LiveClip: Towards Intelligent Mobile Short-Form Video Streaming with Deep Reinforcement Learning

Jianchao He, Miao Hu
School of Data and Computer Science,
Sun Yat-sen University
Guangdong Key Laboratory of Big Data Analysis and Processing
Guangzhou, Guangdong, China

Yipeng Zhou
Department of Computing,
Faculty of Science and Engineering,
Macquarie University
Sydney, Australia

Di Wu
School of Data and Computer Science,
Sun Yat-sen University
Guangdong Key Laboratory of Big Data Analysis and Processing
Guangzhou, Guangdong, China

ABSTRACT

Recent years have witnessed great success of mobile short-form video apps. However, most current video streaming strategies are designed for long-form videos, which cannot be directly applied to short-form videos. Especially, short-form videos differ in many aspects, such as shorter video length, mobile friendliness, sharp popularity dynamics, and so on. Facing these challenges, in this paper, we perform an in-depth measurement study on *Douyin*, one of the most popular mobile short-form video platforms in China. The measurement study reveals that Douyin adopts a rather simple strategy (called *Next-One* strategy) based on HTTP progressive download, which uses a sliding window with stop-and-wait protocol. Such a strategy performs poorly when network connection is slow and user scrolling is fast. The results motivate us to design an intelligent adaptive streaming scheme for mobile short-form videos. We formulate the short-form video streaming problem and propose an adaptive short-form video streaming strategy called *LiveClip* using a deep reinforcement learning (DRL) approach. Trace-driven experimental results prove that LiveClip outperforms existing state-of-the-art approaches by around 10%-40% under various scenarios.

CCS CONCEPTS

• **Networks** → **Mobile networks**; **Network measurement**; • **Computing methodologies** → **Planning and scheduling**.

KEYWORDS

Measurements, reinforcement learning, short-form video

ACM Reference Format:
Jianchao He, Miao Hu, Yipeng Zhou, and Di Wu. 2020. LiveClip: Towards Intelligent Mobile Short-Form Video Streaming with Deep Reinforcement Learning. In *30th Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'20), June 10–11, 2020, Istanbul, Turkey*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386290.3396937

*Corresponding author: Miao Hu (humiao5@mail.sysu.edu.cn)

NOSSDAV'20, June 10–11, 2020, Istanbul, Turkey

ACM ISBN 978-1-4503-7945-8/20/06…$15.00
https://doi.org/10.1145/3386290.3396937

1 INTRODUCTION

In recent years, mobile video has experienced a great success in the market and occupied a large amount of traffic over the Internet. Cisco has announced that video traffic accounted for 59% of the world's mobile data in 2017, and it is expected to account for 79% by 2022 [4]. IHS Markit has pointed out that China's short-form video market will hit around 14 billion dollars by 2020, with Douyin valued between $8 billion and $10 billion [5].

Nevertheless, it is challenging to satisfy users' quality of experience (QoE) for short-form video applications because of unpredictable network conditions and frequent user interaction events such as impatient scrolls and early quits [6]. Recently, a few studies explored the characteristics of short-form video platforms. Zhang *et al.* [14] presented an in-depth measurement of Twitter's Vine, a mobile instant video clip sharing platform. Chen *et al.* [3] presented a study on characteristics of Douyin videos. However, they did not provide detailed solutions to improve short-form video services. Different from user stickiness on a long-form video, a short-form video is more likely to be skipped without viewing due to a scrolling event or early quit.

Facing the very special characteristics of user behaviors on short-form videos, it is inefficient to directly use existing video streaming strategies to guarantee users' QoE requirement for short-form videos. Recently, the reinforcement learning technique is becoming popular in the long-form video streaming design. Hu *et al.* [7] proposed an AP-assisted TV series prefetching paradigm based on reinforcement learning. Mao *et al.* [10] designed adaptive bitrate algorithms using reinforcement learning. Zhang *et al.* [15] adaptively allocated rates for tiles of future 360-degree video frames. Huang *et al.* [8] aimed to deliver videos with higher quality but at a lower sending rate. In a word, recent DRL-related work aims to work on bitrate selection for traditional long-form videos rather than mobile short-form videos, which are playlist-driven and usually have constant bitrate.

Despite the work mentioned above, the challenges remain for short-form video services. *First*, long delay is intolerable because the length of a typical short-form video clip is only of several seconds. *Second*, caching decisions have cascading effects. That is, smooth playback needs sufficient prefetched data, which will preempt the caching of other videos and cause freezes in the future. *Third*, wireless mobile network connection varies dynamically. Whereas, most short-form video service platforms, such as Douyin, heuristically adopted the HTTP progressive download strategy so that the streaming is not bitrate adaptive and thus is more vulnerable to network dynamics. *Last but not the least*, a user can easily

generate lots of scroll events in a very short period, which might lead to an extremely high cache miss ratio.

In this paper, we first conduct a systematic measurement study on Douyin, a popular short-form mobile video app in China. We collect the metadata of short-form videos as a dataset and analyze network traces in detail. Next, we formulate the short-form video streaming problem mathematically, and propose an adaptive streaming strategy, named *LiveClip*, employing the deep reinforcement learning (DRL) technique. *LiveClip* is especially designed for time-varying scenarios where an agent is to interact with the environment in a session. Moreover, deep neural network structure makes it capable to deal with high dimensional state space, and have a decent representation of policy and value estimation.

Our contributions in this paper are summarized as follows:

- We conduct an in-depth measurement study on Douyin, a leading mobile short-form video sharing app. We find that short-form video streaming is heavily driven by user gestures, e.g., clip action. Measurement results show that simple next-one strategy adopted by Douyin causes degraded user's view experience and extra resource usage cost.
- We adopt deep reinforcement learning to design an adaptive streaming policy for mobile short-form videos. External factors like download speed and user interactions are incorporated using extra convolutional layers. By considering multiple videos in the sliding window, LiveClip is also able to take advantage of information of videos in the playlist. Besides, different neural network models are trained to balance QoE and resource usage cost in distinct network scenarios.
- The proposed LiveClip framework is practical and only needs minor add-ons to the original short-form video system. The evaluation demonstrates that LiveClip outperforms existing state-of-the-art algorithms in terms of QoE and wastage cost by around 10%-40%.

The rest of this paper is organized as follows. Section 2 systematically discusses the measurement study and data analysis on Douyin app. Section 3 proposes our solution with deep reinforcement learning. Section 4 conducts evaluations and provide explanations of the results. Finally, Section 5 conclude our work.

2 A MEASUREMENT STUDY OF DOUYIN

While using the Douyin app, the videos will be displayed in full screen. The player has rather limited playback controls, i.e., videos to be played are not controlled by users. Users can only switch viewing videos by scrolling screen. In contrast, in traditional video sites, users can flexibly change video playback by explicitly clicking the link to the viewed videos. The scrolling logic of Douyin is that users cannot skip videos without scrolling screen.

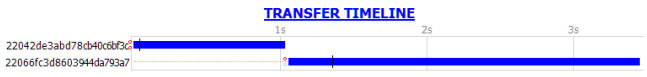


Figure 1: Timeline of delivery of consecutive videos.

Traditional mainstream video sites (e.g., Youtube) have adopted *DASH (Dynamic Adaptive Streaming over HTTP)* for video streaming. Douyin, however, adopts progressive download instead [12]. It is also based on HTTP but CDNs host the intact video files. Figure 1 shows the delivery timeline of consecutive videos with no user input, which is generated by Fiddler [2]. It is observed that the player always downloads the current video. When it is completed, the player starts to download the next video file. Previous connection is closed immediately when there is a scrolling event. We call such a fixed streaming scheme as the *Next-One* strategy.

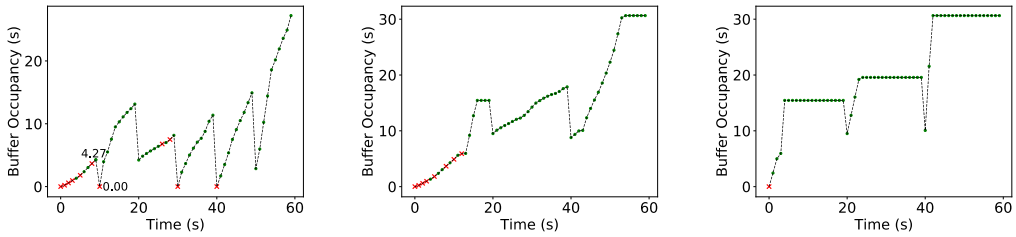
Figure 2 displays how buffer occupancy changes in one minute under different scrolling intervals and network conditions. The player keeps downloading video content, so the buffer level increases at the beginning. When a viewer scrolls the screen, video content is evicted from the buffer, so we can observe an obvious decline. When the player buffers enough data, the downloading process stops, which can be clearly observed in Figure 2(c). In the figure, a red cross indicates a rebuffering event and a green dot indicates a smooth playback. From the figure, we observe that longer delay and faster network speed decrease the number of rebuffering events. We divide the curve by declines into sections, and each one is a period when we stay focused on a particular video. The total wastage cost is defined as the difference of decline and the number of green dots in a section, or 0 if the difference is negative. For example, in Figure 2(a), the player gains smooth playback for only 4 seconds in the first section, and the decline after it is 4.27 seconds, so the discontinuity is 6 seconds while the wastage is 0.27 seconds.

By using Appium [1], we can emulate a real user's scrolling behavior and collect around 78,000 feed files with more than 520,000 video items during March 28th and April 25th in 2019. Figure 3 shows the statistics of videos collected in Douyin. Figure 3(a) shows the distribution of video length. During the period, normal users are only allowed to post videos up to 15 seconds in duration in Douyin, while advanced users are allowed to post videos up to a minute in duration. From Figure 3(a), we observe that more than half of all the videos (68.6%) are not more than 15 seconds. Figure 3(b) shows the distribution of video bitrate. The average video bitrate is 1190 Kbps. Figure 3(c) shows the distribution of video size. We fit the curve with exponentiated Weibull distribution. Figure 3(d) shows the distribution of timespan, namely, the number of days after the release of a video. Video timespan reveals popularity trends. The result shows that 33% of recommended videos are posted within a day, 72% are within a week, and 96% are within a month. In the first 10 days after the release of a video, the number of recommended videos falls dramatically as time elapses. The oldest video dates back to 400 days ago. From the results, we conclude that the recommender system tends to recommend recent videos, but older videos may still have small chance to be recommended.

Table 1: STATISTICAL PROPERTIES OF VIDEOS

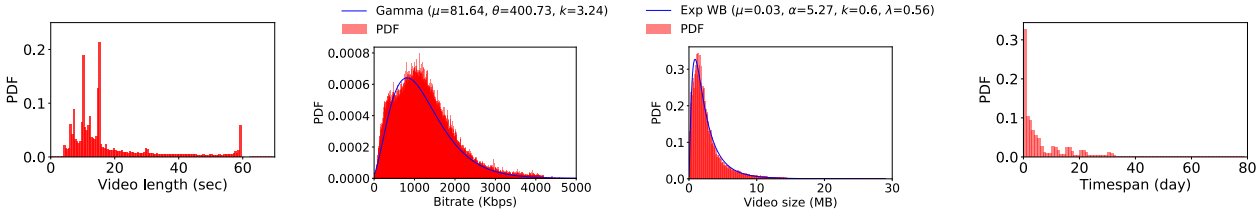
	Min	Max	Mean	Std
Length (sec)	0.13	153.37	18.64	14.11
Bitrate (Kbps)	1.53	4586.00	1216.60	702.93
Size (MB)	0.00	28.41	2.66	2.53
Timespan (day)	82.80 sec	400.28	6.80	10.33

In summary, Table 1 concludes the statistical properties of short-form videos collected from Douyin.



(a) 10 s interval, slow network. (b) 20 s interval, slow network. (c) 20 s interval, fast network.

Figure 2: The buffer occupancy changes in a minute with different user interaction patterns (indicated by interval of scrolling events) and network conditions (indicated by download speed). The red cross indicates a rebuffering event at that moment, and the green dot indicates a smooth playback.



(a) Distribution of video length. (b) Distribution of video bitrate. (c) Distribution of video size. (d) Distribution of video timespan.

Figure 3: Figure 3(a), 3(b) and 3(c) display the probability density of video length, bitrate and size respectively of all collected videos. Figure 3(d) presents the PDF of video timespan (how long the video has been released).

3 DESIGN OF LIVECLIP ALGORITHM

In this section, we introduce Markov Decision Process to better formulate our problem. Let T denote the total length of a viewing session. We use b_t, c_t, d_t to represent download speed, user interactions and player state at time t respectively. As a decision maker, the player needs to choose which video in the playlist to be downloaded at each time slot, denoted by T_{slot}. At each time step, the decision maker chooses an action a, transits to a new state s' from state s given the transition probability $P_a(s, s')$, and gains corresponding reward $R_a(s, s')$.

States. Let $S = \{s_1, s_2, \ldots, s_T\}$ denote the state space. T is the total length of a viewing session. $s_t = (b_t, c_t, d_t)$, where b_t, c_t, d_t represents download speed, user interactions and player state at time t respectively.

The player, download speed and user interactions all constitute the current state, which are introduced as follows.

(1) Player. As a decision maker, the player takes action with the guide of the policy. Given the observation from the measurement study in Section II, we follow some basic settings revealed by the measurement results.

We set time slot T_{slot} and video segment to be 1 second unit. Let v_t denote the video in the viewport at time t, and B_{v_t}, Q_{v_t}, M_{v_t} denote the buffered content, play progress and video length of the video in the viewport at time t. The player starts its playback only when the current segment is downloaded completely. The playback is described as follows:

$$Q_{v_{t+1}} = \begin{cases} Q_{v_t^{next}} & c_t = 1 \\ Q_{v_t} & B_{v_t} - Q_{v_t} < T_{slot} \\ & \text{and } B_{v_t} < M_{v_t} \text{ and } c_t = 0 \\ Q_{v_t} + 1 & \text{otherwise} \end{cases} \quad (1)$$

We assume a video is encoded with constant bitrate so that each segment with the same length is of the same size.

At each time slot the player can only request one video in the sliding window, so $Ef_t = \{\{v_t\}, \{v_t^{next}\}, \ldots, \varnothing\}$. When a user scrolls the screen, video content swiped away will be evicted immediately. The eviction set Ed_t is denoted as follows:

$$Ed_t = \begin{cases} \{v_t^{prev}\} & c_t = 1 \\ \varnothing & \text{otherwise} \end{cases} \quad (2)$$

The buffered content of swapped videos will be evicted immediately from the cache to reserve space for future videos. Given that time slot is 1 second, video playback stops when the next 1-second segment has not been completely buffered yet.

(2) Download speed and user interactions. In our work, we construct a 1-dimension CNN layer and take the download speed from the past k seconds as the input. CNN is used to learn the pattern of download speed. Similar to download speed, we build another 1D-CNN to learn the pattern of user behavior. It takes staying time of the past k videos as the input. It learns the pattern of historical user interactions and hopefully it predicts how soon a user will most probably scroll the screen.

(3) State space. To be more specific, we summarize all relevant factors that constitute the state space as follows.

- Download speed measurement of past k seconds.
- User staying time of past k videos.
- Play progress.
- Current staying time.
- Replay round.
- Bitrate of videos in the sliding window.
- Length of videos in the sliding window.
- Buffered content of videos in the sliding window.
- Time spent downloading videos in the sliding window.
- Videos completed in the sliding window.

The system state is related to download speed, user behavior and video player. Items 1 and 2 provide historical information of download speed and user behavior. Items 3-5 represent current playback state of the player, and items 6-10 are related to the videos in the sliding window, which are being watched or are to be watched in the next few seconds.

Actions. Let $A = \{a_1, a_2, \ldots, a_T\}$ denote the action space. Given a state, the agent needs to take action $a_t = (Ef_t, Ed_t)$, where Ef_t is the set of videos that are to be fetched while Ed_t is the set of videos that are to be evicted from the buffer at time t. Without loss of generality, swapped videos will be evicted from the buffer immediately.

At each time slot, the player needs to determine which videos in the sliding window should be downloaded. A time slot is defined as a minimal time unit, which is exclusive for one video transmission. The player can also decide to stop downloading at any time slot to avoid wastage cost. Action $a_t = (Ef_t, Ed_t)$ is defined as follows, where Ed_t depends on user behavior.

- Download current video. $Ef_t = \{v_t\}$.
- Download the next video. $Ef_t = \{v_t^{next}\}$.
- Download the second next video. $Ef_t = \{v_t^{after\_next}\}$.
- Pause. $Ef_t = \varnothing$.

State Transitions. The system state at time s_{t+1} is denoted as a tuple $(b_{t+1}, c_{t+1}, d_{t+1})$. The transition probability from state s_t to s_{t+1} given action a_t is as follows.

$$\begin{aligned} &P(S_{t+1}|s_t, a_t) \\ =&P((b_{t+1}, c_{t+1}, d_{t+1})|(b_t, c_t, d_t), a_t) \\ =&P(b_{t+1}|b_t)P(c_{t+1}|c_t)P(d_{t+1}|(b_t, c_t, d_t), a_t) \end{aligned} \tag{3}$$

The transition probability satisfies the Markov property. Here, $P(d_{t+1}|(b_t, c_t, d_t), a_t)$ is the probability given by the player model. It is difficult to build an accurate model based on partial knowledge of download speed and user behavior in a session.

Reward. The decision maker takes action a_t in state s_t, and transits to another state s_{t+1}, generating reward r_t. We view the reward here as the loss function in our optimization problem, which consists of QoE and wastage cost.

(1) QoE. We define QoE at time t as an indicator of rebuffering event $e_t \in \{0, 1\}$, which indicates whether the current video v_t stops its playback due to cache misses at time t. Rebuffering events greatly influence user experience. The player starts playing current video v_t only if the buffer of current video B_{v_t} is one segment (T_{slot}) ahead of play progress Q_{v_t} or has reached the total length

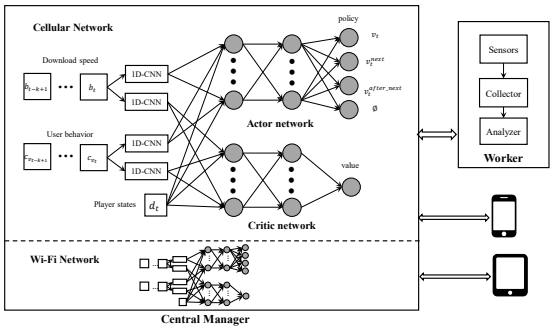


Figure 4: LiveClip framework.

M_{v_t}. Many studies like [13] adopt video quality as QoE. However, video quality in a short-form mobile video platform is determined when a playlist is generated. In addition, video quality remains unchanged during streaming a video because Douyin has adopted the progressive download strategy. Therefore, we safely neglect video quality here in QoE.

$$e_t = \begin{cases} 0, & B_{v_t} - Q_{v_t} \geq T_{slot} \text{ or } B_{v_t} = M_{v_t} \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

(2) Cost. Here, we only consider the unnecessary payment as the cost, which comes from the content that is buffered but has never been watched due to early scrolling events. Denote the cost at time t as w_t, we have

$$w_t = \sum_{i \in Ed_t, t>0}^{T} B_i - Q_i \tag{5}$$

The cost w_t is positive when $Ed_t \neq \varnothing$ or 0 otherwise. The cost is generated from scrolling events, indicating how many segments of the evicted video has not been watched.

(3) Loss function. Then, we combine both QoE and cost, and obtain the final loss function at time t, i.e.,

$$L_t = -e_t + \eta * w_t \tag{6}$$

where η is the weight that controls the trade-off between the importance of time spent on rebuffering events and time spent on downloading unused video content in a viewing session. In our experiment, η is a hyperparameter. It measures the importance of cost, which should be a large value when using cellular network and a small value when using Wi-Fi.

Our goal is to find a policy $\pi(a|s)$ that minimizes the total loss function in a session. $\pi(a|s)$ tells the agent the probability that an action will be chosen given the current state. The optimization problem is formally defined as:

$$\begin{aligned} \min \quad & L = \sum_{t=1}^{T} -e_t + \eta * w_t \\ \text{s.t.} \quad & (4)(5). \end{aligned}$$

In this paper, we use the state-of-the-art asynchronous Advantage Actor-Critic (A3C) method [11] to solve the formulated problem. Figure 4 shows the whole framework of LiveClip. Two general models are maintained by the central manager. They are trained with different reward functions. One of them is for the Wi-Fi scenario, and another is for the cellular network. A mobile device fetches both models from the central manager periodically, and determines which model to generate policies in real time based on the current network connection. When a user is viewing videos, a sensor process keeps monitoring download speed, user inputs and player statistics. The collector gathers traces in a session and the analyzer acts as a worker in A3C network.

4 EVALUATION

4.1 Experiment Settings

We use a real short-form video dataset (i.e., Douyin dataset) for our experiments. The collected dataset contains over 520,000 video items. We randomly select videos to form a complete playlist in each viewing session. For the network traces, we use the dataset generated by [10]. We use network traces of 22 hours for training and the left 8-hour trace for testing. As it is difficult to obtain a large real trace of user scrolling behaviors, we assume that intervals between scrolling events follow Gaussian distribution for simplicity. The detailed experiment settings are summarized in Table 2.

Table 2: Parameter Settings

Notation	Explanation	Value
T_{slot}	Duration of a time slot (sec)	1
η	weight of cost	0.1/1
α	Learning rate for actor network	0.00001
α'	Learning rate for critic network	0.00001
T	Number of steps in a session	720
γ	Discount factor	0.99
β	Entropy weight	0.1
δ	Global update iterations	40
k	Number of historical records	8

For each session, we randomly generate a synthetic list which contains a complete video playlist, a download speed trace and a user trace for simulation. Our simulator is able to calculate QoE and cost immediately whenever the agent takes action.

4.2 Algorithms

We compare LiveClip with the following benchmark algorithms, each representing a policy for streaming.

(1) **Optimal**. Given knowledge of current download speed and user staying time of the video in the viewport, the optimal method is the best policy for the player to gain the most rewards. It is regarded as the upper bound.
(2) **Next-One**, which is adopted by Douyin app. There is a sliding window with stop-and-wait protocol to indicate which videos to request. The player will not download the next video until current video finishes its delivery. And it will stay idle until a new video enters the viewport.
(3) **Waterfall**, which is an extension of the Next-One strategy, except that it is able to see the next 2 videos.
(4) **Buffer-Based (BB)**, which prefers to prefetch multiple videos in the sliding window. It is inspired by the idea of using only the buffer to determine video bitrate [9]. Here it chooses a byte range of some video file in the sliding window.
(5) **Random**, which randomly selects an action. This method provides the lower bound in QoE.

4.3 Results

4.3.1 Different download speed. We divide the download speed traces for test set into fast (>1700 Kbps), medium (1100-1700 Kbps) and slow (<1100 Kbps) groups based on different download speed patterns. The upper bound and lower bound are chosen based on the bitrate distribution of Douyin videos.

We simulate Wi-Fi network with moderate user behavior pattern. Figure 5(a) presents the total loss under different download speed patterns. Noted that nQoE is the negative of QoE. LiveClip is the best among all knowledge-free algorithms. Specifically, LiveClip outperforms other knowledge-free algorithms by 40%, 25% and 10% in fast, medium, slow download speed respectively. Specifically, LiveClip has much less cost compared to Next-One and Waterfall.

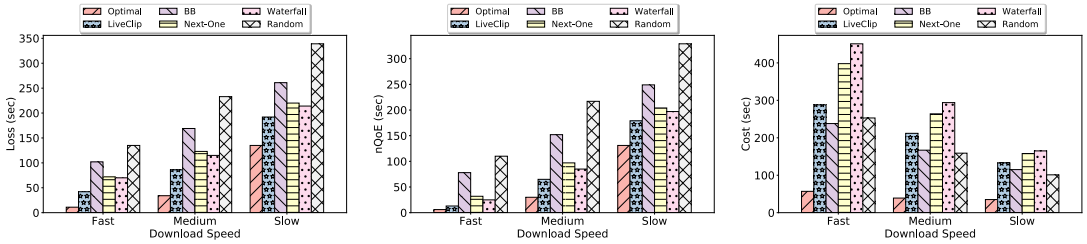
4.3.2 Different user behavior patterns. We assume that intervals between scrolling events in a session follow the Gaussian distribution. We set the same coefficient of variation to keep the same dispersion of distribution. μ and σ represent the mean and standard deviation of Gaussian distribution respectively. μ is chosen based on the measurement study presented in Section 2. Specifically, we group different user behavior patterns into 3 genres: frequent (μ=6, σ=3), moderate (μ=12, σ=6) and patient (μ=18, σ=9).

The evaluation is conducted in the simulation of Wi-Fi network with medium download speed. Figure 6(a) presents the total loss under different user behavior patterns. As shown in the figure, LiveClip has significant decrease in terms of total loss, which outperforms other knowledge-free methods by 17%, 26%, 16% in fast, moderate and slow behavior type respectively.

4.3.3 Different network connections. We consider three different scenarios of network connection: (1)*Vanilla-WiFi*, where $\eta = 0.1$ in loss function. (2) *Vanilla-Cellular*, where $\eta = 1$ in loss function. (3) *Alternate*, where access point changes to the other every 30 seconds. Figure 7 shows the total loss with different network connections. The buffer-based method focuses on the limited usage of resources, while the waterfall method values playback smoothness more than resource overuse. That's why the waterfall method performs better than BB in the Wi-Fi case and vice versa in the cellular case. But in Alternate case, they are similar with each other. However, LiveClip outperforms other knowledge-free methods in all cases.

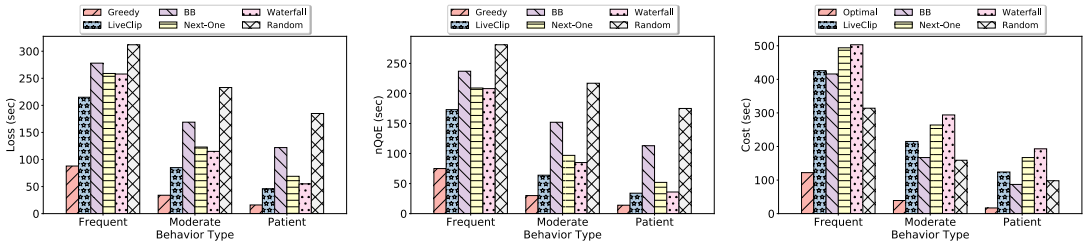
5 CONCLUSION

We conducted a large-scale measurement study on Douyin and obtained quite a few interesting findings. We also proposed an adaptive streaming strategy called *LiveClip*, which is based on the reinforcement learning framework. The proposed method can lead to better QoE with low extra resource usage cost. Evaluation results have shown that LiveClip outperforms other existing methods



(a) loss (b) nQoE (c) cost

Figure 5: Different network conditions.



(a) loss (b) nQoE (c) cost

Figure 6: Different user behavior patterns.

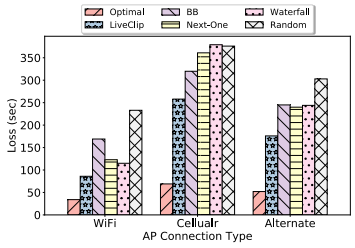


Figure 7: Different AP-based scenarios.

under different scenarios. Our future work will focus on designing a personalized caching system on an edge server.

ACKNOWLEDGEMENT

This work was supported by the National Key R&D Program of China under Grant 2018YFB0204100, Guangdong Special Support Program under Grant 2017TX04X148, and the National Natural Science Foundation of China under Grant 61802452.

REFERENCES

[1] 2019. Appium: Mobile App Automation Made Awesome. http://appium.io/
[2] 2019. Fiddler - Free Web Debugging Proxy. https://www.telerik.com/fiddler
[3] Zhuang Chen, Qian He, Zhifei Mao, Hwei-Ming Chung, and Sabita Maharjan. 2019. A Study on the Characteristics of Douyin Short Videos and Implications for Edge Caching. *arXiv preprint arXiv:1903.12399* (2019).
[4] Cisco. 2019. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022 White Paper. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html
[5] CNBC. 2018. The biggest trend in Chinese social media is dying, and another has already taken its place. https://www.cnbc.com/2018/09/19/short-video-apps-like-douyin-tiktok-are-dominating-chinese-screens.html
[6] Wenjie Hu and Guohong Cao. 2015. Energy-aware video streaming on smartphones. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 1185–1193.
[7] Wen Hu, Yichao Jin, Yonggang Wen, Zhi Wang, and Lifeng Sun. 2017. Towards Wi-Fi AP-Assisted Content Prefetching for On-Demand TV Series: A Reinforcement Learning Approach. *arXiv preprint arXiv:1703.03530* (2017).
[8] 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 *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 1208–1216.
[9] 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 *Proceedings of the 2014 ACM conference on SIGCOMM*. 187–198.
[10] 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.
[11] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. 1928–1937.
[12] Rubem Pereira and Ella Grishikashvili Pereira. 2014. Dynamic adaptive streaming over http and progressive download: Comparative considerations. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 905–909.
[13] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 325–338.
[14] Lei Zhang, Feng Wang, and Jiangchuan Liu. 2018. Mobile instant video clip sharing with screen scrolling: Measurement and enhancement. *IEEE Transactions on Multimedia* 20, 8 (2018), 2022–2034.
[15] Yuanxing Zhang, Pengyu Zhao, Kaigui Bian, Yunxin Liu, Lingyang Song, and Xiaoming Li. 2019. DRL360: 360-degree Video Streaming with Deep Reinforcement Learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1252–1260.