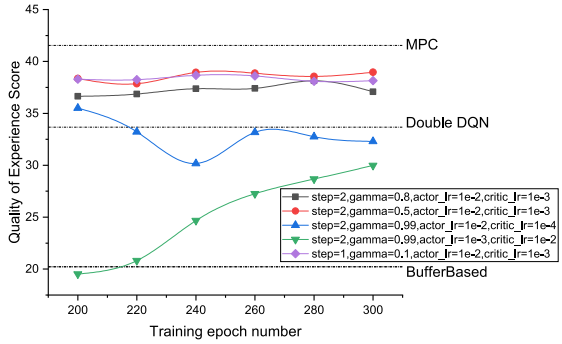
We conducted experiments with different settings of the step size for the calculation of return G_t, i.e. the value of n in Equation (16), to test the efficiency of the enhancement and find the best value for the step size. The experiment results are depicted in Fig. 6. Fig. 6(a) shows that all the step size settings less than 15 converge to a relative high QoE score, whereas a bigger step size can reduce the performance. Fig. 6(b) is the partially enlarged view of the left subfigure. From the subfigure, we can see that, the step size of 10 converges most quickly, whereas the step size of 4 converges to the highest QoE score when the training process is stable. We calculated the average QoE score when the algorithm converges, i.e., from epoch 420 to 500. The average QoE scores are 46.35676, 47.00095, 48.30249, 46.53466, 46.85764 and 24.36822 when the step size are 1, 2, 4, 8, 10, and 15, respectively, which means that the best performance is seen when the step size of 4. To show how significant the improvement is, we calculate the average rebuffering and the end delay for the one-step and the four-step settings. The result shows that the rebuffering time of the one-step setting is 43.97% more than that of the four-step setting, and the end delay of the one-step setting is 16.58% more than that of the four-step setting. As the four-step setting has the best performance, we use this setting for the later analysis.

We further tested the learning rate in the four-step size setting, which is shown in Fig. 7. The result suggests that the best learning rate for the actor network is 1e-2, whereas there are several choices for the learning rate of the critic network.

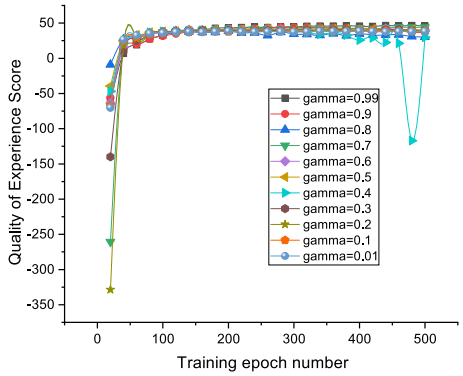
In the above-mentioned experiments, we increased the step size of the return of the critic. We also verified the performance



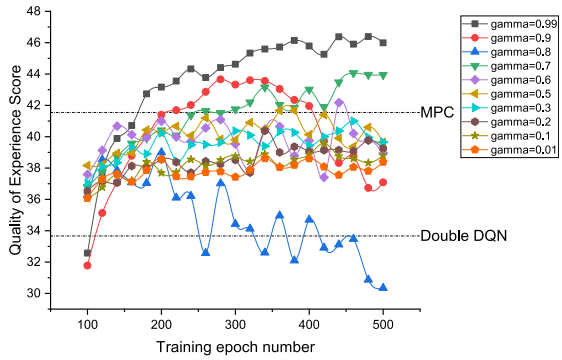
(a) Tunning Step Size, Gamma and Learning Rate



(b) Sub graph of Fig 5(a)

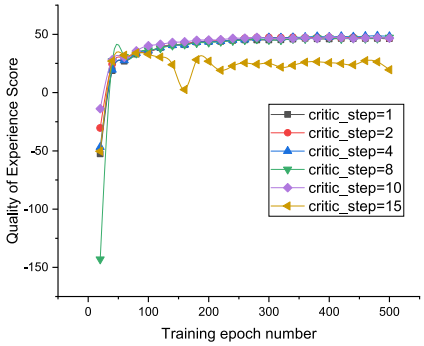


(c) Tunning Gamma

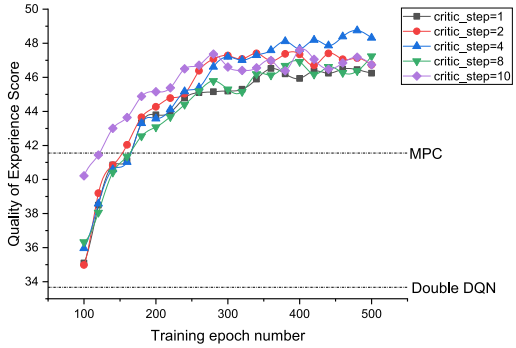


(d) Sub graph of Fig 5(c)

Fig. 5. Hyper Parameters Tunning.



(a) Tunning Critic Step Size



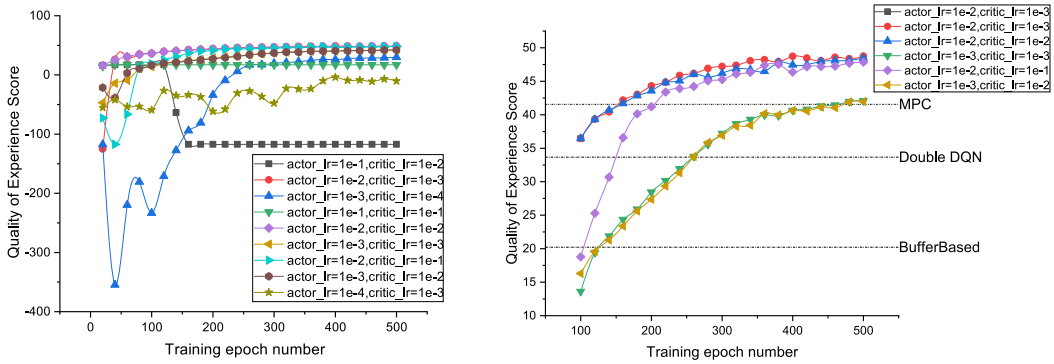
(b) Sub graph of Fig 6(a)

Fig. 6. The Performance of Different Critic Value Step Size.

when increasing the step sizes of the return of both the actor and the critic, whose results are shown in Fig. 8. However, the performance is not improved in this setting as the figure shows.

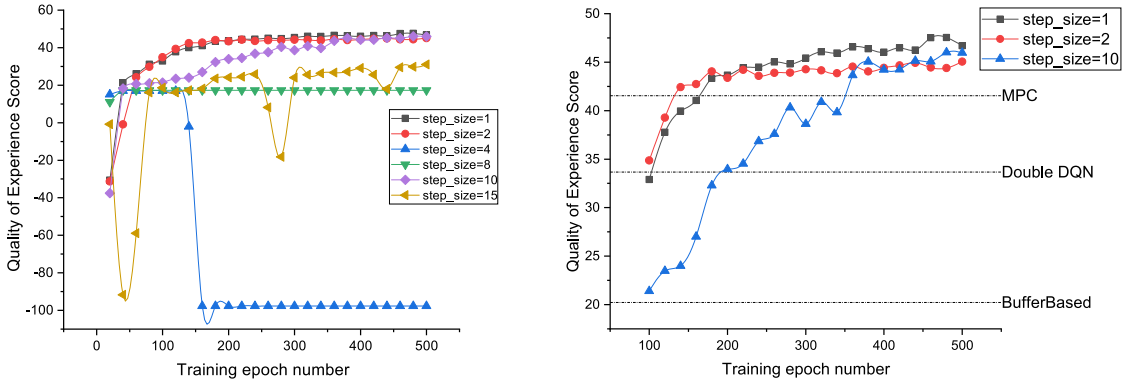
C. Results and Analysis

1) Sensitivity Analysis: We conduct experiments in different live streaming scenarios with different transmission control algorithms to show the superior generalization ability of TCLiVi. The tested scenarios include the AsianCup competition (China vs. Uzbekistan), a talent show (Fengtimo_2018_11_3), and the Dota competition (YYF_2018_08_12). We tested four transmission control algorithms, including TCLiVi, Double DQN, MPC, and the BufferBased algorithms. For Double DQN [17], we used the same structure of neural network and state, action, and reward definitions in TCLiVi. For the buffer-based algorithm, we used a 0.6 s and 2 s as the values of RESERVOIR and CUSHION, which have been extensively used in different experimental scenarios and can deliver a relatively high performance in all scenarios. In our experiments, we used the video traces of AsianCup competition scenario to train the two DRL-based



(a) Tunning Learning Rate for Four Step Critic Network (b) Sub graph of Fig 7(a)

Fig. 7. Learning Rate Test in the Four-Step Setting.



(a) N Step Size for both Actor and Critic Network (b) Sub graph of Fig 8(a)

Fig. 8. The Step Size Test for Critic and Actor Network.

algorithms, i.e. TCLiVi and Double DQN, and then applied the trained models in the other two video streaming scenarios to test their performance. Based on the parameter tunning results, we utilized the four-step return for the critic network.

The experiment results are shown in Fig. 9. We can see that TCLiVi outperforms the other three algorithms in all scenarios. The average QoE score of TCLiVi is 47.369, and the average QoE score of other algorithms is 33.633, so TCLiVi increase QoE score by 40.84% in average. In different video streaming scenarios, the performance of TCLiVi does not fluctuate severely whereas the rule-based algorithms, i.e. the MPC and Buffer-Based algorithms, have high variations of performance in different scenarios. More specifically, the sample variances of TCLiVi, Double DQN, MPC and Buffer-Based are 2.94, 3.84, 21.69, and 72.52 respectively. Significant change of scenarios does have influence in TCLiVi, however, with very little online training, the performance can still reach its peak.

2) Specific Observation Comparision: In the last section, we show the generalization ability and performance of TCLiVi with the overall QoE score. In this section, we will separately analyze the four components of QoE, i.e. bitrate, rebuff, end-to-end delay, and bitrate switch. For the experiments of this section, we apply the four algorithms in the AsianCup competition scenario and use the four-step return for the critic network.

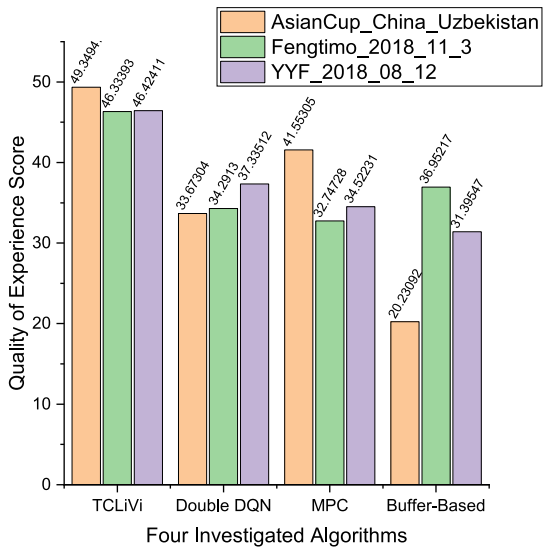


Fig. 9. Sensitivity Analysis for Different Algorithms.

The bitrate can represent video quality to some degree. We separately count the occurrences of the four bitrate levels for four algorithms. The experiment result is shown as Fig. 10. We only concern the boundary frames, since the inner frames use the same

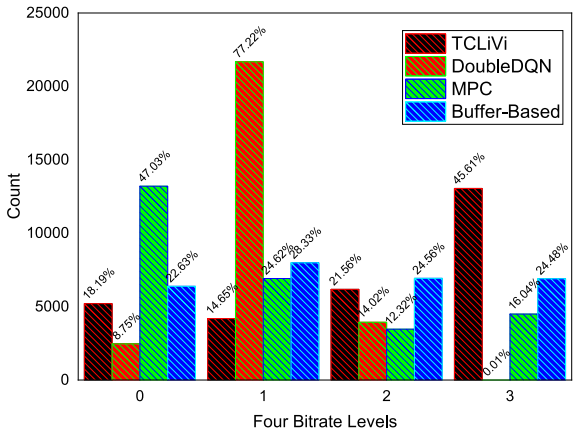


Fig. 10. Bitrate Level Statistic Data for Different Algorithms.

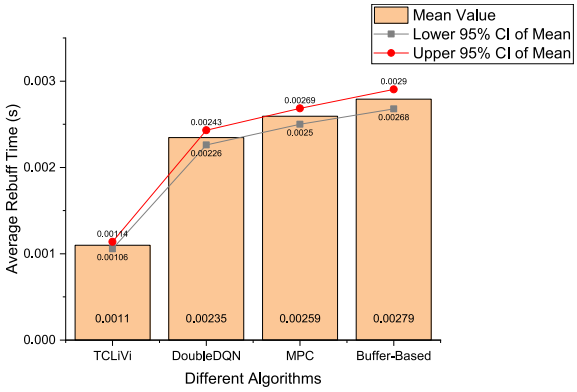


Fig. 11. Average Rebuff Time Comparison for Different Algorithms.

bitrate as the boundary frame. The four bitrate levels, i.e. {0, 1, 2, 3}, represent {500 kbps, 850 kbps, 1200 kbps 1850 kbps} respectively. From Fig. 10, we can see that TCLiVi has 45.61% of the highest bitrate level (1850 kbps), while the other three algorithms only have 0.01%, 16.04%, and 24.48%, respectively. The percentage of the second highest level of bitrate (1200 kbps) is also higher than two other algorithms (MPC and DoubleDQN) and almost the same as the BufferBased algorithm.

For the rebuff time, TCLiVi also outperforms the other three algorithms, which are shown in Fig. 11. The average rebuff time (0.0011 s) of TCLiVi is less than half of the rebuff time of the other three algorithms (0.00235 s, 0.00259 s, and 0.00279 s respectively). The rebuff time is the most important factor influencing QoE for most viewers. So far, we can see that TCLiVi can increase video quality while reducing rebuffering time. The promising optimization result of these two sub-objectives is the reason why TCLiVi can achieve higher overall QoE than the other algorithms.

The comparison of end-to-end delay for the four algorithms is shown as Fig. 12. We can see that the four algorithms have almost the same level of average end-to-end delay, which are 3.27342 s, 3.07364 s, 3.21245 s, and 3.12534 s, respectively. Fig. 13 shows the average bitrate difference at the boundaries of the video chunks. For this specific sub-objective, TCLiVi performs better than the two rule-based algorithms and worse than

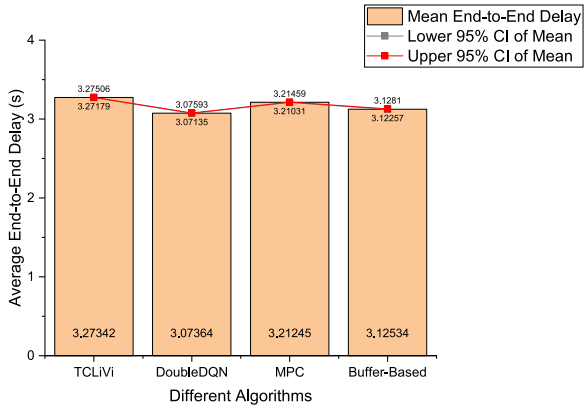


Fig. 12. Average End-to-End Delay for Four Algorithms.

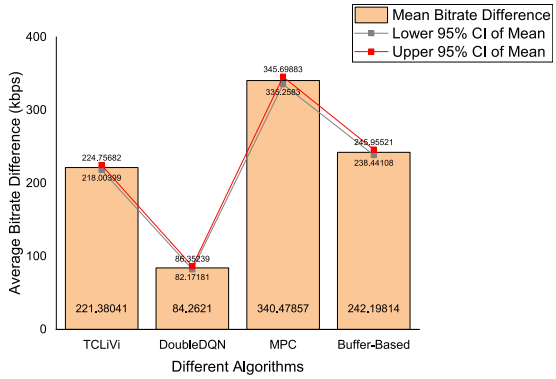


Fig. 13. Average Bitrate Difference for Four algorithms.

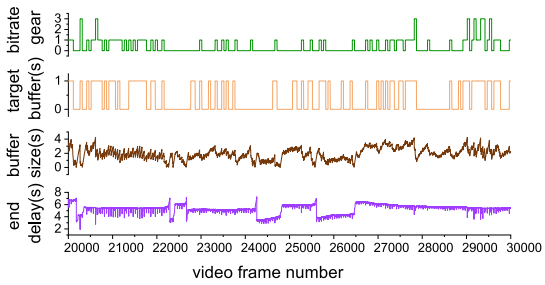


Fig. 14. Bitrate, Target Buffer, Buffer Size and End Delay vs. video frame number.

the Double DQN algorithm. We can conclude that in these settings of QoE preference, TCLiVi mainly optimizes the video quality and rebuff while balancing end-to-end delay and stability, which produce a promising overall QoE result.

From the above analysis, we conclude that TCLiVi can optimize the sub-objectives, such as video quality and rebuffering time, that users concern most, and balance other metrics according to the preferences of different users, which can be specified by the weights of the QoE formula.

To visually show the decisions made by our approach and the resulting metrics, we extract the traces of a sample playback information to draw Fig. 14. It shows the detailed decisions of video bitrate and target buffer as well as the resulting buffer size and end delay in a sample of 10000 video frames. The figure

shows that the buffer size and end-to-end delay can be controlled at a low level without causing zero buffer size.

VI. Conclusion

In the article, we first introduced the challenge of providing a favorable transmission control policy in live video streaming scenarios, and reviewed some recent advanced algorithms for solving this problem, including the rule-based and the learning-based algorithms for VoD and real-time streaming scenarios. We then presented the advantages of the DRL-based algorithm, introduced the process of live video streaming and the system model, and formulated the problem and the optimization objective. Further, we introduced some basic DRL theory and presented our basic solution and some carefully-designed enhancements in detail. Finally, we described our simulation methodology and showed that the proposed DRL-based algorithm can generate better results in different live video streaming scenarios compared to the other baseline algorithms.

References

[1] C. V. N. Index, "Forecast and methodology, 2015–2020," White paper, pp. 1–41, 2016.

[2] "2018 global internet phenomena report," 2018. [Online]. Available: https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf

[3] C. Ge, N. Wang, G. Foster, and M. Wilson, "Toward QoE-assured 4k video-on-demand delivery through mobile edge virtualization with adaptive prefetching," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2222–2237, Oct. 2017.

[4] Z. Lu, S. Ramakrishnan, and X. Zhu, "Exploiting video quality information with lightweight network coordination for HTTP-based adaptive video streaming," *IEEE Trans. Multimedia*, vol. 20, no. 7, pp. 1848–1863, Jul. 2018.

[5] C.-C. Chen and Y.-C. Lin, "What drives live-stream usage intention? The perspectives of flow, entertainment, social interaction, and endorsement," *Telematics Informat.*, vol. 35, no. 1, pp. 293–303, 2018.

[6] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 362–373, 2011.

[7] C. E. Palau, J. Mares, B. Molina, and M. Esteve, "Wireless CDN video streaming architecture for IPTV," *Multimedia Tools Appl.*, vol. 53, no. 3, pp. 591–613, 2011.

[8] K. Khan and W. Goodridge, "S-MDP: Streaming with markov decision processes," *IEEE Trans. Multimedia*, vol. 21, no. 8, pp. 2012–2025, Aug. 2019.

[9] C. Zhou, C. Lin, and Z. Guo, "MDASH: A markov decision-based rate adaptation approach for dynamic http streaming," *IEEE Trans. Multimedia*, vol. 18, no. 4, pp. 738–751, Apr. 2016.

[10] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," in *Proc. NIPS Deep Learning Workshop*, 2013, pp. 1–9.

[11] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[12] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao, "Deep reinforcement learning for dialogue generation," in *Proc. 2016 Conf. Empirical Methods in Natural Language Processing*, Nov. 2016, pp. 1192–1202.

[13] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3357–3364.

[14] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 197–210.

[15] V. Konda, "Actor-critic algorithms," Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, 2002, aAI0804543.

[16] K. De Asis, J. F. Hernandez-Garcia, G. Z. Holland, and R. S. Sutton, "Multi-step reinforcement learning: A unifying algorithm," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2902–2909.

[17] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[18] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 187–198, Aug. 2014.

[19] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[20] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, Feb. 2014.

[21] G. Tian and Y. Liu, "Towards agile and smooth video adaptation in http adaptive streaming," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2386–2399, Aug. 2016.

[22] Y. Sun *et al.*, "CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 272–285.

[23] S. Kim and C. Kim, "XMAS: An efficient mobile adaptive streaming scheme based on traffic shaping," *IEEE Trans. Multimedia*, vol. 21, no. 2, pp. 442–456, Feb. 2019.

[24] M. Aguayo, L. Bellido, C. M. Lentisco, and E. Pastor, "Dash adaptation algorithm based on adaptive forgetting factor estimation," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1224–1232, May 2018.

[25] Z. Li *et al.*, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719–733, 2014.

[26] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 325–338, Aug. 2015.

[27] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: Picking a video streaming rate is hard," in *Proc. Internet Meas. Conf.*, 2012, pp. 225–238.

[28] W. Choi and J. Yoon, "SATE: Providing stable and agile adaptation in http-based video streaming," *IEEE Access*, vol. 7, pp. 26 830–26 841, 2019.

[29] S. Han, Y. Go, H. Noh, and H. Song, "Cooperative server-client http adaptive streaming system for live video streaming," in *Proc. Int. Conf. Inf. Netw.*, Jan. 2019, pp. 176–180.

[30] K. Nihei, H. Yoshida, N. Kai, K. Satoda, and K. Chono, "Adaptive bitrate control of scalable video for live video streaming on best-effort network," in *Proc. IEEE Global Commun. Conf.*, 2018, pp. 1–7.

[31] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[34] X. Ma *et al.*, "Steward: Smart edge based joint QoE optimization for adaptive video streaming," in *Proc. 29th ACM Workshop Netw. Operating Syst. Support Digit. Audio Video*, 2019, pp. 31–36.

[35] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-DASH: A deep Q-learning framework for DASH video streaming," *IEEE Trans. Cognitive Commun. Netw.*, vol. 3, no. 4, pp. 703–718, Dec. 2017.

[36] J. Fu, X. Chen, Z. Zhang, S. Wu, and Z. Chen, "360SRL: A sequential reinforcement learning approach for ABR tile-based 360 video streaming," in *Proc. IEEE Int. Conf. Multimedia Expo.*, Jul. 2019, pp. 290–295.

[37] Y. Zhang *et al.*, "DRL360: 360-degree video streaming with deep reinforcement learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 1252–1260.

[38] R.-X. Zhang *et al.*, "Enhancing the crowdsourced live streaming: A deep reinforcement learning approach," in *Proc. 29th ACM Workshop Netw. Operating Syst. Support Digit. Audio Video*, 2019, pp. 55–60.

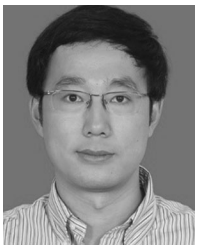
[39] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. 2nd Annu. ACM Conf. Multimedia Syst*, 2011, pp. 157–168.

[40] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. thesis, Cambridge Univ., Cambridge, England, 1989.

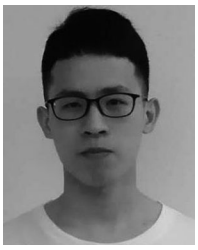
[41] M. L. Puterman, "Markov decision processes," *Handbooks Operations Res. Manage. Sci.*, vol. 2, pp. 331–434, 1990.

[42] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, pp. 116–123.

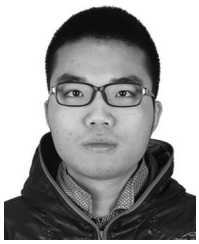
[43] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.
[44] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Netw.*, vol. 3, no. 5, pp. 551–560, 1990.
[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[46] L. Xu, C. Choy, and Y. Li, "Deep sparse rectifier neural networks for speech denoising," in *Proc. IEEE Int. Workshop Acoust. Signal Enhancement*, Sep. 2016, pp. 1–5.
[47] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," 2012. [online]. Available: https://cs.toronto.edu/csc321/slides/lecture_slides_lec6.pdf
[48] A. Balachandran *et al.*, "Developing a predictive model of quality of experience for internet video," in *Proc. ACM SIGCOMM Conf.*, 2013, Art. no. 339350.
[49] "Global ai transmission competition," 2019. [Online]. Available: https://www.aitrans.online/competition_detail/competition_id=2



Laizhong Cui (Senior Member, IEEE) received the B.S. degree from Jilin University, Changchun, China, in 2007 and the Ph.D. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012. He is currently a Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. He led more than 10 scientific research projects, including National Key Research and Development Plan of China, National Natural Science Foundation of China, Guangdong Natural Science Foundation of China and Shenzhen Basic Research Plan. His research interests include future internet architecture and protocols, edge computing, multimedia systems and applications, blockchain, internet of things, cloud and big data computing, software-defined network, social network, computational intelligence and machine learning. He has authored more than 70 papers, including the IEEE TRANSACTIONS ON MULTIMEDIA, IEEE IoT JOURNAL, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, *ACM Transactions on Internet Technology*, IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, and IEEE NETWORK. He serves as an Associate Editor or a member of Editorial Board for several international journals, including *International Journal of Machine Learning and Cybernetics*, *International Journal of Bio-Inspired Computation*, *Ad Hoc and Sensor Wireless Networks,* and *Journal of Central South University*. He is a Senior Member of the CCF.



Dongyuan Su received the B.Sc. degree from Shenzhen University, Shenzhen, China, where he is currently working toward the M.Sc. degree. His research interests include edge computing and machine learning.



Shu Yang received the B.Sc. degree from the Beijing University of Posts and Telecommunications, Beijing, China and the Ph.D. degree from Tsinghua University, Beijing, China. He is currently an Associate Researcher with Shenzhen University, Shenzhen, China. His research interest includes network architecture and high performance router.

Zhi Wang is currently an Associate Professor with Tsinghua Shenzhen International Graduate School, Shenzhen, China. His research areas include multimedia networks, mobile cloud computing, and large-scale machine learning systems. He was the recipient of the Outstanding Doctoral Dissertation Award from China Computer Federation in 2014, Best Paper Award at ACM Multimedia 2012, and Best Student Paper Award at MMM 2015. He is also the recipient of the Second Prize of National Natural Science Award and the First Prize of Natural Science Award of Ministry of Education in 2017. He is an Associate Editor for the IEEE TMM and Guest Editor of ACM TIST and JCST. His research has been covered by prestigious media including MIT Technology Review.

Zhong Ming received the Ph.D. degree in computer science and technology from Sun Yat-Sen University, Guangzhou, China, in 2003. He is currently a Professor with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include software engineering and web intelligence.