Intelligent Video Ingestion for Real-time Traffic Monitoring

XU ZHANG, University of Exeter, UK
YANGCHAO ZHAO, Beijing Kuaishou Technology Co. Ltd, China
GEYONG MIN and WANG MIAO, University of Exeter, UK
HAOJUN HUANG, Huazhong University of Science and Technology, China
ZHAN MA, Nanjing University, China

As an indispensable part of modern critical infrastructures, cameras deployed at strategic places and prime junctions in an intelligent transportation system can help operators in observing traffic flow, identifying any emergency situation, or making decisions regarding road congestion without arriving on the scene. However, these cameras are usually equipped with heterogeneous and turbulent networks, making the real-time smooth playback of traffic monitoring videos with high quality a grand challenge. In this article, we propose a lightweight Deep Reinforcement Learning-based approach, namely, *sRC-C* (smart bitRate Control with a Continuous action space), to enhance the quality of real-time traffic monitoring by adjusting the video bitrate adaptively. Distinguished from the existing bitrate adjusting approaches, *sRC-C* can overcome the bias incurred by deterministic discretization of candidate bitrates by adjusting the video bitrate with more fine-grained control from a continuous action space, thus significantly improving the Quality-of-Service (QoS). With carefully designed state space and neural network model, *sRC-C* can be implemented on cameras with scarce resources to support real-time live video streaming with low inference time. Extensive experiments show that *sRC-C* can reduce the frame loss counts and hold time by 24% and 15.5%, respectively, even with comparable bandwidth utilization. Meanwhile, compared to the-state-of-art approaches, *sRC-C* can improve the QoS by 30.4%.

CCS Concepts: • **Information systems** → **Multimedia streaming**; • **Computing methodologies** → *Reinforcement learning*; • **Computer systems organization** → Real-time systems;

Additional Key Words and Phrases: Traffic monitoring, video ingestion, bitrate control, deep reinforcement learning

This work was supported by National Key Research and Development Program of China under Grant No. 2018YFB2100804, the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 898588, the EU Horizon 2020 INITIATE Project under Grant No. 101008297, the Leading Technology of Jiangsu Basic Research Plan under Grant No. BK20192003, the National Natural Science Foundation of China under Grant No. 92067206, and the Chongqing Key Laboratory of Digital Cinema Art Theory and Technology under Grant No. 2021KF01. This article reflects only the authors' view. The European Union Commission is not responsible for any use that may be made of the information it contains.
Authors' addresses: X. Zhang, G. Min (corresponding author), and W. Miao, Department of Computer Science, College of Engineering, Mathematics, and Physical Sciences, University of Exeter, Exeter, EX4 4QF; emails: {x.zhang6, g.min, wang.miao}@exeter.ac.uk; Y. Zhao, Beijing Kuaishou Technology Co. Ltd, Beijing 100085, China; email: zhaoyangchao@smail.nju.edu.cn; H. Huang, School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China; email: hhj0704@hotmail.com; Z. Ma, School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, Jiangsu, China; email: mazhan@nju.edu.cn.

1550-4859/2022/09-ART47 $15.00
https://doi.org/10.1145/3529511

ACM Reference format:
Xu Zhang, Yangchao Zhao, Geyong Min, Wang Miao, Haojun Huang, and Zhan Ma. 2022. Intelligent Video Ingestion for Real-time Traffic Monitoring. *ACM Trans. Sen. Netw.* 18, 3, Article 47 (September 2022), 13 pages. https://doi.org/10.1145/3529511

1 INTRODUCTION

With the rapid development of urbanization and the continuous increase of vehicles on roadways, an increasing number of social problems have emerged. For example, the urban mobility takes up 40% of all CO_2 emissions of road transport and up to 70% of other pollutants from transport in **European Union (EU)**. In addition, traffic congestion from vehicles costs EU nearly 1% of its GDP per year.[1] To address this problem, more and more financial supports are provided to build smart transportation infrastructure and bolster the adoption of **intelligent transportation system (ITS)**, which can provide consumers an access to a smarter, safer, and faster way to travelling [1–3]. As an indispensable part of ITS, real-time traffic monitoring system deployed at strategic places and prime junctions, can help operators in observing traffic flow, identifying any emergency situation or road congestion without arriving on the scene, where the videos are streamed to control centres for quick decision making and situation awareness. Furthermore, the videos can be further analyzed for traffic flow prediction, vehicle identification, and so on.

A typical framework of a traffic monitoring system usually encompasses two critical processes, that is, the video ingestion process where the camera uploads the live video to the media server, and the video analysis process where the uploaded videos will be observed by operators for quick decision making and real-time situation awareness, and analysed by machine for future traffic flow prediction [4], vehicle classification [5], and so on. However, one of the key limitations faced by the traffic monitoring system is that the quality of video analysis is fundamentally constrained by the video ingestion process due to the frequent network fluctuation. Specifically, the **Quality-of-Service (QoS)** of video transmission from the traffic cameras to the control centres is of great significance when operators make a decision based on the real-time traffic monitoring videos. Flaws, such as rebuffering or unsmoothness, may result in a wrong decision in the situations like determination of liability for traffic accidents. Although many traffic monitoring systems have already been deployed around the world, how to ensure the real-time smooth playback of traffic monitoring video with high quality is still a grand challenge due to the heterogeneous and turbulent network. Different from traditional professional live video services, videos from outdoor cameras are generated with limited computing resource and uploaded via the unstable access networks, which incurs 17% rebuffers [6]. On the one hand, network congestion always happens from time to time, especially when the network path is shared with numerous cameras. On the other hand, the access networks are always unstable and fragile, especially when some cameras are connected to the Internet with cellular networks.

To cope with the bottleneck incurred by the turbulent network of the video ingestion, a promising approach is to adjust the sending bitrate according to the fluctuation of the network bandwidth and latency. Actually, there exist substantial works on improving the video streaming quality within dynamic networks [7]. These works can be categorized into two classes. The first class is adaptive bit rate streaming approaches for **video on demand (VoD)**, which transcode the original video into chunks with several candidate bitrates while viewers select a specific bitrate for each trunk adaptively. The selection process takes into consideration of the buffer occupancy, the estimated bandwidth, or both, based on statistical models [8] or machine learning models [9].

[1] https://ec.europa.eu/transport/home_en.

Another type of works is rate control methods for live video streaming, which controls the sending bitrate by estimating the network available bandwidth, thus making full use of the limited bandwidth of the uploading path [10]. However, these works select the bit rate from a discrete and small decision space with sacrificed QoS, thus striking a balance between QoS and the system efficiency when serving millions of users geographically distributed around the world. Nevertheless, in the video ingestion process of a traffic monitoring system, the videos are uploaded to a specific media server, making fine-grained bitrate adjusting from a continuous decision space feasible.

In this article, we propose a lightweight **Deep Reinforcement Learning (DRL)**-based approach *sRC-C* (smart bitRate Control with a Continuous action space) to enhance the quality of the video ingestion process of a traffic monitoring system by adjusting the video bitrate adaptively in a realtime manner. *sRC-C* can learn a control policy from the historical observations without using any pre-programmed rules or assumptions about the underlay network. Besides, instead of selecting a value from pre-given candidate bitrates, *sRC-C* outputs the optimal bitrate for the next segment from a continuous action space, overcoming the bias incurred by deterministic discretization of existing bitrate control approaches. Moreover, considering the limited computation capacity and energy consumption at the cameras, *sRC-C* adopts simple fully connected neural network model and carefully designed state space, making it lightweight and feasible on cameras with limited computing resources.

In detail, *sRC-C* aims to keep the buffer in the sender at a "safe range" stably, balancing the transmission latency and the network bandwidth utilization. It runs at the cameras and learns an optimal bitrate according to the past buffer occupancy and throughput information at both the decision interval level and frame interval level, combined with the past bitrate decisions and buffer instantaneous variations. Furthermore, *sRC-C* adopts the trust region-based method PPO [11] to train the RL-based model with a continuous action space based on both real-world and synthesized network trace. To the best of our knowledge, our work is the first of its kind to enhance the video ingestion process by adjusting the video bitrate with more fine-grained control from a continuous action space, overcoming the bias incurred by deterministic discretization of existing bitrate adjusting approaches. This is reasonable as the video ingestion is from the camera to a specific media server, distinguished from the video distribution side where one video is delivered to hundreds of thousands of users. To verify the superior of *sRC-C*, we compare it with the state-of-the-art approaches. Simulation results show that *sRC-C* can reduce the frame loss occurrence and hold time by 24% and 15.5%, respectively, and improve QoS by 30.4% with comparable bandwidth utilization.

The major contributions of the article are summarized as follows:

- We present *sRC-C*, a DRL-based approach to enhance the quality of the video ingestion process in a traffic monitoring system by adjusting the video bitrate adaptively in a real-time manner. *sRC-C* can select the optimal bitrate from a continuous action space, overcoming the bias incurred by deterministic discretization of the existing bitrate adjusting approaches.
- *sRC-C* adopts the fully connected neural network model and carefully designed state space, making it lightweight and feasible on cameras with limited available resources.
- We evaluate *sRC-C* with real network traces and compare it with the state-of-the-art approaches. The results demonstrate the superiority of our proposed method.

The rest of the article is organized as follows. Section 2 introduces the background of a traffic monitoring system and summarizes the related work on improving QoS of the video transmission under dynamic networks. Section 3 elaborates the design details of *sRC-C*, while Section 4 evaluates the performance of *sRC-C* with real-world network traces. Finally, we draw the conclusions in Section 5.

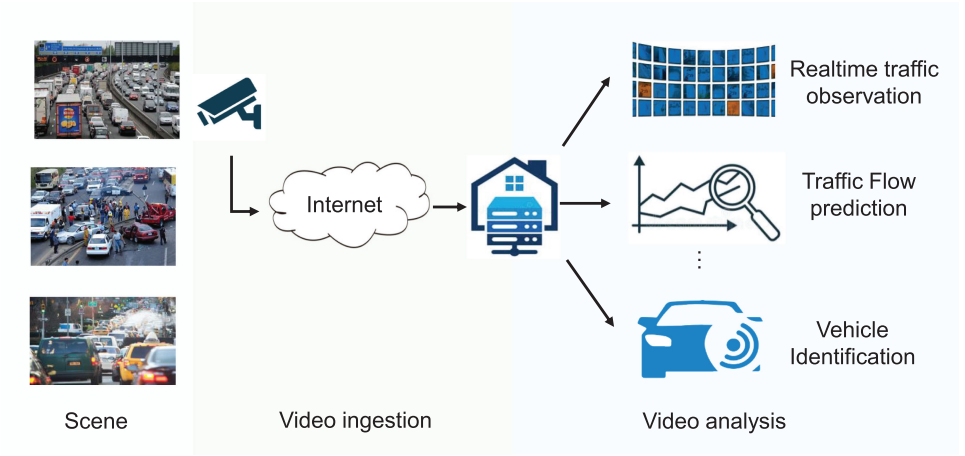


Fig. 1. The pipeline of a traffic monitoring system.

2 BACKGROUND AND RELATED WORK

Traffic monitoring system. Figure 1 depicts the framework of a typical traffic monitoring system, which encompasses two critical processes. In the video ingestion process, the scene is captured into raw data by the cameras deployed at strategic places and prime junctions, which encode the raw data into a high-quality video. The video is then uploaded via the first mile network to an media server for further processing. In the video analysis process, the operators can view the uploaded video for quick decision making and real-time situation awareness, while the media server can analyze the videos for future traffic flow prediction, vehicle identification, and so on. As the overall performance of the system depends on the quality of video ingestion, uploading the videos with high quality and low latency is of paramount significance. However, the heterogeneous and turbulent networks of the cameras make the real-time video uploading with high quality a challenging task.

Live video quality enhancement. Vantage [12] takes into account the time-shifted characteristics of social live video streaming and the relationship between the video quality metrics (SSIM) and bitrate, leveraging the surplus bandwidth resources during the period of high bandwidth to re-upload the low-quality frames. LiveNAS [13] utilizes the computation resources at the media server to apply neural super-resolution on the original stream, thus breaking the strong dependency between the quality of live video and the ingesting client's bandwidth. To the best of our knowledge, our work is orthogonal with these approaches and further improvements can be achieved by integrating them.

Rate control Methods. There are extensive work on real-time rate control [10, 14–16]. GCC [17] has been proposed for the congestion control task in WebRTC, which controls the sending bitrate by combining the delay-based controller placed at the receiver and the loss-based controller in the sender. Rebera [15] decides whether to discard an encoded frame with the prediction of available bandwidth, with the objective of maximizing the video transmission rate for interactive video calls between two users. There also exist learning-based approaches to control the bit rate adaptively. T-Gaming [16] adjusts the rate to network fluctuation based on deep reinforcement learning. Distinguished with Reference [16], which selects one bitrate from given candidates, our work can make fine-grained adjusting from a continuous decision space.

Adaptive bit rate (ABR) streaming. Many efforts have been devoted to the adaptive bitrate streaming for VoD [8, 10, 14, 18–21]. BBA [22] decides the bitrate of video chunks based on the

buffer occupancy information while [23] made decisions according to the estimated available bandwidth. Recently, motivated by the success of deep learning [24–27], there emerge works applying learning-based algorithms to adaptive bitrate [9, 28]. Pensieve [9] is the most representative one in the works, which combine learning methods with ABR tasks. Nevertheless, Pensieve was implemented for chunk-based VoD systems, which has slack limit on the delay. However, these works always attempt to select one video bitrate from pre-given candidates, which make the decision less efficient.

3 METHODS

3.1 Overview

The access networks of cameras are always highly dynamic, which hinder the accurate prediction of the available network bandwidth. Instead of adjusting the bitrate based on predicted network bandwidth, another significant indicator of whether the current video bitrate is suitable or not is the buffer occupancy in the streaming device. When the buffer occupancy grows too fast, buffer overflow is likely to happen, leading to frame loss or even rebuffering events. When the buffer occupancy decreases rapidly, it means that the bandwidth utilization is poor, resulting in low video quality. To guarantee the liveness of the video streaming, the buffer occupancy should be kept as low as possible. In other words, a good solution should keep the sending buffer of the camera stably at a certain "safe range."

Actually, buffer-based adaptive bitrate solutions has already been explored in VoD streaming, such as the BBA [22]. However, they are not straightforward to be used in rate control for a social live video streaming. On the one hand, they can only select one from pre-given candidate bitrates, while the camera can have a wide and continuous optional range. On the other hand, they rely on the proper tunable parameters, which are difficult to be optimized and only work under the specific scenario.

To this end, we propose a DRL-based approach *sRC-C* with a continuous action space to keep the buffer stable and within a "safe range." *sRC-C* maps the historical observations on the buffer, throughput, and bitrates to the sending bitrate. As the cameras are always with limited computation resources, we try to make our network structures as simple as possible while guaranteeing the performance. Specifically, *sRC-C* adopts fully connected neural networks instead of using **recurrent neural networks (RNN)** to simplify the DRL model. The agent gets the states from the environment and takes actions regarding the bitrate decision based on the policy, which is parameterized by a deep neural network. To train the model effectively, we have implemented a simulation environment in Python at the frame level for the live video streaming in a real-time manner.

3.2 Design

Specifically, *sRC-C* is based on deep reinforcement learning, which is defined by three key elements, that is, states, actions, and rewards. Specifically, it considers a general setting where an agent interacts with an environment in an iterative manner. For each time step, the agent observes some states from the environment, and then determines an action with the objective to maximize the expected cumulative discounted reward. In what follows, we will describe how we define the state s_t, the action a_t, the reward r_t, and the training approach, respectively.

States: The key idea behind the state space design is to strike a good balance between the performance and the capacity of computation resources and energy on the cameras. Intuitively, the more information the DRL can obtain, the higher performance it can achieve, but the higher computation and energy consumption it would cost. In detail, instead of solely considering the buffer at the sender, *sRC-C* takes into account not only the observations of the buffer but also the

Table 1. The Details of the States

Feature	Sampling Interval	Description
$\overrightarrow{BL_t}$	Decision Interval	Buffer occupancies over past decision moments.
$\overrightarrow{A_t}$	Decision Interval	Past bitrate decisions.
$\overrightarrow{TL_t}$	Decision Interval	Average throughputs over the past decision intervals.
$\overrightarrow{\Delta B_t}$	Decision Interval	Differences between the buffer occupancy at the moment of decisions and the buffer occupancy at the frame interval before the decisions.
$\overrightarrow{BS_t}$	Frame Interval	Buffer occupancies when the past frames were encoded.
$\overrightarrow{TS_t}$	Frame Interval	Average throughputs over the past frame intervals.

throughput to make decisions. Due to the limited capacity of computation resources and energy on the cameras, the decision interval is set to 1 s to avoid too-frequent bitrate adjusting. During the decision interval, it is likely the available bandwidth on the network has already changed. Therefore, instantaneous information at the frame interval about the throughput and the buffer occupancy are also considered in *sRC-C*.

As listed by Table 1, we denote the state as $s_t = \{\overrightarrow{BL_t}, \overrightarrow{A_t}, \overrightarrow{TL_t}, \overrightarrow{\Delta B_t}, \overrightarrow{BS_t}, \overrightarrow{TS_t}\}$, which encompasses historical observations collected at Decision Interval and Frame Interval. The past information collected at Decision Interval (around 1 s) contains patterns from a relatively long time period, while the information collected at Frame Interval (around 30 ms) includes the patterns from a short time period. Specifically, where $\overrightarrow{BL_t}$ is the buffer occupancy over past decision moments, $\overrightarrow{A_t}$ is the past bitrate decisions, $\overrightarrow{TL_t}$ is the average throughput over the past decision intervals, $\overrightarrow{\Delta B_t}$ is the differences between the buffer occupancy at the moment of decisions and the buffer occupancy at the frame interval before the decisions, $\overrightarrow{BS_t}$ is the buffer occupancies when the past frames were encoded, while $\overrightarrow{TS_t}$ is average throughputs over the past frame intervals. Formally, if we denote the frame interval as t_f, the decision interval as t_d, and the last i_{th} item of the frame-interval-level buffer occupancy vector at time t as BS_i^t, then BS_i^t is computed by

$$BS_i^t = B_{t-i*t_f}, \tag{1}$$

where the B_t represents the buffer occupancy at time t. While the last i_{th} item of $\overrightarrow{\Delta B}$ at time t is computed by

$$\Delta B_i^t = B_{t-i*t_d} - B_{t-i*t_d-t_f}. \tag{2}$$

This can approximate the buffer occupancy gradient, which is important to infer the transmission states when making a bitrate choice.

Action: Current mainstream video delivery platforms, such as Youtube and Netflix, always provide video services to millions of viewers simultaneously. In general, they often transcode a video into several typical candidate bitrates and offer a viewer the opportunity to select one bitrate from the discretized candidates according to its network condition. This kind of coarse-grained adaption can enhance the overall system efficiency significantly, but can compromise the overall video quality. Distinguished from them, the cameras deployed for traffic monitoring are only responsible for uploading the videos to a specific media server, making fine-grained bitrate adjusting from a continuous decision space feasible.

Specifically, a continuous action space $[R_{min}, R_{max}]$ is constructed, which means that the sending bitrate can be any value between R_{min} and R_{max}. *sRC-C* will output the mean μ_t^b of the bitrate and its variance σ_t^b, where $R_{min} \leq \mu_t^b \leq R_{max}$, and $0 \leq \sigma_t^b \leq 0.1$. To improve the ability of exploration, the bitrate value a_t is sampled from a normal distribution:

$$a_t \sim N\left(\mu_t^b, \sigma_t^b\right). \tag{3}$$

To guarantee that a_t is between R_{min} and R_{max}, we update it based on

$$a_t \leftarrow clip(a_t, R_{min}, R_{max}), \tag{4}$$

where $clip(x, min, max)$ denotes restricting the variable x in range $[min, max]$:

$$clip(x, min, max) = \begin{cases} min & x < min, \\ x & min \leq x \leq max, \\ max & x > max. \end{cases} \tag{5}$$

Reward: *sRC-C* aims at keeping the buffer within a relative low range $[B_d, B_u]$. For the reward function, we mainly consider the rationality of the actions and the buffer occupancy and the QoS, represented as "action score" r_t^A, "buffer score" r_t^B and "QoS score" r_t^Q, respectively. And the reward function r_t is denoted as

$$r_t = \lambda_1 r_t^A + \lambda_2 r_t^B + \lambda_3 r_t^Q, \tag{6}$$

where $\lambda_1, \lambda_2, \lambda_3$ is the weight for the three scores, respectively.

For the "action score" r_t^A, the action will be preferred, which holds the bitrate stable when the buffer is under the model's control for the sake of smoothness and the codec's efficiency. Moreover, actions against the "intelligent choices" will be penalized such as increasing the bitrate when buffer is out of control. r_t^A is computed based on the buffer capacity, bitrate changes, and the gradient of the sending data size:

$$r_t^A = \begin{cases} 0 & B_d \leq B_t \leq B_u, \\ & |(R_t - R_{t-t_d})/R_{t-t_d}| < \varepsilon \\ -2.0 & B_t < B_d || B_t > B_u, \\ & D_t - D_{t-t_d} < 0, R_t > R_{t-t_d} \\ -2.0 & B_t > B_u, R_t > R_{t-t_d} \\ -2.0 & B_t < B_d, R_t < R_{t-t_d} \\ -1.0 & otherwise, \end{cases} \tag{7}$$

where D_t represents the data size sent at time t while ε weights the acceptable bitrate change ratio. Note that we do not punish wrong buffer occupancy. Instead, we exhaust all the possible buffer occupancy and punish wrong actions in each buffer occupancy circumstance.

For the "buffer score" r_t^B, we penalize the action of the agent, which makes the buffer occupancy out of the ideal range $[B_d, B_u]$,

$$r_t^B = \begin{cases} 0 & B_d \leq B_t \leq B_u, \\ -1 & otherwise. \end{cases} \tag{8}$$

If the buffer occupancy B_t is within the ideal range $[B_d, B_u]$, then the buffer score r_t^B will be set to be 0, otherwise -1. In this way, *sRC-C* aims at keeping the buffer occupancy lower than B_{th} to reduce the transmission latency.

For the "QoS score" r_t^Q, we consider the standard quality metrics in industry referring to Reference [29], which encompasses four aspects: the latency, the buffer overflow frequency,[2] the buffer

[2]The ratio between the total number of overflow events and the trace duration.

overflow ratio,[3] and the average bitrate. As for the latency metric, we approximate it by the median of the buffer occupancy in practice. Note that our buffer is in the streaming device, so we focus on the buffer overflow instead of buffering events in the receiver. Last, we leverage the bandwidth utilization to replace the average bitrate metrics. In conclusion, r_t^Q is denoted as the weighted sum of the four metrics similar to [30]

$$\begin{aligned} r_t^Q = &- \alpha * bufOccu - \beta * overFreq - \zeta * overRatio \\ &- \omega * (1 - bwUtil), \end{aligned} \tag{9}$$

where $bufOccu$, $overFreq$, $overRatio$, and $bwUtil$ denote the Q3 (Third quartile) value of the buffer occupancy, the overflow frequency, the overflow ratio and the bandwidth utilization, respectively.

Training Approach: *sRC-C* adopts the trust region-based method PPO [11] to train the RL-based model with a continuous action space reinforcement learning, which is the advanced version of TRPO [31]. Different from the policy gradient method, PPO formulates the learning process as an optimization problem:

$$\max_{\theta} \ \mathbb{E}_t[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \tag{10}$$

where θ_{old} denotes the network parameter before its one-step updating, $r_t(\theta)$ is given by $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and the advantage function A_t (same as $A(a_t|s_t)$) can be estimated according to Equation (11) leveraging a critic network. It represent how better the certain action a_t is than taking a arbitrary actions sampled from the policy to optimize given state s_t, can be approximated by

$$A(s_t, a_t) = r_t + \gamma V^{\pi_\theta}(s_{t+1}; \theta_v) - V^{\pi_\theta}(s_t; \theta_v), \tag{11}$$

where the $V^{\pi_\theta}(s_t; \theta_v)$ is the value function of state s_t, which means the expectation of the cumulative discount reward from state s_t following the policy $\pi_\theta(a_t|s_t)$.

4 EVALUATION

4.1 Experiment Setup

We implement the model with Tensorflow's Python API. To train the model more effectively and flexibly, we first implement a simulation environment according to video ingestion process of a live video streaming. In this way, we can train several agents simultaneously with the parallel-actor-based reinforcement learning methods, and make each of them experience different network traces and different video traces (because we use arbitrary video frame sizes), which makes the model easier to converge.

Simulation environment. The simulation environment takes the available bandwidth traces as input, and approximates the uploading pipeline from the encoding process to the sending of bits. The buffer in the streaming device is set to with the capacity of 5 s and is carried out at the frame level for convenience. Current codecs can not guarantee the output bitrate of a single frame is exactly a given value. For instance, if we want to output the video streaming at an average speed of 1 Mbps, the streaming that we actually obtain may be with the instantaneous speed of 0.9 Mbps. This is because that the contents and motions in the video are unpredictable. So, we sample it from a uniform distribution $S_f^R \sim U(S_{avg}^R * 0.8, S_{avg}^R * 1.2)$ to get a frame size S_f^R based on the average frame data size S_{avg}^R at a certain bitrate R instead of using a series of video size traces. In this way, the model will perform more robust for different videos when applying it to real systems. What is more, we assume that the size of an I-frame is 3 to 5 times of a P-frame in average with the IPPP structure.

[3]The fraction of the cumulative hold time of overflow events over the trace duration.

Network Traces. To make our model more robust to various network conditions, we collect the training bandwidth traces from available public datasets: broadband upload traces from **Federal Communications Commission (FCC)** [32], the 4G wireless bandwidth traces collected on mobile devices from Ghent [33], and the 3G HSDPA-bandwidth logs from mobile HTTP steaming scenarios [34]. What is more, we synthesize some traces in the sine waves and rectangular waves styles for the sake of intuitive analysis. We use the network traces from Reference [35] for the performance evaluation, which contains the bandwidth logs from three major ISPs: AT&T, TMobile, and Verizon.

Evaluation Setup. When evaluating the performance of our model, we choose the network traces from the Mahimahi dataset [35]. The trace set consists of 11 traces totally, where 8 traces from Mahimahi dataset, 1 trace from Norway 3G cellular networks, and 2 synthesize traces in sine wave and rectangular wave styles, respectively. All these traces cover more than 100 min totally. As for the network traces from Mahimahi, 3 of them have an average available bandwidth higher than 5 Mbps, which is set to be the upper limit of the bitrates. We set the decision interval as 1 s, which is the highest bitrate switch frequency acceptable for the codec. The videos are configured with 15 FPS and GOP length of 45 frames, that is, 3 s for 1 GOP. And we set the ideal buffer range $[B_d, B_u]$ as [0.2 s, 1 s] and the acceptable bitrate change ratio ε is set to be 0.1. Moreover, to make the four submetrics have approximately equal contributions to the QoS, we set $\alpha = 1, \beta = 50, \zeta = 20, \omega = 10$ to evaluate the buffer occupancy, the overflow frequency, the overflow ratio and the bandwidth utilization to *QoS*, respectively.

Our models are trained on a machine with Intel Core i7-7700K CPU @ 4.20 GHz processor, using the CPU mode of tensorflow 1.13.1. The training of the model takes tens of minutes and 10^4 episodes, each of which covers a period of 100 s monitoring video with eight parallel agents.

Alternative Approaches. To verify the performance of our proposed method, we also implement a similar RL-based approach with discrete action space (referred as to *DRL*) [16], which selects the next bitrate from a discrete action space and trains the neural network with the actor-critic algorithm A3C. Besides, we replace the fully connected neural network model in *DRL* with LSTM-based approach (referred as to *LSTM-D*) [27]. Specifically, *sRC-C* and *DRL* exploit a fully connected network with one single hidden layer of 256 neurons while the *LSTM-D* model encompasses a LSTM cell with 128 hidden states and the time length of 6. In addition, we migrate the representative buffer-based ABR approach *BBA-0* [22] to the social live streaming scenario, as an alternative approach. Note that BBA-0 is used for video trunk downloading for the VoD service with a large buffer in the receiver. We just leverage the idea to use a linear function mapping the buffer occupancy to the target bitrate for the next video chunk and modify the algorithm proper for the video monitoring settings. Besides, we implement an ideal bandwidth estimation methods using a sliding window with the length of the decision interval applying to the real bandwidth traces (denoted as *BWE*). With this methods, we just decide the target bitrate according to the real average available bandwidth from last decision to current time with a factor of 0.95, in case the available bandwidth suddenly decrease.

4.2 Experiment Results

Figure 2 shows the comparison between the performance of *sRC-C* and that of four alternatives. We mainly focus on the buffer overflow counts and buffer overflow hold time, which represent the frequency and duration of frame loss, respectively. What is more, the bitrate hold time can reflect the video smoothness and the extra overhead caused by switching bitrate for the codec. As depicted in the figure, *sRC-C* outperforms its rivals in terms of reducing the frame loss (i.e., the buffer overflow counts and duration) but switching the codec bitrate more frequently. Compared with the state-of-the-art approach *DRL*, *sRC-C* can reduce the frame loss frequency and its duration

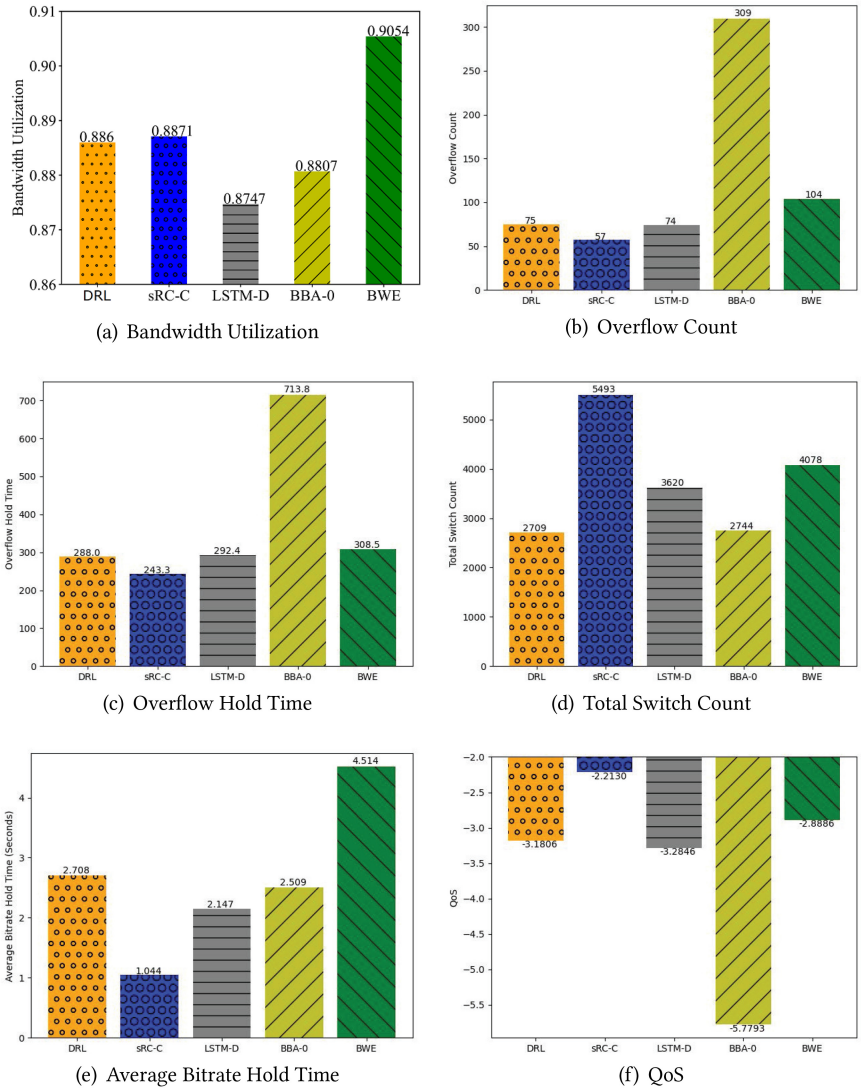
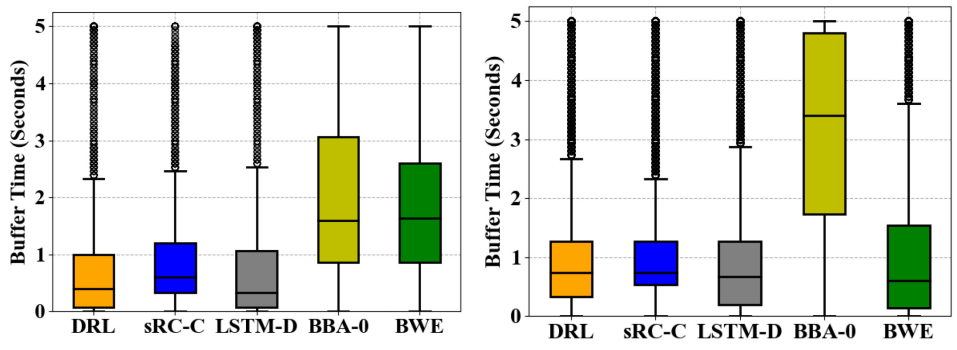


Fig. 2. The performance overview of the five methods.

by 24% and 15.5%, respectively. Furthermore, *sRC-C* outperforms *BWE*, which decides the bitrate based on the real bandwidth, and can reduce the frame loss frequency and its duration by 45.2% and 21.1%, respectively. And we can find that *LSTM-D* and *DRL* can achieve similar performance in our experiments, as they only differ from each other in terms of the policy network architecture.

Bandwidth utilization. To guarantee the smoothness and liveness of the video monitoring, the bandwidth utilization ratio achieved by *sRC-C* is slightly lower than *BWE*. This is because that BWE is based on the real bandwidth between the streaming device and the media server, which can hardly be achieved in practice. However, *sRC-C* has the highest bandwidth utilization ratio than the other three alternatives, followed by *DRL* and *BBA-0*.

Quality-of-Service. As shown in Figure 2(f), *sRC-C* can achieve the highest QoS compared with its rivals, followed by *BWE* and *DRL*, while *BBA-0* performs the worst. Specifically, *sRC-C* can improve the QoS by 23.4% compared to*BWE* and 30.4% compared to *DRL*.



(a) Trace1 (b) Trace2

Fig. 3. The box-plot of buffer occupancy for two network traces. Trace1 is a network trace from ATT-LTE, and Trace2 is a network trace from TMobile.

Buffer occupancy. It is crucial for the device of a live streamer to keep the buffer occupancy at a low level to achieve as high liveness as possible. Figure 3 shows the overall buffer occupancy box-plots of two network traces. We can see that *sRC-C* can keep the buffer occupancy mostly stay under 3 s, while the *BBA-0* cannot control the buffer effectively because of the small buffer capacity. This means our model can guarantee much lower latency caused by the sending buffer. However, low buffer occupancy would have stronger ability to adapt the dramatically fluctuant bandwidth.

Total Switch Count. As shown in Figure 2(d), *sRC-C* has the highest count. This is because *sRC-C* can adjust the bitrate in a fine-grained manner by selecting bitrates from a continuous action space, thus achieving better performance. However, a higher switch count will incur higher energy consumption. In the wild, operators can strike a balance between the performance and energy consumption at the camera by controlling the decision interval.

4.3 Computational Overhead

Note that *sRC-C* is suitable for video ingestion from the camera to a single specific media server. However, when applying it to more-general live-streaming, where the video is delivered to hundreds of thousands of users, it will incur high computation costs on the streaming server. In this scenario, the streaming server should compute which bitrate to send to each user, which brings so high overhead that the system cost will be intolerable. To evaluate the computational overhead of *sRC-C*, we deploy *sRC-C* on an android device with Tensorflow Lite API to simulate the camera with limited resources. The android device carries one Qualcomm Snapdragon 835 @ 2.45 GHz processor with 6 GB RAM. And we compare *sRC-C* with the other two learning-based approach *DRL* and *LSTM-D*, where the *LSTM-D* model encompasses a LSTM cell with 128 hidden states and the time length of 6. And we leverage the inference time to weigh the computational overhead needed by the AI models. Experiments shows that the average one-step inference time of *sRC-C*, and *LSTM-D* is 220.4 and 589.8 ms, respectively. Compared with *LSTM-D*, *sRC-C* can save 62.63% time consumption without performance degradation.

5 CONCLUSION

In this article, we propose *sRC-C*, a lightweight DRL-based approach with a continuous action space to make the best codec bitrate choice in cameras, enhancing the quality of live video ingestion by adjusting the video bitrate adaptively in a real-time manner. *sRC-C* can learn a control

policy from the historical observations without using any pre-programmed rules or assumptions about the underlay network, trading off the transmission latency and network bandwidth unitization by keeping the buffer of the sender at a low level stably. Moreover, considering the limited computation capacity on cameras, *sRC-C* adopts a fully connected neural network model and carefully designed state space, making it lightweight and feasible in the wild. Compared to the state-of-art approaches, simulations shows that *sRC-C* can reduce the frame loss occurrence and hold time by 24% and 15.5%, respectively, and improve QoS by 30.4% with comparable bandwidth utilization.

REFERENCES

[1] Christian Creß and Alois C. Knoll. 2021. Intelligent transportation systems with the use of external infrastructure: A literature survey. Retrieved from https://arXiv:2112.05615\/.

[2] Laizhong Cui, Dongyuan Su, Yipeng Zhou, Lei Zhang, Yulei Wu, and Shiping Chen. 2021. Edge learning for surveillance video uploading sharing in public transport systems. *IEEE Trans. Intell. Transport. Syst.* 22, 4 (2021), 2274–2285. DOI: http://dx.doi.org/10.1109/TITS.2020.3008420

[3] Yulei Wu, Hong-Ning Dai, and Hao Wang. 2021. Convergence of blockchain and edge computing for secure and scalable IIoT critical infrastructures in industry 4.0. *IEEE Internet Things J.* 8, 4 (2021), 2300–2317. DOI: http://dx.doi.org/10.1109/JIOT.2020.3025916

[4] Zichuan Liu, Rui Zhang, Chen Wang, and Hongbo Jiang. 2021. Spatial-temporal conv-sequence learning with accident encoding for traffic flow prediction. Retrieved from https://arXiv:2105.10478\/.

[5] Myounggyu Won. 2020. Intelligent traffic monitoring systems for vehicle classification: A survey. *IEEE Access* 8 (2020), 73340–73358.

[6] H. Pang, Z. Wang, C. Yan, Q. Ding, K. Yi, J. Liu, and L. Sun. 2019. Content harvest network: Optimizing first mile for crowdsourced live streaming. *IEEE Trans. Circ. Syst. Video Technol.* 29, 7 (2019), 2112–2125.

[7] Zhiqian Jiang, Xu Zhang, Wei Huang, Hao Chen, Yiling Xu, Jenq-Neng Hwang, Zhan Ma, and Jun Sun. 2020. A hierarchical buffer management approach to rate adaptation for 360-degree video streaming. *IEEE Trans. Vehic. Technol.* 69, 2 (2020), 2157–2170. DOI: http://dx.doi.org/10.1109/TVT.2019.2960866

[8] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *Proceedings of the IEEE 35th Annual IEEE International Conference on Computer Communications (INFOCOM'16)*. IEEE, 1–9.

[9] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 197–210.

[10] Tianchi Huang, Rui-Xiao Zhang, Chao Zhou, and Lifeng Sun. 2018. QARC: Video quality aware rate control for real-time video streaming based on deep reinforcement learning. In *Proceedings of the 26th ACM International Conference on Multimedia (MM'18)*. Association for Computing Machinery, New York, NY, 1208–1216. DOI: http://dx.doi.org/10.1145/3240508.3240545

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. Retrieved from https://arXiv:1707.06347\/.

[12] Devdeep Ray, Jack Kosaian, K. V. Rashmi, and Srinivasan Seshan. 2019. Vantage: Optimizing video upload for time-shifted viewing of social live streams. In *Proceedings of the ACM Special Interest Group on Data Communication*. ACM, 380–393.

[13] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'20)*. Association for Computing Machinery, New York, NY, 107–125. DOI: http://dx.doi.org/10.1145/3387514.3405856

[14] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the Google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys'16)*. Association for Computing Machinery, New York, NY, Article 13, 12 pages. DOI: http://dx.doi.org/10.1145/2910017.2910605

[15] Eymen Kurdoglu, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu. 2016. Real-time bandwidth prediction and rate adaptation for video calls over cellular networks. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys'16)*. Association for Computing Machinery, New York, NY, Article 12, 11 pages. DOI: http://dx.doi.org/10.1145/2910017.2910608

[16] H. Chen, X. Zhang, Y. Xu, J. Ren, J. Fan, Z. Ma, and W. Zhang. 2019. T-gaming: A cost-efficient cloud gaming system at scale. *IEEE Trans. Parallel Distrib. Syst.* 30, 12 (Dec. 2019), 2849–2865. DOI: http://dx.doi.org/10.1109/TPDS.2019.2922205

[17] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the Google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 13.

[18] R. Rejaie, M. Handley, and D. Estrin. 1999. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies: The Future is Now (INFOCOM'99)*, Vol. 3. 1337–1345.

[19] L. S. Brakmo and L. L. Peterson. 1995. TCP vegas: End to end congestion avoidance on a global internet. *IEEE J. Select. Areas Commun.* 13, 8 (1995), 1465–1480.

[20] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. 2018. Salsify:{Low-Latency} network video through tighter integration between a video codec and a transport protocol. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI'18)*. 267–282.

[21] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, Xin Yao, and Lifeng Sun. 2019. Comyco: Quality-aware adaptive video streaming via imitation learning. In *Proceedings of the 27th ACM International Conference on Multimedia (MM'19)*. Association for Computing Machinery, New York, NY, 429–437. DOI: http://dx.doi.org/10.1145/3343031.3351014

[22] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 187–198.

[23] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the ACM SIGCOMM Conference*. ACM, 272–285.

[24] Zhongxia Yan, Jingguo Ge, Yulei Wu, Liangxiong Li, and Tong Li. 2020. Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks. *IEEE J. Select. Areas Commun.* 38, 6 (2020), 1040–1057. DOI: http://dx.doi.org/10.1109/JSAC.2020.2986662

[25] Jin Cheng, Yulei Wu, Yuepeng E., Junling You, Tong Li, Hui Li, and Jingguo Ge. 2021. MATEC: A lightweight neural network for online encrypted traffic classification. *Comput. Netw.* 199 (2021), 108472. DOI: http://dx.doi.org/10.1016/j.comnet.2021.108472

[26] Xu Zhang, Zhengnan Qi, Geyong Min, Wang Miao, Qilin Fan, and Zhan Ma. 2022. Cooperative edge caching based on temporal convolutional networks. *IEEE Transactions on Parallel and Distributed Systems* 33, 9, (2022), 2093–2105.

[27] Xiang Gao. 2018. Deep reinforcement learning for time series: Playing idealized trading games. Retrieved from http://arxiv.org/abs/1803.03916.

[28] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: Auto-tuning video ABR algorithms to network conditions. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 44–58.

[29] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. 2011. Understanding the impact of video quality on user engagement. *ACM SIGCOMM Comput. Commun. Rev.* 41, 4 (2011), 362–373.

[30] Rui-Xiao Zhang, Ming Ma, Tianchi Huang, Haitian Pang, Xin Yao, Chenglei Wu, Jiangchuan Liu, and Lifeng Sun. 2019. Livesmart: A QoS-guaranteed cost-minimum framework of viewer scheduling for crowdsourced live streaming. In *Proceedings of the 27th ACM International Conference on Multimedia*. 420–428.

[31] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*. 1889–1897.

[32] FCC. 2018. Raw Data—Measuring Broadband America—Eighth Report. Retrieved from https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-eighth.

[33] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks. *IEEE Commun. Lett.* 20, 11 (2016), 2177–2180.

[34] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute path bandwidth traces from 3G networks: Analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 114–118.

[35] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate record-and-replay for HTTP. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC'15)*. 417–429.

Received November 2021; revised February 2022; accepted March 2022