Enhancing Neural Adaptive Wireless Video Streaming via Lower-Layer Information Exposure

Lingzhi Zhao*, Ying Cui†, *Member, IEEE*, Yuhang Jia‡, Yunfei Zhang‡, and Klara Nahrstedt*, *Fellow, IEEE, ACM*
*Department of Computer Science, University of Illinois Urbana-Champaign, USA
†Internet of Things Thrust, The Hong Kong University of Science and Technology (Guangzhou), CN
‡Tencent Technology Co. Ltd, CN

ICC 2024 - IEEE International Conference on Communications | 978-1-7281-9054-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICC51166.2024.10622554

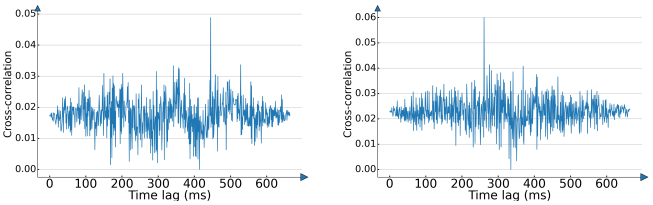
***Abstract*—Deep reinforcement learning (DRL) demonstrates its promising potential in the realm of adaptive video streaming. However, existing DRL-based methods for adaptive video streaming use only application (APP) layer information and adopt heuristic training methods. This paper aims to boost the quality of experience (QoE) of adaptive wireless video streaming by using lower-layer information and deriving a rigorous training method. First, we formulate a more comprehensive and accurate adaptive wireless video streaming problem as an infinite stage discounted Markov decision process (MDP) problem by additionally incorporating past and lower-layer information, allowing a flexible tradeoff between QoE and computational and memory costs for solving the problem. Then, we propose an enhanced asynchronous advantage actor-critic (eA3C) method by jointly optimizing the parameters of parameterized policy and value function. Specifically, we build an eA3C network consisting of a policy network and a value network that can utilize cross-layer, past, and current information and jointly train the eA3C network using pre-collected samples. Finally, experimental results show that the proposed eA3C method can improve the QoE by $6.8\% \sim 14.4\%$ compared to the state-of-the-arts.**

I. Introduction

Video on demand (VoD) services, responsible for 29% of Internet traffic, allow users, on demand, to select and view videos, utilizing continuous video streaming technology. During video streaming, network fluctuations can easily cause rebuffering, severely degrading viewers' viewing experiences. To address this issue, adaptive video streaming [1]–[4] adapts the video chunk bitrate to the dynamic network condition and user's buffer occupancy. This involves a Markov decision process (MDP) problem where a (bitrate adaptation) policy that maps the system state to the video chunk bitrate is optimized to maximize the users' quality of experience (QoE).

Deep reinforcement learning (DRL) has been recently applied to solve the MDP problems for adaptive video streaming [1], [2]. There are two main DRL algorithms, i.e., deep Q-learning (DQ) [1] and asynchronous advantage actor-critic (A3C) [2]. A3C naturally balances exploration and exploitation through its randomized policy design and is shown in [2] to achieve better performance than DQ in adaptive video streaming.

This work was funded in part by Guangzhou Basic and Applied Basic Research Scheme-Pilot Voyage Project 2024A04J6459 and the National Science Foundation CCF 22-17144. *(Corresponding author: Ying Cui, e-mail: yingcui@ust.hk)*



(a) $R_{X,C}(\tau)$ on Set-1 (b) $R_{X,C}(\tau)$ on Set-2

Fig. 1: Relationship between APP layer throughput sequence $\{C_n : n = 1, \ldots, \}$ and downlink media access control (MAC) rate sequence $\{X_n : n = 1, \ldots, \}$ on two collected datasets [12] (see Section V for details). The (sample) cross-correlation between X and C is defined as $R_{X,C}(\tau) \triangleq \frac{1}{N-\tau} \sum_{n=1}^{N-\tau} X_n C_{n+\tau}$, where $\tau \in \{0, 1, \ldots, N\}$ is time lag, and N is the number of samples.

Notably, adaptive video streaming becomes more challenging in the wireless scenario due to fast varying wireless channels and potential user mobilities. Since the application (APP) layer relies on all lower layers to complete its process and lower-layers can respond more rapidly to the changes in the environment (as shown in Fig. 1), information from the lower layers may also be helpful for wireless adaptive video streaming [5]. The existing work [3], [4] utilizes lower layer information to approximate the APP layer throughput without utilizing the APP layer information and adopts optimization [3] or heuristic [4] methods to design the bitrate adaptation policies.

***Limitations**:* there are two major limitations in the existing work on wireless adaptive video streaming [1]–[4]. Firstly, most of the existing work [1]–[4] *fails to jointly utilize the APP layer and lower-layer information* and thus cannot fully capture the dynamic nature of wireless video streaming. Secondly, the existing A3C training methods [2], [6] are *rather heuristic* and *do not have any theoretical convergence or performance guarantee*. Specifically, they break the original optimization problem for the policy and value parameters into two separate but related problems and trains the policy and value networks alternatively by solving the optimization problems for the value and policy parameters alternatively.

To address the above limitations, this paper boosts the

TABLE I: KEY NOTATION.

Notation	Description
B	buffer size (in seconds)
T	length of each chunk (in seconds)
N ($\mathcal{N}$)	number (set) of chunks
D ($\mathcal{D}$)	number (set) of quality levels
M ($\mathcal{M}$)	number (set) of lower-layer quantities
R_n ($\mathcal{R}$)	bitrate (action) of n-th chunk (bitrate set)
$\mathbf{C}_n, B_n$	APP layer throughput, buffer occupancy at stage n
$\overline{\mathbf{X}}_n$	lower-layer quantities at stage n
Y_n	bitrate of $(n-1)$-th chunk
$\mathbf{S}_n \triangleq (B_n, \mathbf{C}_n, \overline{\mathbf{X}}_n, Y_n)$	system state at stage n
$\pi(\cdot;\mathbf{\Theta}_\pi)$	parameterized offline policy
$V^\pi(\cdot;\mathbf{\Theta}_v)$	parameterized value function under offline policy $\pi(\cdot;\mathbf{\Theta}_\pi)$
$\mathbf{\Theta}_\pi$ ($\mathbf{\Theta}_v$)	eA3C policy (value) network parameters

performance (users' QoE) of adaptive wireless video streaming by incorporating lower-layer information in the problem formulation and obtaining an enhanced policy via improving the training method for A3C. First, we model the *impacts of lower-layer information* together with the APP layer information on adaptive wireless video streaming. Then, by additionally incorporating past and lower-layer information, we formulate a more comprehensive and accurate adaptive wireless video streaming problem as an infinite stage discounted MDP problem, offering possibilities for enhancing QoE. The proposed formulation allows a flexible tradeoff between QoE and computational and memory costs for solving the problem. Next, we propose an *enhanced A3C (eA3C) method* which jointly trains the policy and value neural networks (i.e., jointly optimizes the policy and value parameters) using pre-collected samples. eA3C can better capture intrinsic relationship between policy and value parameters and thus achieve faster convergence speed and better convergent performance than the state-of-art A3C method [6]. Finally, the experimental results based on real-world collected data [12] show that the proposed policy can improve the QoE by $6.8\% \sim 14.4\%$ compared to the state-of-the-arts, which reveals the importance of utilizing lower-layer information and the advantage of the eA3C method. The key notation used in this paper is listed in Table I.

II. System Model

A. System Model

As illustrated in Fig. 2, we consider adaptive wireless streaming of one video from a video server to a user. On the server side, the video is segmented into N chunks. Let $\mathcal{N} \triangleq \{1, \ldots, N\}$ denote the set of video chunks. The length of each chunk is T (in seconds). The typical value of T is $1 \sim 4$. To well adapt to the network conditions, each chunk is pre-encoded into D representations corresponding to D quality levels using High Efficiency Video Coding (HEVC), as in Dynamic Adaptive Streaming over HTTP (DASH). Let $\mathcal{D} \triangleq \{1, \ldots, D\}$ denote the set of D quality levels. For ease of exposition, assume that the bitrates of the chunks with the same quality level are identical [3]. For all $d \in \mathcal{D}$, the bitrate of the d-th representation of a chunk is denoted by r_d (in bits/s). Note that $r_d, d \in \mathcal{D}$ satisfy $r_1 < \cdots < r_D$. Let $\mathcal{R} \triangleq \{r_1, \ldots, r_D\}$ denote the set of bitrates corresponding to the D quality levels. The N chunks will be sequentially

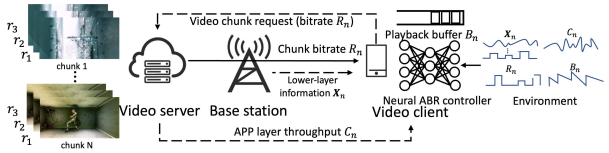


Fig. 2: System model. $D = 3, \mathcal{D} = \{1, 2, 3\}, \mathcal{R} = \{r_1, r_2, r_3\}$.

transmitted to the user (downloaded by the user). For each chunk, only one of the D representations will be transmitted to the user. Let R_n (in bits/s) denote the (APP layer) bitrate of the n-th chunk that is transmitted to the user, where

$$R_n \in \mathcal{R},\ n \in \mathcal{N}. \tag{1}$$

Let $C_n \in \mathcal{C}$ represent the average APP layer throughput experienced during the download process for chunk $n \in \mathcal{N}$, where $\mathcal{C}$ denotes the APP layer throughput space. We model $C_n, n \in \mathcal{N}$ as random variables, since the APP layer throughput usually varies over time. Then, the download time (in seconds) for chunk n is given by $\frac{R_n T}{C_n}$. Since the APP layer relies on all lower-layers to complete its process, and lower-layers are able to respond more rapidly to the changes in the environment as shown in Fig. 1, information from the lower-layers, such as downlink media access control (MAC) rate, number of occupied physical resource blocks (PRBs), and modulation and coding scheme (MCS) index, may also be helpful for adaptive wireless video streaming. Therefore, it is important to explicitly model lower-layer information [5]. Specifically, let M and $\mathcal{M} \triangleq \{1, \ldots, M\}$ denote the number and set of lower-layer quantities, respectively. For all $n \in \mathcal{N}$ and $m \in \mathcal{M}$, let $X_{n,m}$ denote the value of the m-th lower-layer quantity (e.g., MAC rate, PRB number or MCS index) when downloading chunk n, referred to as the m-th lower layer information. Similarly, we model $X_{n,m}, n \in \mathcal{N}, m \in \mathcal{M}$ as random variables to reflect the possible changes over time. Denote $\mathbf{X}_n \triangleq (X_{n,m})_{m \in \mathcal{M}} \in \boldsymbol{\mathcal{X}}$ as the overall lower-layer information when downloading chunk n where $\boldsymbol{\mathcal{X}} \triangleq \mathcal{X}^M$ denotes the overall lower-layer information space, and $\mathcal{X}$ denotes the largest state space of the M lower-layer quantities. We assume that $\mathcal{X}$ is a finite set for simplicity. Mathematically, $C_n, n \in \mathcal{N}$ and $\mathbf{X}_n, n \in \mathcal{N}$ are characterized by probability distributions $\Pr[C_n | C_{n-1}, C_{n-2}, \ldots, C_1, \mathbf{X}_n, \mathbf{X}_{n-1}, \ldots, \mathbf{X}_1]$ and $\Pr[\mathbf{X}_n | \mathbf{X}_{n-1}, \ldots, \mathbf{X}_1]$, respectively.

The N video chunks are downloaded into a playback buffer at the user side, which contains downloaded but as yet unviewed video chunks. Let $B > 0$ denote the buffer size (in seconds) which depends on the policy of the service provider and storage limitation on the user. A typical buffer may hold a few tens of seconds of video chunks. Let $B_n \in [0, B]$ denote the (APP layer) buffer occupancy (in seconds), i.e., the play time of the video chunks left in the buffer, when the user starts to download chunk $n \in \mathcal{N}$. Thus, the buffer occupancy evolves according to

$$B_{n+1} = \max\left(\left(B_n - \frac{R_n T}{C_n}\right)^+ + T, B\right),\ n \in \mathcal{N}, \tag{2}$$

where $(x)^+ \triangleq \max\{x, 0\}$. Let $\mathcal{B}$ denote the buffer occupancy space. Note that $B_n, n \in \mathcal{N}$ are also random due to the randomness of $C_n, n \in \mathcal{N}$. The buffer update equation in (2) indicates that the buffer occupancy increases by T seconds after each chunk is downloaded and decreases as the user watches the video. While downloading chunk n, the buffer occupancy keeps on decreasing if $B_n \geq \frac{R_n T}{C_n}$, as the user is continuously watching the video. The buffer is empty, leading to rebuffering of time $\left(\frac{R_n T}{C_n} - B_n\right)^+$, if $B_n < \frac{R_n T}{C_n}$.

B. Performance Metrics

We refer to the stage where chunk n is being downloaded as stage n, for all $n \in \mathcal{N}$. The system evolves over N stages. We consider three metrics for QoE of video streaming, i.e., video quality, video quality variation, and rebuffering time. Specifically, let $U(R)$ denote the utility for a chunk with the bitrate R. Here, $U(\cdot)$ can be any nonnegative, nondecreasing, concave function, and $U(0) = 0$. Its monotonicity can capture the notion that perceptual quality increases with bitrate. Its concavity can capture the notion that the increased rate of perceptual quality decreases with bitrate. We define the user's QoE at stage n as:

$$g(C_n, B_n, R_n, R_{n-1}) \triangleq U(R_n) - \alpha|U(R_n) - U(R_{n-1})| - \beta\left(\frac{R_n T}{C_n} - B_n\right)^+, \quad (3)$$

where $\alpha > 0$ and $\beta > 0$ are the associated weights for the video quality variation (i.e., quality smoothness) $|U(R_n) - U(R_{n-1})|$ [2], [10] and rebuffering time (i.e., stall time) $\left(\frac{R_n T}{C_n} - B_n\right)^+$ [2], [10], respectively.

III. QoE Maximization Problem Formulation

Note that a video usually consists of a large number of chunks (e.g., a 40-minute video contains 600-2400 chunks). Thus, in the rest of the paper, we consider the extreme scenario where $N \to \infty$, and $\mathcal{N}$ turns to the set of natural numbers, denoted by $\mathbb{N}$, for tractability. In this section, we formulate the adaptive wireless streaming problem as an infinite stage discounted MDP under certain conditions. Specifically, to ensure a stationary MDP, we first adopt the following assumption.

Assumption 1 (Stationary Markov Chains of Order k): For all $n \in \mathbb{N}$ and for some $k \in \mathbb{N}^+$, the probability distributions $\Pr[C_n|C_{n-1}, C_{n-2}, \ldots, C_1, \mathbf{X}_n, \mathbf{X}_{n-1}, \ldots, \mathbf{X}_1]$ and $\Pr[\mathbf{X}_n|\mathbf{X}_{n-1}, \ldots, \mathbf{X}_1]$ satisfy:

$$\Pr[C_n|C_{n-1}, C_{n-2}, \ldots, C_1, \mathbf{X}_n, \mathbf{X}_{n-1}, \ldots, \mathbf{X}_1] = \Pr[C_n|C_{n-1}, C_{n-2}, \ldots, C_{n-k}, \mathbf{X}_n, \mathbf{X}_{n-1}, \ldots, \mathbf{X}_{n-k}],$$
$$\Pr[\mathbf{X}_n|\mathbf{X}_{n-1}, \ldots, \mathbf{X}_1] = \Pr[\mathbf{X}_n|\mathbf{X}_{n-1}, \mathbf{X}_{n-2}, \ldots, \mathbf{X}_{n-k}],$$

and are independent of stage n.

Under Assumption 1, $\{(\mathbf{X}_n, \mathbf{X}_{n-1}, \ldots, \mathbf{X}_{n-k+1}) : n \in \mathbb{N}\}$ and $\{(C_n, C_{n-1}, \ldots, C_{n-k+1}, \mathbf{X}_n, \mathbf{X}_{n-1}, \ldots, \mathbf{X}_{n-k+1}) : n \in \mathbb{N}\}$ are stationary Markov chains of order k. Furthermore, we define the system state at stage n, denoted by $\mathbf{S}_n$, as follows. Given Assumption 1, we include $\mathbf{C}_n \triangleq (C_n, C_{n-1}, \ldots, C_{n-k+1})$ and $\overline{\mathbf{X}}_n \triangleq (\mathbf{X}_n, \mathbf{X}_{n-1}, \ldots, \mathbf{X}_{n-k+1})$ in $\mathbf{S}_n$ by using state augmentation [7, pp. 38]. Besides, noting that for all $n \in \mathbb{N}$, $g(C_n, B_n, R_n, R_{n-1})$ in (3) depends not only on (C_n, B_n, R_n) but also on R_{n-1}, we include R_{n-1} in $\mathbf{S}_n$ by using action augmentation [7, pp. 38]. Specifically, we introduce auxiliary variable Y_n at stage n, and let

$$Y_n = R_{n-1}, \ n \in \mathbb{N}. \quad (4)$$

Eventually, we have $\mathbf{S}_n \triangleq (B_n, \mathbf{C}_n, \overline{\mathbf{X}}_n, Y_n) \in \boldsymbol{\mathcal{S}}$, where $\boldsymbol{\mathcal{S}} = \mathcal{B} \times \mathcal{C}^k \times \boldsymbol{\mathcal{X}}^k \times \mathcal{R}$ denotes the system state space. Define $\mathbf{s}_n \triangleq (b_n, \mathbf{c}_n, \overline{\mathbf{x}}_n, y_n), n \in \mathbb{N}$. Therefore, by Assumption 1 and the definition of the system state, we can show the following result on the transition probabilities of the system.

Lemma 1 (Transition Probabilities): For all $n \in \mathbb{N}$,

$$\begin{aligned}&\Pr[\mathbf{S}_{n+1} = \mathbf{s}_{n+1}|\mathbf{S}_n = \mathbf{s}_n, \ldots, \mathbf{S}_1 = \mathbf{s}_1, R_n = r_n, \ldots R_1 = r_1]\\ &= \Pr[\mathbf{C}_{n+1} = \mathbf{c}_{n+1}|\overline{\mathbf{X}}_{n+1} = \overline{\mathbf{x}}_{n+1}, \mathbf{C}_n = \mathbf{c}_n, \overline{\mathbf{X}}_n = \overline{\mathbf{x}}_n]\\ &\times \Pr[\overline{\mathbf{X}}_{n+1} = \overline{\mathbf{x}}_{n+1}|\overline{\mathbf{X}}_n = \overline{\mathbf{x}}_n]\\ &\times \mathbb{I}\left[b_{n+1} = \max\left(\left(b_n - \frac{r_n T}{c_n}\right)^+ + T, B\right)\right] \times \mathbb{I}\left[y_{n+1} = r_n\right],\end{aligned}$$

which is independent of n. Here, $\mathbb{I}[\cdot]$ denotes the indicator function.

Proof 1: Based on Assumption 1, the equations in (2) and (4), the conditionally independence of B_{n+1} and Y_{n+1} given C_n, B_n, R_n, and the conditionally independence of Y_{n+1} and B_{n+1} given R_n, we can show Lemma 1. We omit the details due to page limitations. Please refer to [12] for the details. ■

Define $\mathbf{s}' = (b', \mathbf{c}', \overline{\mathbf{x}}', y')$ and $\mathbf{s} = (b, \mathbf{c}, \overline{\mathbf{x}}, y)$. Based on Lemma 1, for all $n \in \mathbb{N}$, $\mathbf{s}, \mathbf{s}' \in \boldsymbol{\mathcal{S}}$ and $r \in \mathcal{R}$, we denote $\Pr[\mathbf{S}_{n+1} = \mathbf{s}'|\mathbf{S}_n = \mathbf{s}, R_n = r]$ by $p_{\mathbf{s},\mathbf{s}'}(r)$, referred to as the transition probability from the system state $\mathbf{s}$ to a successor system state $\mathbf{s}'$ for a given bitrate r.

We refer to the bitrate at stage n, R_n, as the action at stage n and term $\mathcal{R}$ as the action space. Consider a stationary randomized (bitrate adaptation) policy, denoted by π, which is a function that maps the system state $\mathbf{s}$ into a probability distribution $\pi(\mathbf{s}, r), r \in \mathcal{R}$. Note that for all $\mathbf{s} \in \boldsymbol{\mathcal{S}}$, $\pi(\mathbf{s}, \cdot)$ satisfies $\pi(\mathbf{s}, r) \geq 0, r \in \mathcal{R}$ and $\sum_{r\in\mathcal{R}} \pi(\mathbf{s}, r) = 1$. Under Assumption 1 and the stationary randomized policy π, we have a homogeneous Markov chain $\{\mathbf{S}_n : n \in \mathbb{N}\}$ with transition probabilities $\sum_{r\in\mathcal{R}} \pi(\mathbf{s}, r) p_{\mathbf{s},\mathbf{s}'}(r), \mathbf{s}', \mathbf{s} \in \boldsymbol{\mathcal{S}}$. Given the notion of the system state $\mathbf{S}_n$, we rewrite $g(C_n, B_n, R_n, R_{n-1})$ as $g(\mathbf{S}_n, R_n) = U(R_n) - \alpha|U(R_n) - U(Y_n)| - \beta\left(\frac{R_n T}{C_n} - B_n\right)^+$, referred to as the reward at stage n. Thus, given an initial state $\mathbf{s}$, the value function i.e., expected discounted QoE, under the stationary randomized policy π is given by

$$V^\pi(\mathbf{s}) = \limsup_{N\to\infty} \mathbb{E}\left[\sum_{n=1}^{N} \gamma^{n-1} \sum_{r\in\mathcal{R}} \pi(\mathbf{S}_n, r) g(\mathbf{S}_n, r)\right], \quad (5)$$

where $\mathbf{S}_1 = \mathbf{s}$, the system states $\mathbf{S}_n, n = 1, 2, 3, \ldots, N$ evolve according to $\sum_{r\in\mathcal{R}} \pi(\mathbf{s}, r) p_{\mathbf{s},\mathbf{s}'}(r), \mathbf{s}', \mathbf{s} \in \boldsymbol{\mathcal{S}}$, and the expectation is taken over $\mathbf{S}_n, n = 2, 3, \ldots, N$.

We aim to optimize the stationary randomized policy π to maximize the expected discounted QoE given in (5). The problem is readily formulated as follows.

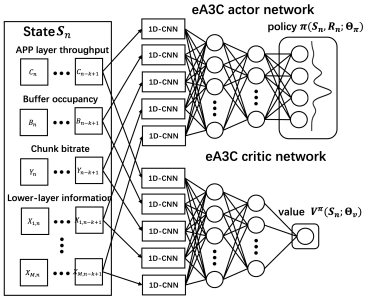


Fig. 3: Structure of the eA3C network

Problem 1 (QoE Maximization):

$$V(\mathbf{s}) = \max_{\pi} \ \limsup_{N\to\infty} \mathbb{E}\left[\sum_{n=1}^{N} \gamma^{n-1} \sum_{r\in\mathcal{R}} \pi(\mathbf{S}_n, r) g(\mathbf{S}_n, r)\right],$$

where $\mathbf{S}_1 = \mathbf{s}$, and $V(\mathbf{s})$ represents the optimal value function.

Based on the above illustration, we can conclude that Problem 1 is a stochastic discounted MDP over an infinite number of stages. Standard DP and Q-learning can be used to obtain an optimal stationary policy by solving the Bellman equation [8, Proposition 1.2.3]. However, they have prohibitively high computational complexities due to the extremely large state spaces and may not provide satisfactory QoE when Assumption 1 is not satisfied. DRL, leveraging deep learning and approximate DP, is put forward as a solution to address the issue. A3C [6], a widely used DRL algorithm, demonstrates promising potential in the realm of adaptive video streaming. The optimal value of Problem 1 and the computational and memory costs for solving Problem 1 generally grow with k and M due to the increases of the state information utilization and state space, respectively. Therefore, by properly selecting k and M, we can achieve flexible a tradeoff between QoE and computational and memory costs.

IV. Enhanced A3C Method

A. Joint Optimization Formulation

We approximate an arbitrary stationary randomized policy π by a parametric form $\pi(\cdot; \mathbf{\Theta}_\pi)$, with policy parameters $\mathbf{\Theta}_\pi \in \mathbb{R}^{n_\pi}$, and optimize $\mathbf{\Theta}_\pi$ instead of π for tractability as in [2], [6]. Specifically, we optimize $\mathbf{\Theta}_\pi$ to maximize the expected value function (the expected discounted QoE):

$$\max_{\mathbf{\Theta}_\pi} \quad \sum_{\mathbf{s}\in\boldsymbol{\mathcal{S}}} \eta^\pi(\mathbf{s}) V^\pi(\mathbf{s}), \tag{6}$$

where $\eta^\pi(\cdot)$ is the stationary probability distribution of the system state under policy π, and $V^\pi(\cdot)$ is the solution of the Bellman equation under policy π [8, Proposition 1.2.3]:

$$V^\pi(\mathbf{s}) = \sum_{r\in\mathcal{R}} \pi(\mathbf{s}, r; \mathbf{\Theta}_\pi) \left(g(\mathbf{s}, r) + \gamma \sum_{\mathbf{s}'\in\boldsymbol{\mathcal{S}}} p_{\mathbf{s}',\mathbf{s}}(r) V^\pi(\mathbf{s}')\right), \quad \mathbf{s}\in\boldsymbol{\mathcal{S}}. \tag{7}$$

Note that $V(\mathbf{s}) = \max_\pi V^\pi(\mathbf{s})$ [8, Proposition 1.2.3]. Solving the problem in (6) is challenging, since $V^\pi(\cdot)$ is a solution to an extremely large number of equations given by (7) and cannot be obtained analytically. To address the challenge, we equivalently transform the problem in (6) into the problem in (8), as shown at the top of the next page, with $(V^\pi(\mathbf{s}))_{\mathbf{s}\in\boldsymbol{\mathcal{S}}}$ as additional optimization variables, where $\xi > 0$. The equivalence is shown by the following lemma.

Lemma 2 (Relationship between the Problems in (6) *and* (8)*):* Consider a sequence $\{\xi^k\}$ with $0 < \xi^k < \xi^{k+1}$ for all $k = 0, 1, 2, \ldots$, and $\xi^k \to \infty$. Then every limit point of the sequence $\{\mathbf{\Theta}_\pi^k\}$ is a globally optimal point of the problem in (6), where $\mathbf{\Theta}_\pi^k$ represents a globally optimal point of the problem in (8) with $\xi = \xi^k$.

Proof 2: First, we equivalently convert the problem in (6) to a constrained problem by including $V^\pi(\mathbf{s}), \mathbf{s} \in \boldsymbol{\mathcal{S}}$ and equations in (7) as optimization variables and constraints:

$$\max_{\mathbf{\Theta}_\pi, (V^\pi(\mathbf{s}))_{\mathbf{s}\in\boldsymbol{\mathcal{S}}}} \quad \sum_{\mathbf{s}\in\boldsymbol{\mathcal{S}}} \eta^\pi(\mathbf{s}) V^\pi(\mathbf{s}), \tag{9}$$
$$\text{s.t.} \quad (7).$$

Next, we transform the constrained problem in (9) into the unconstrained problem in (8) whose objective function is the weighted sum of the objective of the problem in (7) and the penalty for violating the constraints of the problem in (7) by the penalty method [11, pp. 388]. According to [11, Proposition 5.2.1], we can show Lemma 2. ■

Based on Lemma 2, we can solve the problem in (8) with a sufficiently large ξ for tractability. Note that the problem in (8) is still challenging since the number of optimization variables, $|\boldsymbol{\mathcal{S}}|+n_\pi$, is prohibitively large. As in [2], [6], we approximate the value function under policy π, $V^\pi(\cdot)$, by a parametric form $V^\pi(\cdot; \mathbf{\Theta}_v)$, with value parameters $\mathbf{\Theta}_v \in \mathbb{R}^{n_v}$. Then, we consider the joint optimization of the policy and value parameters given by the problem in (10), as shown at the top of the next page. It is a simplified version of the problem in (8), and the number of the optimization variables reduces from $|\boldsymbol{\mathcal{S}}|+n_\pi$ to $n_v + n_\pi$. The problem in (10) is generally a non-convex stochastic problem with unknown stationary probability distribution $\eta^\pi(\cdot)$ and transition probabilities $p_{\mathbf{s},\mathbf{s}'}(r), \mathbf{s}, \mathbf{s}' \in \boldsymbol{\mathcal{S}}, r \in \mathcal{R}$. We can obtain a stationary point of it using standard stochastic gradient methods [11].

B. Neural Network Architecture

We present an actor-critic neural network corresponding to the eA3C method, namely eA3C network. The eA3C network consists of two neural networks, namely actor network (policy network) and critic network (value network), as shown in Figure 3. The inputs of two neural networks are the states over the most recent k chunks, $\mathbf{S}_n$. For the eA3C actor network, the input layer consists of one-dimensional convolutional neural networks which are used to extract the temporal features of the system states; the hidden layer is a fully connected layer that utilizes the Rectified Linear Unit activation function; the output layer is a fully connected layer that utilizes the softmax activation function. The eA3C actor network has weights $\mathbf{\Theta}_\pi \in \mathbb{R}^{n_\pi}$ and outputs $\pi(\mathbf{s}, r; \mathbf{\Theta}_\pi)$. The eA3C critic network has the same neural network structure as the eA3C actor network except that its output layer is a linear neural network

$$\max_{\boldsymbol{\Theta}_\pi,(V^\pi(\mathbf{s}))_{\mathbf{s}\in\boldsymbol{\mathcal{S}}}} \quad \sum_{\mathbf{s}\in\boldsymbol{\mathcal{S}}} \eta^\pi(\mathbf{s}) \sum_{r\in\mathcal{R}} \pi(\mathbf{s},r;\boldsymbol{\Theta}_\pi) \left(g(\mathbf{s},r) + \gamma \sum_{\mathbf{s}'\in\boldsymbol{\mathcal{S}}} p_{\mathbf{s},\mathbf{s}'}(r) V^\pi(\mathbf{s}') \right) - \xi \sum_{\mathbf{s}\in\boldsymbol{\mathcal{S}}} \eta^\pi(\mathbf{s}) \left(\sum_{r\in\mathcal{R}} \pi(\mathbf{s},r;\boldsymbol{\Theta}_\pi) \left(g(\mathbf{s},r) + \gamma \sum_{\mathbf{s}'\in\boldsymbol{\mathcal{S}}} p_{\mathbf{s},\mathbf{s}'}(r) V^\pi(\mathbf{s}') \right) - V^\pi(\mathbf{s}) \right)^2. \tag{8}$$

$$\max_{\boldsymbol{\Theta}_\pi,\boldsymbol{\Theta}_v} \quad \sum_{\mathbf{s}\in\boldsymbol{\mathcal{S}}} \eta^\pi(\mathbf{s}) \sum_{r\in\mathcal{R}} \pi(\mathbf{s},r;\boldsymbol{\Theta}_\pi) \left(g(\mathbf{s},r) + \gamma \sum_{\mathbf{s}'\in\boldsymbol{\mathcal{S}}} p_{\mathbf{s},\mathbf{s}'}(r) V^\pi(\mathbf{s}';\boldsymbol{\Theta}_v) \right) - \xi \sum_{\mathbf{s}\in\boldsymbol{\mathcal{S}}} \eta^\pi(\mathbf{s}) \left(\sum_{r\in\mathcal{R}} \pi(\mathbf{s},r;\boldsymbol{\Theta}_\pi) \left(g(\mathbf{s},r) + \gamma \sum_{\mathbf{s}'\in\boldsymbol{\mathcal{S}}} p_{\mathbf{s},\mathbf{s}'}(r) V^\pi(\mathbf{s}';\boldsymbol{\Theta}_v) \right) - V^\pi(\mathbf{s};\boldsymbol{\Theta}_v) \right)^2. \tag{10}$$

without any activation function. The eA3C critic network has weights $\boldsymbol{\Theta}_v \in \mathbb{R}^{n_v}$ and outputs $V^\pi(\mathbf{s};\boldsymbol{\Theta}_v)$. Please refer to [12] for the detailed architecture.

Let $\{(\mathbf{C}_n, \mathbf{X}_n) : n = 1, 2, \ldots, N\}$ denote the pre-collected data from a large number of users. Given the sample $\mathbf{S}_n$ at stage n, we generate the action R_n according to the policy distribution $\pi(\mathbf{S}_n, r; \boldsymbol{\Theta}_{\pi,n}), r \in \mathcal{R}$, i.e., $\Pr[R_n = r] = \pi(\mathbf{S}_n, r; \boldsymbol{\Theta}_{\pi,n})$ for all $r \in \mathcal{R}$, and obtain sample $\mathbf{S}_{n+1}$ at stage $n+1$ as follows: obtain $\mathbf{C}_{n+1}$ and $\mathbf{X}_{n+1}$ from the pre-collected sequence $\{(\mathbf{C}_n, \mathbf{X}_n) : n = 1, 2, \ldots, N\}$; based on C_n, obtain B_{n+1} and Y_{n+1} according to (2) and (4), respectively. Thus, we have the sample $\mathbf{S}_{n+1} = (\mathbf{C}_{n+1}, \mathbf{X}_{n+1}, \mathbf{B}_{n+1}, \mathbf{Y}_{n+1})$. At each stage n, we use a batch of latest q samples and choose:

$$\sum_{i=n-q}^{n-1} \frac{1}{q} (\log \pi(\mathbf{S}_i, R_i; \boldsymbol{\Theta}_\pi)(g(\mathbf{S}_i, R_i) + \gamma V^\pi(\mathbf{S}_{i+1}; \boldsymbol{\Theta}_v)) - \xi (g(\mathbf{S}_i, R_i) + \gamma V^\pi(\mathbf{S}_{i+1}; \boldsymbol{\Theta}_v) - V^\pi(\mathbf{S}_i; \boldsymbol{\Theta}_v))^2 + \beta \sum_{r\in\mathcal{R}} \pi(\mathbf{S}_i, r; \boldsymbol{\Theta}_\pi) \log \pi(\mathbf{S}_i, r; \boldsymbol{\Theta}_\pi)) \tag{11}$$

as the loss function for jointly training the eA3C actor and critic networks. Here, $V^\pi(\cdot; \boldsymbol{\Theta}_v)$ is the output of the critic network, $\sum_{r\in\mathcal{R}} \pi(\mathbf{S}_i, r; \boldsymbol{\Theta}_\pi) \log \pi(\mathbf{S}_i, r; \boldsymbol{\Theta}_\pi)$ is the entropy regularization term and β is the associated weight for the entropy. The entropy regularization term is used to ensure adequate exploration of the action space for discovering good policies. Large β encourages policy exploration, whereas small β encourages policy exploitation. We train eA3C network using the root mean squared propagation algorithm.

C. Comparisons with the A3C Method

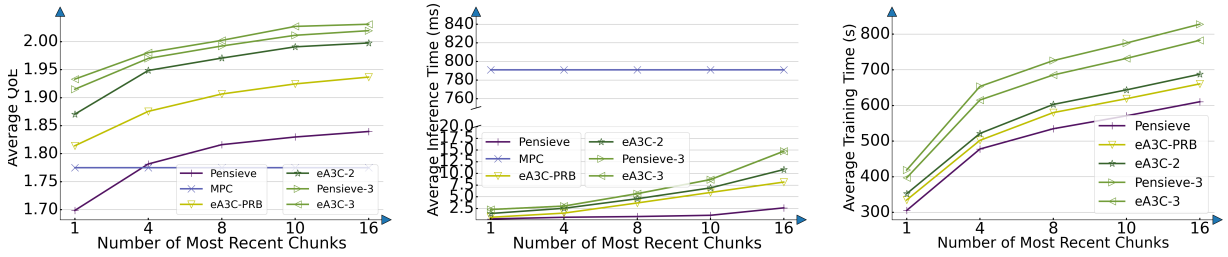
First, we compare the optimization formulations and solution methods for the policy and value parameters. The presented joint optimization of policy and value parameters in (10) comes from an equivalent transformation of the problem in (6) which captures more interactions between policy parameters $\boldsymbol{\Theta}_\pi$ and value parameters $\boldsymbol{\Theta}_v$ and can be solved by standard stochastic gradient method with guarantee to converge to stationary points. By contrast, in [2], [6], the problem in (6) is separated into the two optimization problems for $\boldsymbol{\Theta}_\pi$ and $\boldsymbol{\Theta}_v$ which are solved alternatively without any theoretical guarantee to converge to stationary points. Next, we compare the neural network structures and training methods. Firstly, compared to the A3C network in [2], [6], the eA3C network has additional input that is lower-layer information $\mathbf{X}_m, m = 1, \ldots, M$. Secondly, unlike [2], [6] that train the actor and critic networks of the A3C network alternatively, we jointly train these two networks of the eA3C network (i.e., jointly optimize the policy and value parameters) which may reduce the training time and improve the QoE. Therefore, it is expected that our proposed eA3C method has higher QoE and shorter training time than the state-of-art A3C method [2], [6].

V. Performance Evaluation

A. Experimental Setup

We consider adaptive streaming of one video, i.e., *EnvivioDash3* provided by [9]. The video lasts 192 seconds and is encoded into 48 chunks, each of 4 seconds. That is to say, we set $N = 48$ and $T = 4$ seconds. We set $D = 6$ and set $r_d, d \in \mathcal{D}$ according to [12, Table IV]. We set $B = 60$ seconds and adopt the utility function in [2], i.e., $U(R) = \log(\frac{R}{r_1})$. We conduct five experiments under different network environments and collect 5 datasets. Each dataset has multiple traces, each containing 200 samples. Each sample consists of the APP layer throughput and $M = 3$ lower-layer quantities including MAC rate, PRB number, and MCS index. The distributions of the APP layer throughput, MAC rate, PRB number, and MCS index across all datasets are shown in [12, Fig. 5]. For all $i = 1, 2, \ldots, 5$, we partition the i-th dataset into three subsets with 60%, 20%, and 20% of samples, respectively, and merge all samples into one training/validation/testing set. We choose $k = 8$ as in [2], which can achieve a good balance among QoE, training time, and inference time, unless otherwise specified. We set $\gamma = 0.99$, $\xi = 10$. The entropy weight β decays from 3 to 0.1 during training.

We consider three instances of the proposed eA3C method which use different lower-layer information. Specifically, the instances with one lower-layer quantity ($M = 1$) being PRB number index is referred to as eA3C-PRB; the instance with two lower-layer quantities ($M = 2$) being MCS index and PRB number is referred to as eA3C-2; and the instance with all three lower-layer quantities ($M = 3$) is referred to as eA3C-3. We consider three baseline methods, including Pensieve [2], Pensieve-3 (an enhanced version of Pensieve [2] which utilizes all three lower-layer quantities), and MPC [10]. Specifically,



(a) Average QoE (b) Average inference time (c) Average training time

Fig. 4: Different performance metrics of proposed and baseline schemes under different numbers of most recent chunks k.

Pensieve (Pensieve-3) adopts the A3C method [6] and trains the policy network and value network of the A3C network alternatively. MPC adapts the video bitrate according to the current APP layer throughput and buffer occupancy based on model predictive control optimization method [8].

B. Experimental Results

Fig. 4 shows the average QoE, inference time, and training time versus the number of most recent chunks (k). We can make the following observations. Firstly, the average QoE, inference time, and training time of the proposed eA3C method increase with M, and the average QoE, inference time, and training time of each DRL-based method increase with k. These phenomena imply that the tradeoff among average QoE, inference time, and training time can be achieved by choosing different M and k. Secondly, eA3C-3 outperforms Pensieve-3 in the average QoE and training time and has the same inference time as Pensieve-3. The gains of eA3C-3 over Pensieve-3 in average QoE and training time come from the joint optimization of the policy and value parameters. Their identical inference time derives from the fact that their optimized networks have the same structure. Thirdly, the proposed eA3C method outperforms Pensieve in the average QoE at the cost of increased training time and inference time. The gains in QoE of the three instances of the proposed eA3C method over Pensieve ($6.8\% \sim 13.8\%$) mainly come from utilizing the lower-layer information (at different amounts). The increased training time and inference time are due to the more complex structures of the optimized eA3C networks for effectively utilizing lower-layer information. Fourthly, the proposed eA3C method outperforms MPC in the average QoE and inference time. The gains of the three instances of the proposed eA3C method over MPC in the average QoE ($9.1\% \sim 14.4\%$) come from wise utilization of past and lower-layer information. Their gains over MPC in the average inference time are due to lower computation time for obtaining the bitrate via neural network than exhaustive search. Finally, eA3C-PRB and eA3C-2 at $k = 4$ outperform Pensieve at $k = 16$ in the average QoE, inference time, and training time, implying that a small amount of lower-layer information can compensate for the lack of a large amount of past APP layer information. It becomes evident that judicious utilization of lower-layer information can reduce the memory requirement without compromising performance.

VI. Conclusion

This paper focused on enhancing DRL-based adaptive wireless video streaming by incorporating lower-layer information and deriving a rigorous training method. We formulated a more comprehensive and accurate infinite stage discounted MDP problem for adaptive wireless video streaming. We presented an enhanced A3C method, eA3C, which improves the state-of-art DRL method, A3C, based on lower-layer information and a rigorous training method. Experimental results showed the superiority of the proposed eA3C method.

References

[1] R. Bhattacharyya, A. Bura, D. Rengarajan, M. Rumuly, S. Shakkottai, D. Kalathil, R. K. P. Mok, and A. Dhamdhere, "Qflow: A reinforcement learning approach to high qoe video streaming over wireless networks," in *Proc. of ACM Mobihoc*, Jul. 2019, pp. 251-260.

[2] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. of ACM SIGCOMM*, Aug. 2017, pp. 197-210.

[3] L. Zhao, Y. Cui, Z. Liu, Y. Zhang, and S. Yang, "Adaptive streaming of 360 videos with perfect, imperfect, and unknown FoV viewing probabilities in wireless networks," *IEEE Trans Image Process.*, vol. 30, pp. 7744-7759, 2021.

[4] Y. Xie, and K. Jamieson, "Ng-scope: Fine-grained telemetry for NextG cellular networks," in *POMACS*, vol. 6, no. 1, pp.1-26, 2022.

[5] Y. Zhang, G. Li, C. Xiong, Y. Lei, W. Huang, Y. Han, A. Walid, Y.R. Yang, and Z.L. Zhang, "MoWIE: toward systematic, adaptive network information exposure as an enabling technique for cloud-based applications over 5G and beyond," in *Proc. of ACM SIGCOMM NAI*, Aug. 2020, pp. 20-27.

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. of ICML*, Jun. 2016, pp. 1928-1937.

[7] D. Bertsekas, *Dynamic programming and optimal control: Volume I* (Vol. 1). Athena scientific, 2018.

[8] D. Bertsekas, *Dynamic programming and optimal control: Volume II* (Vol. 2). Athena scientific, 2018.

[9] "DASH Industry Form. 2016. Reference Client 2.4.0.," 2016. [Online]. Available: http://mediapm.edgesuite.net/dash/public/nightly/samples/dash-if-reference-player/index.html

[10] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. of ACM SIGCOMM*, Aug. 2015, pp. 325-338.

[11] D. Bertsekas, *Nonlinear programming*. Athena Scientific, 2016.

[12] L. Zhao, Y. Cui, Y. Jia, Y. Zhang, and K. Nahrstedt, "Enhancing neural adaptive wireless video streaming via lower-layer information exposure and online tuning," *Technical Report*, 2023. [Online]. Available:https://drive.google.com/file/d/1MT2nPZ5R6XymBQupqrUj-GNfytdjb9DL/view?usp=sharing