Preference-Aware Dynamic Bitrate Adaptation for Mobile Short-Form Video Feed Streaming

DEZHI RAN\textsuperscript{1}, (Student Member, IEEE), HUADUN HONG\textsuperscript{1}, YANG CHEN\textsuperscript{2}, BONAN MA\textsuperscript{3}, YUANXING ZHANG\textsuperscript{\textregistered1}, (Member, IEEE), PENGYU ZHAO\textsuperscript{1}, (Member, IEEE), AND KAIGUI BIAN\textsuperscript{1}, (Senior Member, IEEE)

\textsuperscript{1}Peking University, Beijing 100871, China
\textsuperscript{2}China Development Bank, Beijing 100031, China
\textsuperscript{3}Beijing University of Chemical Technology, Beijing 100029, China
Corresponding author: Dezhi Ran (dezhiran@pku.edu.cn)

ABSTRACT Short-form video feed becomes increasingly popular among younger generations, where mobile users watch videos of a few seconds one-by-one in a session or a pre-defined list. The common solution to improve the quality of experience (QoE) for short-video feed is to implement dynamic adaptive streaming strategies to decide bitrate for each video. However, legacy bitrate adaptation strategies fail to differentiate the weights of videos in the list according to the user preferences, leading to the bandwidth waste on downloading those video content skipped immediately by the users without being displayed. In this paper, we propose RecDASH, which consists of an attention-based user modeling module leveraging advanced recommendation algorithms and a reinforcement learning (RL) based bitrate adaptation module. Specifically, we use gated recurrent unit (GRU) network with attention mechanism to encode the session into a representation vector. Then, RL module combines the representation of session and other observations within playback, and yields the appropriate bitrate for the next short-form video for optimizing a given QoE objective. The low space and time complexity of the model enables it to be easily deployed on mobile devices. Trace-driven emulations verify the efficiency of RecDASH compared to several state-of-the-art streaming strategies with at least 5\%-15\% improvement on the video quality under various QoE objectives.

INDEX TERMS Bitrate adaptation, short-video feed, list-based recommendation, reinforcement learning.

I. INTRODUCTION

According statistics from Cisco [1], 75\% of all mobile data will be consumed by videos by 2020. Given abundant bandwidth of access networks today, communications using video contents has never been easier over feed-based mobile social platforms (e.g., Facebook and Instagram). Compared to the long-form video, the short-form video is more popular for attention capturing (especially for young generation)—to capture the user attention right away and consolidate the message within a few seconds in the video. Many video service providers have launched short-form video services such as TikTok, Vine, and Lasso, which allow users to upload and share self-filmed video clips of a few seconds.

In most of short-form video feed applications, there are usually many scenarios where users follow the predefined video list, rather than search the videos one-by-one, such as (1) a billboard chart ordered by the popularity of the videos; (2) a list of videos from specific creator sorted by the released time; (3) the list-based recommendation to avoid overloading of servers from the highly-frequent requests. It cannot be guaranteed that all short-form videos attract users when they are displayed to the users. Users have two options in the short-form video feed applications: (1) watch the entire short video if interesting, or (2) quickly skip the videos in the list against their interests. Apparently, requesting high bitrate for those videos skipped by users leads to a waste of bandwidth. Hence, to improve the quality of experience(QoE) for users under the limited and dynamic bandwidth resources, video service providers should assign preference-aware bitrates to the videos in the list for a specific user along with the playback.

The past few years have witnessed the success of long-form video bitrate adaptation strategies, guiding the clients to request an appropriate bitrate for future video chunks based on the playback statistics. However, these bitrate adaptation
strategies may not work well on streaming short-form video feed due to the following challenges.

- **Differentiated weights of short videos.** The long-form video bitrate adaptation strategies equally treat the chunks of the entire video. However, it is not lucrative to request high bitrate for the short-form videos that may probably be skipped by users immediately, or the bandwidth will be wasted on downloading content without being displayed.

- **Balance between efficiency and accuracy.** Since the short-form video services are mainly aimed for mobile devices, it is unwise to use complex model pursuing little elevation in the accuracy of recommendation and bitrate allocation. We must balance between the efficiency and accuracy, using models with low space and time complexity and yielding a satisfactory performance.

- **QoE of short-form video feed.** Strategies of streaming long-form video have reached a consensus on maximizing bitrate, minimizing rebuffering and maximizing smoothness. However, the optimal streaming strategies could not guarantee the quality of specific video chunks, while the QoE should assign rewards to user-preferred short-form videos with high quality.

To achieve better user experience, it is essential for short-form video’s recommendation module to deliver videos that user prefers. With frequent user behavior towards different videos, session-based recommendation for short-form videos appears adequate and easy at first glance. However, due to the short length of each video, the recommendation module generates a video list that user may favor based on user’s feedback on previous video lists (same as the list-wise recommendation in the news-feed scenario [2]), instead of reacting instantly to deliver one single video for next. This means videos in the current playing video list are fixed. When the recommendation is not accurate or more likely, the user preference migrates during the playback of the list, some videos in the current list but not played can be identified as unattractive. These disliked videos have to be played since the video list is fixed, but allocating high bitrate to them is an obvious waste of bandwidth.

Our research findings indicate that recent advances in recommender system and reinforcement-learning (RL) based streaming strategies hold the promise of addressing these challenges. In the literature, many methods have been proposed for capturing users’ interest and click-through-rate (CTR) prediction, which model the users and items (i.e., short videos in this paper) into embedding vectors and predict whether a certain group of users would click (or fully watch) the items in the list with the impressions (i.e., displays). Though the order of the short-form video feed list is usually generated by the state-of-the-art prediction models, few works focus on integrating the intermediate or final recommendation results to the bitrate allocation scheme. Meanwhile, due to the various preferences on QoE for videos among users, the RL based streaming strategies are proposed to adaptively learn the optimal policies for maximizing the given QoE objective in any form. Priority attempt is presented to ensure the quality of annotated video chunks with cascaded RL design. In the short-form video feed streaming task, we are expected to dynamically allocate appropriate bitrates to the short-form videos in the list, along with the prediction of whether the users would fully watch the next short-form video.

In this paper, we propose RecDASH, a pull-based video streaming strategy which leverages advanced recommendation algorithm for preference prediction and adaptively requests bitrate for the next short-form video. RecDASH applies the attention mechanism over a gated recurrent unit (GRU) network to identify the user preferences based on the previous records of the current session and yields a session-level representation to model the user behavior. A bitrate adaptation module is then employed to differentiate the value of each video to the user and decide the bitrate to be requested for the next video over the observation of the playback and the session-level representation. The RL strategy, i.e., Q Actor-Critic, is proposed in bitrate adaptation, which adaptively optimizes the given objective with various weights on QoE metrics. We construct a dataset by the real-world records in the Avazu CTR prediction dataset and the public bandwidth traces from HSDPA to examine the performance of RecDASH. Compared to several state-of-the-art bitrate adaptation strategies for long-form videos, RecDASH could improve the video quality by at least 5%-15%. Extensive analysis reveals that RecDASH does improve the quality of viewing the short-form videos for the interests of users, indicating that RecDASH is practical to be implemented in the real-world short-form video feed applications.

We summarize the contribution of this paper as follows:

- We propose a pull-based video streaming strategy for mobile short-form video feed with state-of-the-art recommendation technique and RL model, which adaptively determines bitrate for the next short-form video based on the records in the current session.
- The proposed RecDASH can adapt to QoE objectives with various weights of the QoE metrics in the short-form video feed by integrating the session-level representation learned by the attention-based user behavior modeling method.
- We analyze the complexity of RecDASH, and find that RecDASH can be executed on the off-the-shelf mobile devices in real-time with periodically updating the parameters of the model.
- Trace-driven emulation confirms the performance of RecDASH compared to several state-of-the-art bitrate adaptation strategies (strategies originally designed for long-form videos).

## II. RELATED WORK

### A. ADAPTIVE VIDEO STREAMING

Adaptive video streaming is a new trend to optimize QoE for users in video services. During the past few years, researchers...
have agreed on the standard of Dynamic Adaptive Streaming over HTTP (DASH) [3] to make bitrate adaptation decision based on the playback statistics of the long-form videos. In general, the common bitrate adaptation strategy attempts to optimize the QoE by maximizing the quality of video chunks, minimizing rebuffering duration and maximizing smoothness between the consecutive video chunks [4]–[7]. These strategies can be divided into two categories of QoE-sensitive and QoE-insensitive by whether the strategies perform differently given the QoE objective in various formats. For example, buffer-based (BB) rate adaptation [8] is a typical QoE-insensitive strategy to avoid rebuffering based on the mobile devices’ playback buffer condition. Regarding the QoE-sensitive video streaming strategies, theoretical control, such as MPC [5], convex optimization, such as BOLA [6], and RL techniques, such as Pensieve [7], are three most common methods which can work well in specific scenarios. Recently deep learning methods [9]–[11] have also been leveraged to improve the performance of adaptive video streaming. Beyond that, much efforts have been made on implementing collaborative information to facilitate video streaming, e.g., CS2P [12] which focuses on improving the wireless network bandwidth estimation for bitrate adaptation, SDNDASH [13] which introduces central control to coordinate the usage of the limited bandwidth resources. Although the previous bitrate adaptation strategies attempt to optimize the overall QoE of the playback, they are not appropriate for the short-form video feed as they cannot guarantee the high quality of specific short-form videos for interests of users. This paper presents a task-aware QoE objective for short-form video feed, and proposes a pull-based strategy with RL model for the optimization.

B. SESSION-BASED RECOMMENDATION OF VIDEOS

Recommender systems are one of the most successful and widespread applications of machine learning technologies in business, creating a delightful user experience while driving incremental revenue. In terms of the online services such as video watching, the session-based recommendation is essential to predict the user preferences only with the records in the session. Many researchers transform the sessions into sequences and capture the user interests within the sequences by recurrent neural network such as GRU4Rec [14]. Furthermore, some works introduce attention mechanism [15] and self-attention mechanism [16] to enhance the learning capability of the recurrent models. Compared to the recommending common items, recommender systems of online videos can retrieve both video-relevant features and collaborative features [17], [18] for accurate recommendation. Usually, content providers would deploy list-wise recommendation for feed-like content [2] such as videos. In this case, click-through rate (CTR) and fully-view-rate are two practical indicators that help companies decide which short-form videos should be collected in the list and in what order the videos should be transferred to user’s mobile devices [19]. However, most of these recommender systems should be executed on the server side due to the complex models. RecDASH learns the session-level representation via a GRU network with attention mechanism to predict whether a user will fully view a short-form video on the mobile device side, considering both accuracy and efficiency.

III. DESIGN OF RecDASH

A. SHORT-FORM VIDEO FEED STREAMING

In the scenario of short-form video feed streaming, users watch the short-form videos one-by-one following a predefined or selected watch list \(v = (v_1, v_2, \ldots, v_N)\), where each short-form video \(v_i\) is composed of a series of frames with length \(l_i\) and \(N\) is the length of the watch list. The watch lists are generated by the recommender system or selected by the users themselves. The lists are fixed throughout the watching processes unless reaching the end, and then users can choose to leave or watch the short-form videos in another list. During the playback, users can watch the entire short-form videos they like or quickly skip the videos against their interests.

**FIGURE 1.** Collaboration between bitrate adaptation and recommendation module.

Fig. 1 provides an illustration for the collaboration between bitrate adaptation module and recommendation module inside client. When the current video list is about to finish, for example, 8 out of 10 videos in the list have been watched, the recommendation module generates list-wise video recommendation, e.g., the next video list to be forwarded to bitrate adaptation module. The bitrate adaptation module takes user preference information and bandwidth condition together into consideration to provide client with decisions on the bitrate allocation for videos which haven’t been downloaded yet.

Meanwhile, the client deployed on the mobile device repeatedly sends the requests for downloading the short-form videos in the watch list. Since each short-form video \(v_i\) may have multiple sizes corresponding to different bitrates, the client needs to determine a bitrate \(br_i \in BR\) for \(v_i\) from the candidate bitrate set \(BR\) before the request. The downloaded videos are stored in the playback buffer, and the buffer is consumed by the user playbacks. We will introduce the client in detail in the following sections.

1) SHORT-FORM VIDEO REQUEST

Different from the long-form video streaming that divides the videos into chunks (short segments of consecutive frames) or
tiles (spatial portions in a chunk), the client requests an entire video in a single request in the short-form video feed streaming setting. Suppose the client begins to send the request for downloading the $i$-th video $v_i$ of bitrate $br_i$ at timestamp $t_i$. Since the network condition is dynamic, we assume that the bandwidth varies with time and its value at timestamp $t$ is $N(t)$. Then, the download time $t_i^{d}$ of $v_i$ can be calculated by:

$$q_{ti, br_i} = \int_{t_i}^{t_i + t_i^{d}} N(t) \, dt,$$

(1)

where $q_{ti, br_i}$ denotes the quality (size) of video $i$ of $br_i$. Currently on short-form video platforms, videos are mostly home-made vlogs (short videos sharing daily life), which makes it feasible to use the video size to represent the video quality in short video streaming scenario. Because of this fact and the purpose of simplifying mathematical notations, we assume that the videos of the same length and bitrate have the same content size. Note that there may be a short delay $\Delta t_i$ between the $i$-th and $(i+1)$-th short-form videos. Hence, the relationship between $t_i$ and $t_{i+1}$ can be expressed by:

$$t_{i+1} = t_i + \Delta t_i + t_i^{d}.$$  

2) BUFFER UPDATE

Assume the size of the playback buffer is $b_f$, where the unit size is measured by one second of playback time. Define $B(t) \in [0, b_f]$ as the remaining playback time of the video content in the buffer, i.e., buffer occupancy, at timestamp $t$. The buffer occupancy decreases along with video playback, and increases by $l_i$ when the $i$-th short-form video has been downloaded completely.

Let $B_i = B(t_i)$ be the buffer occupancy when the request for $v_i$ is sent. To overcome the cold-start problem, the client first downloads $n$ short-form videos before the playback begins. After the warm-up stage, the buffer occupancy changes according to both the downloading and playback of the users. In addition, the client drops the remaining playback in the buffer when the users try to skip the current video. Therefore, the buffer occupancy at $t_{i+1}$ could be calculated by:

$$B_{i+1} = \begin{cases} \sum_{j=1}^{i-1} v_j & i \leq n, \\ \rho ((B_i - t_i^{d} - t_i) + l_i - \Delta t_i) & n < i \leq N, \end{cases}$$

(3)

where $t_i^{d}$ represents the total length of the user-skipped videos between $t_i$ and $t_{i+1}$, and $\rho(x) = \max(x, 0)$ which guarantees that buffer occupancy is no less than 0.

During the playback, there are two special events worth noting:

- **Rebuffering given empty buffer.** When the buffer occupancy reaches 0, the playback suffers from rebuffering, and the user needs to wait until the current short-form video is downloaded into the buffer. The rebuffering would hurt the user experiences.

- **Downloading suspended given full buffer.** When there is no buffer space for a new short-form video $v_i$, i.e., $B_i + l_i > b_f$, the downloading process should suspend for a short delay of $\Delta t_i$:

$$\Delta t_i = \rho \left( \rho ((B_i - t_i^{d} - t_i) + l_i - b_f) \right).$$

(4)

3) BITRATE ALLOCATION

When the $(i-1)$-th video has been completely downloaded into the buffer, the client allocates bitrate $br_i$ from the candidate bitrate set for the $i$-th video. The bitrates are usually generated by an algorithm based on the environmental state for optimizing several QoE objectives. Formally, given the environmental state $s_i$, including bandwidth, buffer occupancy, requested video length, etc., the allocated bitrate $br_i$ for $v_i$ is calculated by $br_i = f(s_i)$ where $f$ is the proposed allocation algorithm.

B. PREFERENCE-AWARE DYNAMIC BITRATE ALLOCATION

To adapt the changing environment, we follow the DASH (Dynamic Adaptive Streaming over HTTP) framework proposed for the long-form video streaming, and introduce RecDASH which adaptively allocates bitrates and optimizes the QoE objectives over the playback statistics during the short-form video feed streaming. Specifically, we utilize the reinforcement learning in RecDASH because it is inherently suitable for portraying the dynamic action-picking process (bitrate allocations) in the changing environment (video streaming) to optimize the accumulated rewards (QoE).

1) PREFERENCE-AWARE BITRATE ALLOCATION

In the short-form video feed streaming, the watch list is fixed for each user. However, users may not be interested in all the videos in the predefined watch list, and they will skip the shows against their interests. It is always beneficial to take user preferences into consideration during the bitrate allocation. For example, the client may allocate low bitrates to the short-form videos that users may not like. We show the analytic results of the actual watching time of the videos against user interests in this section.

In this paper, we propose a preference prediction module in RecDASH. Let $I_i$ denote the indicator function of whether the user is interested in $v_i$:

$$I_i = \begin{cases} 1 & \text{user is interested in } v_i, \\ 0 & \text{user is not interested in } v_i. \end{cases}$$

(5)

The preference prediction module predicts the user preference $I_i (1 \leq i \leq n)$ based on the historical watch list before $t_i$. Then, the user preference results, as well as the playback features are fed into RecDASH for dynamic bitrate allocation.

2) ANALYSIS OF USER WATCHING TIME

Intuitively, it is reasonable to allocate low bitrates to the short-form videos that users may not like. We show the analytic results of the actual watching time of the videos against user interests in this section.

Assume users watch the entire videos that meet their interests, while videos that do not attract users are only...
partially watched. Suppose the short-form videos in the watch list are all of the same length \( L \). Following the statistical analysis in [20], we assume that the watch time \( T \) of each short-form video fits the truncated exponential distribution, where the density function is:

\[
 f_T(t; \lambda) = \frac{\lambda e^{-\lambda t}}{\int_0^L \lambda e^{-\lambda t} dt} ,
\]

where \( 0 \leq t \leq L \), and \( \lambda \) is the rate parameter such that a larger \( \lambda \) indicates that the users tend to skip the videos in the early stage. Then, we have the following proposition:

**Proposition 1:** Assume the user prefers \( kN \) short-form videos out of \( N \), where \( k \in (0, 1) \). Then, the expected playback time against the user interests accounts for less than \( \frac{1-k}{N} \) in the total playback.

**Proof:** The expectation of watching time \( T \) can be calculated by:

\[
 \mathbb{E}(T) = \int_0^L t f_T(t; \lambda) dt = \int_0^L \frac{\lambda t e^{-\lambda t}}{\lambda e^{-\lambda t}} dt = 1 - \frac{1}{\lambda} \int_0^L e^{-\lambda t} dt = 1 - \frac{e^{-\lambda L} - 1}{\lambda}.
\]

Since the user only prefers \( kN \) videos in the watch list, the proportion of the playback time that against user interests is at most:

\[
 \frac{(1-k)N}{kNL + \frac{(1-k)N}{\lambda}} = \frac{1-k}{\lambda kL + 1-k} .
\]

Hence, the proposition is proved.

In the real-world scenario, \( \lambda \) is usually large. Hence, we could infer that the actual playback which is against user interests is very short such that we could neglect the effect of low bitrate allocation on those videos.

**C. QUALITY OF EXPERIENCE METRICS**

The quality of experience (QoE) is a measure of the delight or annoyance of users’ experiences with the applications, which is the optimization objective of the short-form video feed streaming system. There are many kinds of QoE components under different scenarios. However, those QoE measurements are mainly designed for long videos being split into consecutive chunks, whereas do not measure users’ satisfaction indicated by the viewing duration of short videos. In this paper, we focus on the following metrics that significantly affect the user experiences for video \( v_i \) (1 \( \leq i \leq N \)):

1) Effective Video Quality. According to the numerical analysis in the previous section, we only consider the quality of the effective videos, i.e., videos in the watch list that conform to the user preferences, in the watch list. The effective video quality could be computed as:

\[
 QoE_i^e = I_i \log q(i, br_i) .
\]

2) Rebuffering. Since the rebuffering greatly hurts the user experience, we also take the rebuffering time into consideration:

\[
 QoE_i^r = \rho(t_i^d + t_i^s - B_i) .
\]

3) Bitrate Fluctuation. During the video playback, the frequent bitrate changes (fluctuation) have a negative impact on the user experience. Hence, it is necessary to minimize the fluctuation, which can be measured by:

\[
 QoE_i^f = |br_i - br_{i-1}| ,
\]

where specially we assume \( br_0 = br_1 \).

Then, the QoE objective of \( v_i \) could be formulated by:

\[
 QoE_i = \mu_1 QoE_i^e - \mu_2 QoE_i^r - \mu_3 QoE_i^f ,
\]

where \( \mu = (\mu_1, \mu_2, \mu_3) \) are non-negative weighting parameters. Note that different settings of \( \mu \) result in different optimization targets. We follow [5], [7] to use the additive form of QoE objectives which can adapt to different user preferences flexibly. For example, if users have a strong preference for high-quality videos, \( \mu_1 \) should be set relatively larger than \( \mu_2 \) and \( \mu_3 \). On the other hand, if users do not tolerate delays but can bear low bitrates of the videos, it would be better to set \( \mu_2 \) larger than the other QoEs.

To provide high QoE for users, we propose a preference-aware dynamic bitrate allocation mechanism named RecDASH, which determines the bitrates in real time and maximizes the overall QoE objective:

**Problem 1:**

\[
 \arg\max_{\{br_1, br_2, ..., br_N\}} \sum_{i=1}^N QoE_i \\
 \text{subject to Constraints (1)-(4)},
\]

where the constraints (1)-(4) correspond to Eqn.(1)-(4).

**IV. PREFERENCE-AWARE DYNAMIC BITRATE ALLOCATION**

RecDASH model consists of two major modules, as illustrated in Fig. 2. The Preference Prediction Module (Part I)
predicts the user preference over the watch list, while the Bitrate Adaptation Module (Part II) performs dynamic bitrate allocation with deep reinforcement learning based on the monitored statistics in the playback environment. The target of RecDASH is to maximize users’ quality of experience.

A. PREFERENCE PREDICTION WITH GATED RECURRENT UNIT

Let \( v = (v_1, v_2, \ldots, v_N) \) be the watch list. Given a prefix of watch list \( v_1 = (v_1, v_2, \ldots, v_i) \), 1 \( \leq i \leq N \), the preference prediction model outputs the probability \( I'_i \) that user would be interested in short-form video \( v_i \). In the short-form video feed streaming, we assume that the complete watching of the videos signifies the preferences of the users, while skipping behaviors indicates that the short-form videos are against user interests. The network architecture of preference prediction module is shown in Fig. 3.

\[ g_i = (1 - z_i)g_{i-1} + z_i\hat{g}_i, \quad (12) \]

where the update gate \( z_i \) is given by

\[ z_i = \sigma(W_c x_i + U_z g_{i-1}). \quad (13) \]

The candidate activation function \( \hat{g}_i \) is computed as:

\[ \hat{g}_i = \tanh(W_r x_i + U(re_l \odot g_{i-1})), \quad (14) \]

where the reset gate \( r_e \) is given by:

\[ r_e = \sigma(W_r x_i + U_r g_{i-1}). \quad (15) \]

We simply use \( g_i \) as the GRU-based representation for \( v_i \).

2) ATTENTIVE SEQUENCE ENCODER

To capture the latest information in the watch list, we also involve the attention mechanism [25], which allows the network to dynamically select and combine different parts of the input sequence, and generate attentive representation \( c_i \) of sequence \( v_i \):

\[ c_i = \sum_{j=1}^{i-1} \alpha_{ij} g_j, \quad (16) \]

where \( \alpha_{ij} \) is the normalized weighted factor that determines which part of the activation sequence should be emphasized or ignored in the prediction,

\[ \alpha_{ij} = \text{softmax}(v_i^T \tanh(W_1 g_j + W_2 c_i)). \quad (17) \]

3) USER PREFERENCE DECODER

We utilize both GRU-based representation and attentive representation for the user preference prediction \( E(I'_i) \). Specifically,

\[ E(I'_i) = \sigma(v_i^T (W_g g_i + W_c c_i)). \quad (18) \]

The preference prediction model is trained on cross entropy loss between \( I_i \) and \( E(I'_i) \).

B. BITRATE ADAPTATION MODULE

During playback, before the request of video \( v_i \), the bitrate adaptation module determines the bitrate \( b_{r_i} \) of \( v_i \) from the playback statistics. To improve the overall QoE of users, we utilize reinforcement learning (RL) to conduct and optimize dynamic bitrate allocation.

1) BITRATE ALLOCATION WITH REINFORCEMENT LEARNING

In the standard RL settings, an agent interacts with an environment \( \mathcal{E} \) over a number of discrete time steps. At each timestep \( t_i \), the agent receives an observation \( o_i \), takes an action \( a_i \) according to its policy \( \pi \), and receives a scalar reward \( r_i \). The discounted return \( R_i = \sum_{j=i}^{N} \gamma^{j-i} r_j \) is the total accumulated reward from time step \( t_i \) with discount factor \( \gamma \in (0, 1] \). The goal of reinforcement learning is to maximize the expected return by optimizing the policy.

Apparently, the bitrate allocation can be catered to the RL framework. Specifically, the client requesting the playback can be seen as the agent, and everything else can be regarded as the environment. When the client is about to allocate...
bitrate for video \( v_i \), it collects the playback statistics as the observation \( o_i \). The client updates the previous state \( s_{i-1} \) with \( o_i \) and gets the current state \( s_i \). Then, it chooses a bitrate (action) \( a_i = br_i \) from discrete bitrate set \( BR \) based on the learnable allocation policy \( \pi_\theta(a_i|s_i) \) (parameterized by \( \theta \)), and receives the user QoE as the reward \( r_i = Qoe_i \). The goal of the bitrate allocation here is to learn the optimal policy \( \pi^*_\theta \) that maximizes the expectation of the total discounted QoE:

\[
\pi^*_\theta = \arg \max_{\pi_\theta} \mathbb{E}_{a_i \sim \pi_\theta} Q^\pi(a_i, s_i), \tag{19}
\]

where

\[
Q^\pi(a_i, s_i) = \mathbb{E}_{R_i}[R_i]
\]

is the action-value function, i.e., the expected return under from state \( s_i \) and action \( a_i \), under policy \( \pi_\theta \). Note that the Eqn. (19) is a transformation of Problem 1. Following the framework of reinforcement learning.

From theoretical proof in [26], the optimal policy in RL is deterministic. Thus, after training the model with RL algorithms, i.e., Q Actor-Critic [27] in RecDASH, the client only needs to pick the bitrate with the highest probability during playback.

2) BITRATE ALLOCATION MODEL

In RecDASH, the client collects video id \( i \), video buffer state \( B_i \), and video length \( l_i \) from the playback. The client also records the bandwidth of last 10 seconds before \( t_i \), i.e., \( N_i = (N(t_i - 9), N(t_i - 8), \ldots, N(t_i)) \), to capture the dynamic information, where \( N(t_i) \) denotes the bandwidth at timestamp \( t_i \). For the consideration of playback fluctuation, the client should also be aware of the previous bitrate allocation, i.e., \( br_{i-1} \). Besides, to give preference-aware allocation, the preference prediction \( l'_i \) as well as the output of the GRU encoder \( g_i \) and attentive encoder \( c_i \) are also added to the observation. Thus, the observation \( o_i \) can be described as the following feature collection:

\[
o_i = \text{concat}(i, B_i, l_i, N_i, br_{i-1}, l'_i, g_i, c_i). \tag{20}
\]

To combine the historical observations, we utilize another GRU \( H_\theta \). The state \( s_i \) can then be calculated by:

\[
s_i = H_\theta(s_{i-1}, o_i), \tag{21}
\]

where \( s_0 \) is set to zero. Note that the observations are the consequences of previous allocations, so the actions are implicitly included in the state.

We conduct the bitrate allocation based on the policy \( \pi_\theta(a_i|s_i) \). In concrete, \( \pi_\theta \) utilizes a linear project transformation and a softmax layer to convert state \( s_i \) to the discrete action distribution. Besides, as RecDASH uses the Q Actor-Critic algorithm to optimize the model, an extra Q-network (realized by a fully-connected layer) is applied on the state to generate the estimation of the action-value function \( Q_w(s_i, a_i) \) for each bitrate \( a_i \).

3) Q ACTOR-CRITIC BASED MODEL TRAINING

RecDASH exploits Q Actor-Critic algorithm [27] to train the bitrate adaptation module for optimizing the QoE objectives. The Q Actor-Critic algorithm is fully online and incremental, which can be updated during playback.

Specifically, at each iteration \( i \), the client receives the sequence of \( o_i, s_i, a_i, r_i, o_{i+1}, s_{i+1} \), and computes the Q-target:

\[
y_i = r_i + \gamma \max_{a_i} Q_w(s_{i+1}, a_i'). \tag{22}
\]

Then, the gradient of \( w \) is calculated by minimizing the L2 loss between \( y_i \) and \( Q_w(s_i, a_i) \):

\[
dw = \nabla_w (y_i - Q_w(s_i, a_i))^2. \tag{23}
\]

Note that the Q-target \( y_i \) and \( Q_w(s_i, a_i) \) are both the estimations of action-value function, where Q-target is a more precise estimation due to the one-step look ahead. Thus, the Q-network will be more close to the actual action-value function after parameter update.

After that, the client calculates the Q-error:

\[
\delta_i = r_i + \gamma \max_{a_i} Q_w(s_{i+1}, a_i') - Q_w(s_i, a_i), \tag{24}
\]

which is an estimation of the advantage. Based on \( \delta_i \), the gradient of \( \theta \) can be deduced by:

\[
d\theta = \nabla_\theta \pi_\theta(s_i, a_i)\delta_i. \tag{25}
\]

It is worth mentioning that the Q-error is equal to the gradient of the L2 loss between \( y_i \) and \( Q_w(s_i, a_i) \) on Q function, while the update of \( \theta \) is equal to the gradient ascent on the expected value of \( Q^\pi(s_i, a_i) \). Besides, GRU \( H_\theta \) is also updated at the same time when \( Q_w(s_i, a_i) \) and \( \pi_\theta(s_i, a_i) \) are updated.

The whole training methodology of RecDASH is shown in Algorithm 1. It first trains the parameters of preference prediction module based on the user behaviors (line 2). Afterward, the model repeatedly simulates the video playback events and optimizes the bitrate allocation model (line 3-17). The parameters are trained based on the Q Actor-Critic algorithm and eventually, return the optimal policy \( \pi^*_\theta \).

4) COMPLEXITY ANALYSIS

Denote \( L_s \) as the sequence length. Denote \( I \) as the input size of GRU-based RNN, e.g., the embedding size of the video feature. And denote \( H, A_L \), as the hidden size of GRU and attention vector length, respectively.

1) Space Complexity. Note one GRU has three gates. The size of the attention parameters is \( N_{att} = H \times A_L \). So space complexity of the preference prediction module can be calculated as:

\[
N_{pp} = 3 \times H(H + I + 1) + N_{att} = O(H^2 + HI + H). \tag{26}
\]

The space complexity of the RL model consists of the parameters of Actor-Critic Network and the GRU. Similarly denote \( H' \) as the hidden size of GRU in bitrate
Algorithm 1 Training Method of RecDASH

Require: Training data, $D$; Factor discounting future reward, $\gamma$. Learning rate of policy, $\alpha^\theta$; Learning rate of action-value function, $\alpha^w$.

Ensure: Parameters of preference prediction module, GRU $H_o$, policy $\pi^\theta(s, a)$ and Q-network $Q_w(s, a)$; $\xi$, $\eta$, $\theta$, $w$;

1: Initialize parameters $\xi$, $\eta$, $\theta$ and $w$;
2: Train $\xi$ on Training Data $D$;
3: repeat
4:   for each watch list of length $N$ in Training Data $D$ do
5:     Initialize $s_0$
6:     for $i \in [1, N - 1]$ do
7:       Get $l_i$, $c_i$, $g_i$ from preference prediction module;
8:       Sample $a_i \sim \pi^\theta$;
9:       Get $o_{i+1}$ and $r_i$ from environment;
10:      Get $s_{i+1}$ from $H(s_i, o_{i+1})$;
11:      $\delta_i \leftarrow r_i + \gamma \max_{a'} Q_w(s_{i+1}, a') - Q_w(s_i, a_i)$;
12:      $\theta \leftarrow \theta + \alpha^\theta \delta_i \nabla_{\theta} \pi^\theta(s_i, a_i)$;
13:      $w \leftarrow w + \alpha^w \delta_i \nabla_w Q_w(s_i, a_i)$;
14:      Update $\eta$;
15:   end for
16: end for
17: until Reward do not increase.
18: return $\xi$, $\eta$, $\theta$, $w$.

6) CLIENT

We embed the preference prediction module and RL-based bitrate adaptation module of RecDASH into a client application by PyTorch\(^1\) which can be easily deployed on mobile devices. At the beginning of playback, the client downloads a configuration file which describes the information of the current watch list. During the playback, the client collects the statistics in the playback, and dynamically allocates bitrate via both bitrate adaptation module and preference prediction module at each time step. The selected bitrate is injected into the request for the next video, which is then inserted to the request queue of the download module. The download module sends the requests one-by-one. The downloaded videos are fed into the video decoder and then stored into the video buffer waiting for access from the video player.

7) EMULATION PLATFORM

An emulation platform could not only evaluate the performance of specific streaming strategies, but also help RL module conduct training on the provided dataset. Given a bandwidth trace and a sequence of short videos with watching duration, the whole playback procedures can be simulated in a fast manner. In this case, we develop an emulation platform by Python, with RecDASH developed by PyTorch. We could then monitor the detailed statistics of the playback, which could both help evaluate performance and guide the training.

---

\(^1\)https://pytorch.org/mobile/android/
of RecDASH. When the training is finished, the model would be dumped to the mobile devices, and executed in the real-world short-video streaming applications.

VI. EVALUATION
In this section, we evaluate the performance of RecDASH over several real-world scenarios and compare it with some state-of-the-art algorithms.

A. SETTINGS
1) DATASET GENERATION
We randomly select watch lists and bandwidth traces from public datasets and combine them as the final dataset for the evaluation.

- **Videos.** We collect ten movies encoded in 4K, and randomly segment them into 2000 short-form videos of 10 to 20 seconds length. We assume that there are five candidate bitrates in total. Thus, we encode these short-form videos by FFmpeg with X.264 under the average bitrate (ABR) mode, where each video is assigned by 2Mbps (360p), 8Mbps (720p), 17Mbps (1080p), 31Mbps (2K) and 50Mbps (4K).

- **Watch lists.** We select 3000 watch lists of length 20 from the real-world Avazu CTR prediction dataset.2 We pair the IDs in the CTR dataset with the randomly generated short-form videos, and thus each short-form video is assigned with an unique ID. "0" in the dataset represents that the user is not interested in the short-form video, and we assume that the user will skip this video following the truncated exponential distribution. "1" in the dataset represents that the user is interested in the short-form video, and will fully watch that video. We plot the distribution of the proportion of user interested videos within the single watch list in Fig. 5, where we find that most of the user (nearly 90%) would not watch more than 60% of the entire short videos. For the evaluation, 10% of watch lists are selected as the test set, which are guaranteed not to appear in the training procedure.

- **Bandwidth traces.** We choose 100 bandwidth traces with various bandwidth patterns from public datasets [28], [29] to simulate different network bandwidth conditions. The average bandwidth (available bandwidth for short-form video streaming) ranges from 8Mbps to 20Mbps, and 80% of bandwidth falls between 10Mbps and 16Mbps. Similar to watch lists, we randomly select 10 bandwidth traces as the test set for evaluation.

2) IMPLEMENTATION DETAILS OF RecDASH
In the preference prediction module, the size of input embedding is set to 96, i.e., video id embedding of 32 units and the video features of 64 units, and the hidden size of GRU is set to 64. Different from preference prediction module, since the values of different features in bitrate allocation have various scales, we conduct feature scaling on all the features except the output of GRU encoder and attentive encoder. Thus, the size of each observation is 142 and we set the hidden size of GRU as 128 in bitrate allocation module. We utilize the Adam optimizer [30] to train both modules, where the additional linear decay and gradient clipping are also applied to prevent overfitting. The buffer capacity \( b_f \) is set to 30 seconds of playback time, which is a common setting in real-world applications. Besides, we download one short-form video in the buffer before playback to overcome the cold-start problem.

3) BASELINE METHODS
We propose five baseline models for comparison.

- **Bandwidth based strategy (BB):** This Bandwidth based strategy merely evaluates the bandwidth condition and allocates the highest bitrate based on the predicted bandwidth condition.

- **Popularity based strategy (POP):** POP supposes that the video preferred by most users in the previous playback traces is likely to be popular for other people in the future. Thus, it collects the CTR for each video and the videos with higher CTRs, i.e., within top 20%, will be allocated with a high bitrate, while the other videos will be allocated with a low bitrate.

- **Preference-based strategy (PB):** PB prefers more on the videos that have higher predicted user preference. Specifically, it allocates higher bitrate to the videos when the predicted preference is larger than 0.5, while the other videos are allocated with a low bitrate.

- **Smoothness based strategy (SB):** SB aims to avoid temporal fluctuations between adjacent videos by allocating similar bitrate with the previous video.

- **Pensieve:** Pensieve is originally designed to make bitrate adaptation for streaming long-form videos by RL model. In this paper, we modify it to be used in the short-form video scenario by treating each short-form video as a video chunk. Note that whether the user would fully-view the specific short-form video is not considered in Pensieve. It just takes bandwidth conditions and playback statistics into consideration, as conventional DASH streaming systems do.

---

2 https://www.kaggle.com/c/avazu-ctr-prediction/data

---

FIGURE 5. Distribution of the proportion of user interested short-form videos within traces in the dataset.
4) PERFORMANCE METRICS
We select three sets of parameters of $\mu$ to represent three QoE objectives with various preferences. Specifically, the QoE weights are $(2, 0.2, 0.1)$, $(1.8, 0.4, 0.1)$, $(2.7, 0.1, 0.45)$, indicating the preference of maximizing the video rates, minimizing temporal fluctuations over playback, and minimizing rebuffering time respectively. The QoE objectives of the compared streaming systems are normalized to $[0, 1]$ by the maximal value of QoE objective, which is calculated under the ideal bandwidth (always download 2K chunk without rebuffering).

B. PRECISION OF PREFERENCE PREDICTION
First, we examine the performance of the prediction model for user preference. We check the Receiver Operating Characteristic (ROC) for each method to evaluate the prediction accuracy. As shown in Fig. 6, our proposed preference prediction model exhibits better performance over the POP algorithm (i.e., recommending the most popular short-form videos to users), indicating that the proposed algorithm is qualified for this task.

C. INTEGRATION OF RECOMMENDATION WITH RL
In this section, we investigate whether the designed RL structure in RecDASH can take full advantage of the preference prediction with recurrent neural network. To analyze the significance of the preference prediction module, we compare the performance of Pensieve and RecDASH. Further, to show the effective of the user preference feature, we drop the input of GRU encoder and attentive encoder in the observations of RecDASH and yield another compared algorithm, i.e., 01-DASH. The comparison between the three methods is plotted in Fig. 7 under three QoE objectives. The results reveal that the preference prediction greatly improves the QoE (i.e., 01-DASH and RecDASH). Moreover, since RecDASH could appropriately integrate the user preference features in the state, it receives higher QoE than the 01-DASH.

D. COMPARISON WITH EXISTING APPROACHES
We then compare the proposed RecDASH model with the baseline and other state-of-the-art algorithms on all QoE objectives in the settings. Fig. 8 shows the detailed CDF results under three QoE objectives for all the algorithms. It is obvious that RecDASH outperforms the compared algorithms on all QoE metrics over the test traces, and obtains about 10%-20% improvement in average.

In particular, compared to the state-of-the-art streaming systems originally designed for long-form videos, RecDASH improves the average video quality by 5%-15% when seeking to maximize the average quality in the QoE objective, i.e., $(2, 0.2, 0.1)$. This improvement implies that RecDASH is aware of the QoE objective and thus allocates high bitrates to the preferred videos and lower bitrates to the unpreferred ones. When considering most on the bitrate temporal variations in the QoE objective, i.e., $(1.8, 0.4, 0.1)$, RecDASH harmonizes the priorities among the three metrics. In addition, RecDASH suffers from less rebuffering time than the compared approaches when focusing more on the rebuffering time in QoE objective, i.e., $(2.7, 0.1, 0.45)$. Beyond that, we also examine the distribution of the allocated bitrates for the preferred videos, and depict the result in Fig. 9. We can conclude from the result that RecDASH is capable of allocating high bitrate to the preferred short-form videos compared to both the preference-aware methods (i.e., PB and POP) and the preference-unaware methods (i.e., BB, SB, and Pensieve).

E. DISCUSSION OF RecDASH
We then dive into the analysis of why RecDASH works well on the short-form video feed streaming task. Specifically, there are three main observations from the above experiments. First, the RecDASH improves the quality of the user-interested videos much more than the videos against user interests. Since the users would not watch too many videos in the given list (as shown in Fig. 5), the RecDASH can thus allocates high bitrates only to the user-interested short-form videos and results in high QoEs. Second, since it is obvious that different QoE objectives require inherently different streaming strategies, the algorithms of SB, BB, and POP which employ fixed control strategies struggle to optimize different QoE objectives. Therefore, the RL-based methods, e.g., the proposed RecDASH, could adapt to any QoE objectives, and thus outperforms the other methods no
matter how the objective changes. Third, although Pensieve and PB algorithms have the capability to adapt to the various QoE objectives, they do not consider the preference of the users such that they can not utilize the bandwidth resources properly compared with the proposed RecDASH. Based on the above analysis, our RecDASH system integrates the user preference in the RL-based bitrate allocation, which could overcome the problems in the other compared methods.

VII. CONCLUSION

In this paper, we present an RL-based preference-aware short-form video feed streaming system with a focus on not only efficiently utilizing the limited bandwidth resources to improve the QoE of users but also making full use of the playback history by recommendation. The proposed RecDASH system has a video-content independent model, which can dynamically change features in the environments, and thus is capable for optimizing various QoE objectives. The model leverages GRU-based RNN with the attention mechanism to predict the user’s preference to the video, and RL-based bitrate adaptation module to dynamic allocates the bitrates based on the generated preference predictions for short-form videos. Trace-driven evaluations show that the proposed RecDASH outperforms state-of-the-art algorithms over three QoE metrics, indicating the importance of combining users’ preference and dynamic bitrate allocation in short-form video feed.

REFERENCES

[1] C. V. N. Index, “Cisco visual networking index: Forecast and methodology, 2016–2021,” Complete Vis. Netw. Index (VNI) Forecast, vol. 12, no. 1, pp. 749–759, 2017.
[2] Y. Shi, M. Larson, and A. Hanjalic, “List-wise learning to rank with matrix factorization for collaborative filtering,” in Proc. 4th ACM Conf. Recommender Syst., 2010, pp. 269–272.
[3] T. Stockhammer, “Dynamic adaptive streaming over HTTP-: Standards and design principles,” in Proc. 2nd Annu. ACM Conf. Multimedia Syst., Feb. 2011, pp. 133–144.
[4] M. Li and C.-Y. Lee, “A cost-effective and real-time QoE evaluation method for multimedia streaming services,” Telecommun. Syst., vol. 59, no. 3, pp. 317–327, Jul. 2015.
[5] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over HTTP,” ACM SIGCOMM Comput. Commun. Rev., vol. 45, pp. 325–338, Aug. 2015.
[6] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, “Bola: Near-optimal bitrate adaptation for online videos,” in Proc. 35th Annu. ACM Conf. Comput. Commun., May 2016, pp. 1–9.
[7] H. Mao, R. Netravali, and M. Alizadeh, “Neural adaptive video streaming with pensieve,” in Proc. Conf. ACM Special Interest Group Data Commun., Aug. 2017, pp. 197–210.
[8] T.-Y. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, “Comyco: Quality-aware adaptive video streaming via imitation learning,” in Proc. 27th ACM Int. Conf. Multimedia, Oct. 2019, p. 429.
