Quality-aware Neural Adaptive Video Streaming with Lifelong Imitation Learning

Tianchi Huang[1], Chao Zhou[2*], Xin Yao[1], Rui-Xiao Zhang[1], Chenglei Wu[1], Bing Yu[2], Lifeng Sun[3*]

Abstract*—Off-the-shelf Adaptive Bitrate (ABR) algorithms, no matter model-based or learning-based, pick future chunk's bitrates via fixed rules or offline trained models to improve quality of experience (QoE). Nevertheless, data analysis demonstrates that the ABR algorithm is required to update continually for adapting itself to time-varying network conditions. To this end, we propose Comyco, a video quality-aware learning-based ABR approach that enormously improves recent schemes by tackling the above challenge as follows: *1) Comyco is quality-aware that picks the chunk with higher perceptual video qualities rather than video bitrates; 2) Comyco's inner-loop system trains the policy via imitating expert trajectories given by the instant solver, which can not only avoid redundant exploration but also make better use of the collected samples; 3) Comyco's outer-loop system employs the lifelong learning method to continually train the model w.r.t the fresh trace collected by the users.

Consequently, quality-driven Comyco learns policies solely without any presumptions, which can fast adapt to current network distributions. To achieve this, we develop a complete quality-aware lifelong imitation learning-based ABR system, including implementing the inner-loop and outer-loop subsystem, constructing quality-based neural network architecture, collecting video datasets as well as estimating QoE metrics with video quality features. Using trace-driven and real-world experiments, we demonstrate Comyco's sample efficiency has 1700$\times$ improvements in the number of samples required and 16$\times$ improvements in training time required, compared with the prior work. Meanwhile, Comyco outperforms previously proposed methods, with the improvements on the average quality of experience (QoE) of 7.5%-16.79%. Moreover, experimental results on continual training also illustrate that lifelong learning help Comyco further improve the average QoE of 1.07%-9.81% in comparison to the offline trained model.

***Index Terms*—Imitation Learning, Quality-aware, Lifelong Learning, Adaptive Video Streaming.**

I. Introduction

Recent years have witnessed a tremendous increase in the requirements of watching online videos [1]. Adaptive bitrate (ABR) streaming, the method that dynamically controls the video player to download different bitrate video for the next chunk, has become a leading scheme to deliver video streaming services with high quality of experience (QoE) to the users [2] (e.g., ABR technologies are widely used by Youtube [3], Hulu [4], Youku [5], and iQiyi [6]). Recent model-based ABR approaches (§X) pick the next chunk's video bitrate via only current network status [7], [8], or buffer occupancy [9], [10], or joint consideration of these two factors[11], [12]. However, such heuristic methods are usually set up with presumptions, that fail to work well under unexpected network conditions [13]. Thus, learning-based ABR methods adopt reinforcement learning (RL) [13], [14], [15] or self-learning method [16] to *learn* the strategies without any presumptions, and finally, outperforming traditional model-based approaches [17].

While previous work has demonstrated considerable QoE improvement in a different manner, in this study, we attempt to understand whether current ABR methods have already been satisfied with nowadays' network scenarios (§II). We, therefore, collect a large corpus of network traces (Kwai dataset) on the leading video streaming platform Kuaishou [18] (§II-B). The analysis shows that 1) more than 80% of network traces require adaptive streaming to ensure high QoE, as state-of-the-art model-based ABR approach MPC lacks the performance on over 60% of sessions. 2) learning-based ABR approach (i.e., Pensieve [13]) often trains the neural network (NN) under the past network scenario. However, the network distribution has changed dramatically during the time of training convergence. As a result, such pre-trained models can hardly provide comparable performances under the current network condition. 3) as much as the overall network condition shows different throughput distribution at large time intervals, it changes slowly and smoothly with time. Hence, learning-based ABR algorithms should be updated effectively and efficiently for smoothing the vibration of network conditions. To achieve this goal, we summarize the challenges from three perspectives (§III):

- ✓ *How to implement a quality-aware ABR system?* The majority of existing ABR approaches [11], [13], [17] place less importance on the video quality information, while perceptual video quality is a non-trivial feature for evaluating QoE (§VI-A,[19]). Consequently, even though these schemes have achieved higher QoE objectives, they may generate the strategy diverging from the actual demand (§III-B).
- ✓ *How to empowering the training efficiency for learning-based ABR algorithms?* Recent Reinforcement Learning (RL)-based ABR schemes [13], [14] lack the efficiency of both collected and exploited expert samples, which leads to the inefficient training [20].
- ✓ *How to achieve continual learning for the ABR system?* NN-based ABR methods should be incrementally updated with *fresh* network traces, and in the meanwhile, the selected traces should be less but critical enough to represent bandwidth distributions of the current network.

* Corresponding Author, mailto: htc19@mails.tsinghua.edu.cn.
[1] Media Lab, Tsinghua University.
[2] Kuaishou, Beijing, China.
Manuscript received July 21, 2019; revised Oct. 26, 2019.

We find an opportunity to address the last two issues in real-world network environments by leveraging the concept of lifelong imitation learning. On the one hand, imitation learning enables the ABR system to achieve fast training. On the other hand, a lifelong learning method allows the NN to continually integrate the evolution in network distributions into the passing time. Meanwhile, quality-aware learning-based ABR algorithm is still challenging since the state-of-the-art learning-based scheme [13], [16] lacks almost all the modules of constructing a quality-aware ABR system, that includes, viable neural network models, feasible and high-efficiency training methodologies, dedicated video datasets based on video quality metrics, as well as video quality-based QoE methods.

Following this insight, we propose *Comyco*, a novel video quality-aware lifelong imitation learning-based ABR system, aiming to remarkably improve the overall performance of ABR algorithms via tackling the above challenges. Different from previous work [13], Comyco is mainly composed of the inner-loop system and the outer-loop system and is equipped with the following properties (§III-D).

▷ *Comyco aims to select bitrate with high perceptual video quality rather than high video bitrate.* Ideally, constructing a quality-aware ABR system is quite challenging since we have neither reliable metrics nor video and network datasets. To achieve this goal, we first integrate the information of video contents, network status, and video playback states into the Comyco's NN for bitrate selection (§IV-A). Next, we use VMAF [21], an objective full-reference perceptual video quality metric, to measure the video quality. Meanwhile, we also propose a linear combination of a video quality-based QoE metric that achieves the state-of-art performance on Waterloo Streaming SQoE-III [22] dataset (§VI-A). Finally, we collect a DASH-video dataset with various types of videos, including movies, sports, TV-shows, games, news, and music videos (MV) (§VI-B).

▷ *Comyco utilizes the inner-loop system (§IV), which leverages imitation learning [23] for training the neural network (NN).* Since the near-optimal policy can be precisely and instantly estimated via the current state in the ABR scenario, the collected expert policies can enable the NN for fast learning. Specifically, the agent is allowed to *explore* the environment and *learn* the policy via the expert policies given by the solver. Especially, we propose *instant solver* (§IV-B) to estimate the expert action with a faithful *virtual player* (§VII-A4). Furthermore, we utilize *experience replay buffer* (§IV-D) to store expert policies and to train the NN via the specific loss function L_{comyco} (§IV-C).

▷ *Comyco uses the outer-loop system (§V) to achieve continual learning.* We consider the process of continuous adaptation to network status as a lifelong learning process. The key idea is to filter out the useful traces collected from the client, and periodically update the NN via the inner-loop system and learn the strategies using Learning without Forgetting (LwF) [24] method. To achieve this, we implement *optimal estimator* (§V-A) to estimate the normalized QoE for each collected trace at first. We then store the specific trace (the normalized QoE of which is below a threshold) into the *trace collector* (§V-B). Finally, the inner-loop system is periodically enabled to update the NN w.r.t LwF method (§V-D).

Furthermore, we evaluate the inner-loop system via both trace-driven and real-world experiments. From trace-driven emulation (§VII-A2), we find that Comyco significantly accelerates the training process, with 1700x improvements in terms of a number of samples required compared to recent work (§VII-B). Comparing with existing schemes, Comyco outperforms them under various network conditions (§VII-A3) and videos (§VI-B), with the improvements on average QoE of 7.5% - 16.79%. In particular, Comyco performs better than state-of-the-art learning-based approach Pensieve, with the improvements on the average video quality of 7.37% under the same rebuffering time. Further, we present results that highlight Comyco's performance with different hyper-parameters and settings (§VII-D). Finally, we validate Comyco in real-world network scenarios (§VII-E). Extensive results indicate the superiority of Comyco over existing state-of-the-art approaches.

In the rest of the paper, we also validate the performance of Comyco using the outer-loop system, namely lifelong Comyco (§VIII). Experimental results demonstrate that lifelong Comyco improves the average QoE of 1.07% - 9.81% compared with the offline trained model (§VIII-B). Meanwhile, we further analyze the performance of lifelong Comyco with a model-based ABR scheme, and we find that lifelong Comyco is a better scheme, and it can automatically adapt to the frustration of network conditions. Finally, the comparison between lifelong Comyco and the online optimal policy of Comyco illustrates that lifelong Comyco has almost achieved the near-optimal performance (§VIII-D).

Contribution. We summarize the contributions as follows:

▷ Using data-driven analysis, we identify the short-comings of today's ABR schemes and propose Comyco, a video quality-aware lifelong learning-based ABR system, that significantly ameliorates the weakness of the learning-based ABR schemes from three perspectives (§III).

▷ Unlike prior work, Comyco picks the video chunk with high perceptual video quality instead of high video bitrate. Experiments results also demonstrate the superiority of our proposed algorithm (§III-A,§III-D).

▷ To the best of our knowledge, we are the first to leverage imitation learning to accelerate the training process for ABR tasks. Results show that exploring imitation learning can not only achieve fast convergence rates but also improve performance (§IV,§VII).

▷ We consider the ABR's continuous updating task as a lifelong learning process. Results demonstrate the effectiveness of adopting lifelong learning to continuously train the NN for adapting unstable network conditions (§V,§VIII).

II. Background and Motivation

In this section, we begin by introducing ABR's background. Then we analysis today's ABR services. Finally, we highlight the limitations of strawman solutions for ABR schemes without lifelong learning and present key insights that lead to implementing a new ABR system for providing better QoE to the users at any time.

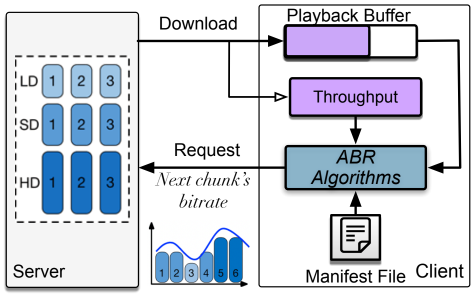


Fig. 1. Adaptive Bitrate (ABR) Streaming System Overview.

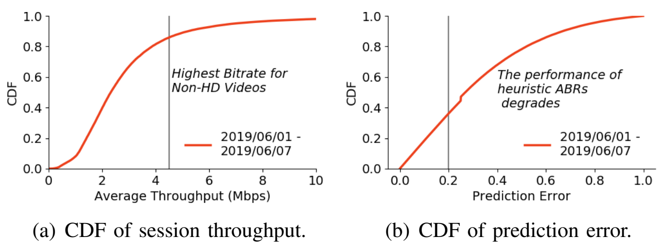


Fig. 2. An overview of Kwai dataset, including the distribution of average throughput and prediction error using popular throughput prediction method [8].

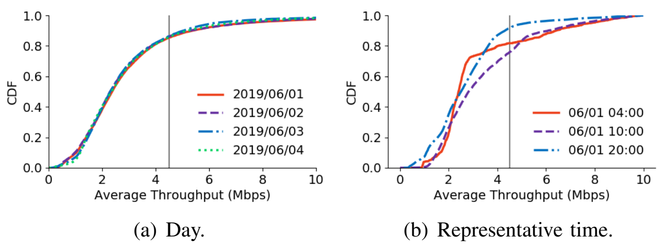


Fig. 3. CDF of session throughput by day and representative time.

A. ABR Overview

Due to the rapid development of network services, watching videos online has already become a common trend. Today, the predominant form for video delivery is adaptive video streaming, such as HLS (HTTP Live Streaming) [25] and DASH [26], which is a method that dynamically selects video bitrates according to network conditions and clients' buffer occupancy. As shown in Figure 1, the traditional video streaming framework consists of a video player client with a constrained buffer length and an HTTP-Server or Content Delivery Network (CDN) [8]. The video player client decodes and renders video frames from the playback buffer. Once the streaming service starts, the client fetches the video chunk from the HTTP Server or CDN in order by an ABR algorithm. Meanwhile, the algorithm, deployed on the client-side, determines the next chunk N and next chunk video quality Q_N via throughput estimation and current buffer utilization. The goal of the ABR algorithm is to provide the video chunk with high qualities and avoid stalling or rebuffering events [2].

B. Analysis for Today's ABR Services

Our work is started by a realistic problem: *with the rapid improvements of today's network bandwidth, are ABR algorithms still necessary for video streaming services to provide better QoE to the users?*

To answer this question, we require a *fresh* throughput dataset in large-scale, continuous throughput measurements, and long session duration. However, revisiting previously proposed public throughput trace datasets [27], [28], [17], we observe that such existing datasets lack either the diversity of throughput traces or the continuous measurement through the entire weeks, which finally unable to use them directly for research purpose [1]. To this end, we collect a large-scale network bandwidth dataset, namely Kwai, from the video streaming viewers of Kuaishou [18]. Kuaishou is a leading video streaming platform in China that has over 1000 million users worldwide. Millions of original videos are published on it every day. In detail, The dataset consists of over 86 thousand traces from 9,941 users, 7 days in total (1104 hours in terms of overall bandwidth time recorded.) from various network conditions collected in June 2019. Then we utilize the Kwai dataset to implement several experiments for answering the questions above and dedicate several observations.

[1]It's notable that CS2P's network dataset [29] is fit for our work, but it still has not been published yet.

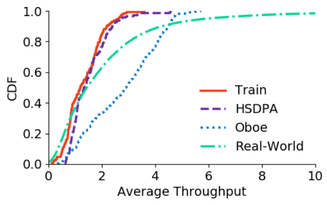
$\mathfrak{Observation}$1. *Experiments illustrate that ABR algorithms are still necessary for 80% of today's network conditions. Meanwhile, the state-of-the-art heuristic method MPC [11] only performs well under almost 40% of all sessions.*

First, to better investigate the importance of utilizing ABR algorithms in the Kwai dataset, we compute average throughput on all the sessions and report them as the CDF distribution plot in Figure 2. Figure 2(a) shows, assuming that the highest video bitrate is 4.3Mbps [2], we find that over 80% of sessions require ABR algorithms to adjust next chunk's bitrate for avoiding rebuffering events. Moreover, recent work [11] demonstrate that MPC's performance heavily depends on the throughput accuracy, when the prediction error is under 20%, the normalized QoE of MPC is close to optimal (over 85%). Thus, we also illustrate the CDF distribution of prediction error in Figure 2(b), where the prediction error is computed as *harmonic mean method*. The method is the default setting of the traditional MPC algorithm. Surprisingly, over 60% of sessions gain a large prediction error (over 20%). To this end, such observations prove that we need *learning-based ABR algorithm* rather than fixed-rule-heuristics for achieving better QoE [9], [11].

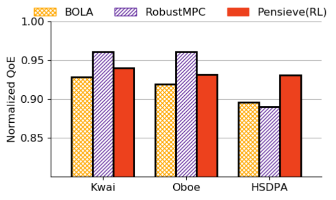
$\mathfrak{Observation}$2. *Measurements show the network distribution will be different if the time gap lasts over 6 hours. However, the training time of recent learning-based ABR algorithms is in the range of 4-23 hours [13], [14], [16]. Such learned strategy (trained on previous network distributions) may perform poorly on current network conditions.*

Note that recent client-based ABRs often trained [13] or designed [11], [30] once and deployed on the users' client

[2]It's a standard-setting for Non-HD (1080p) videos [13], [17]

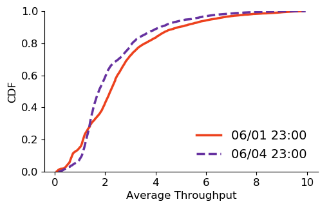


(a) CDF of session throughput on different network traces

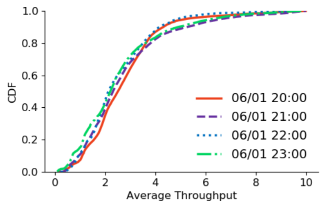


(b) Normalized QoE on the selected network environments

Fig. 4. Comparing the throughput distribution on different network datasets. We also report normalized QoE on each network dataset.



(a) Day with a designated time.



(b) Continuous hours.

Fig. 5. CDF of session throughput by day and continuous time.

without any further changes. Such observation leads to another critical question: *can ABRs tame the complexity of dynamic network conditions without updating?* We, therefore, evaluate the performance of existing ABR algorithms (i.e., Pensieve, RobustMPC and BOLA (§VII-A4) under different network conditions, including HSDPA, Oboe (§VII-A3) and Kwai on the virtual player (§VII-A2). For the sake of fairness, we train Pensieve on the training set and validate it on the various network traces with the same model. Results are demonstrated in Figure 4(b). We show that 1) the overall performance of learning-based ABR algorithms heavily rely on the similarity of the throughput distribution between the training set and the validation network environments, and 2) no scheme can outperform others under all considered networks. Unsurprisingly, Pensieve rivals or outperforms other schemes under the HSDPA network dataset but performs worse than the other under Kwai and Oboe network conditions. It makes sense since Pensieve seems to overfit over the training set, in which the throughput distribution of HSDPA is similar to the training dataset (See more in Figure 4(a)).

Next, we start to thoroughly understand the throughput distribution of the Kwai dataset in terms of *the time dimension*. Figure 3(a) reports the CDF of the average throughput from June 1 to June 4. As shown, there's no obvious difference between the throughput distribution on each day. What's more, we compute the throughput distribution with three representative times on the same day, where the time is 4:00, 10:00, and 20:00, which means the sleeping time, working time, and resting time, respectively. Results are shown in Figure 3(b), and we can see that the network distribution is strongly correlated with human behavior. For example, the average throughput in the night (20:00) is lower than that in the morning (10:00). Previously proposed RL-based method Pensieve will take over 4 hours to train a reliable strategy. Thus, although the training time of Pensieve is rather short compared with other NN-based tasks, we believe that the current strategy, i.e., training a NN from scratch in 4 hours, doesn't always satisfy the users' requirements.

𝔒𝔟𝔰𝔢𝔯𝔳𝔞𝔱𝔦𝔬𝔫3. *ABR algorithms is allowed to be generated or updated efficiently within short duration since the bandwidth distribution evolves slowly over time, as the distribution of two adjacent time points (1 hour) is approximately the same.*

Figure 5 shows the CDF of throughput distribution from another perspective, and we can see that throughput distribution of two adjacent time points (1 hour) is approximately the same. To this end, the intuitive idea is to continually train the model with a short period, which allows the ABR algorithm to update dynamically instead of leveraging fixed parameters or rules. In our work, we set the training period as 1 hour. In detail, we take 55 minutes for collecting network traces and use only 5 minutes for training the NN (§V-D).

Summary. In general, exploring the aforementioned opportunities, however, requires not only achieving outperformed performances with fast efficiency training in a short period but also continually learning to fit the variety of network environments.

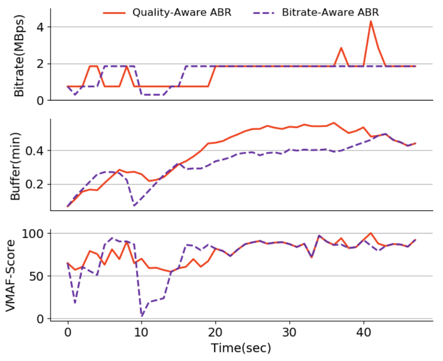
III. Challenges and Key Ideas

In this section, on the basis of the aforementioned observations, we mainly generalize three key challenges from several perspectives, that includes, training NN via imitation learning (§III-B), how to develop a complete video quality-based ABR system (§III-A), and how to deploy the system for lifelong updating. Motivated by the key challenges above, we then propose Comyco, a video quality-aware ABR scheme with lifelong learning.

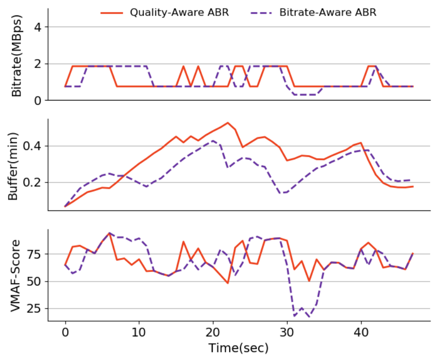
Most traditional ABR algorithms [8], [11], [10] leverage time-series prediction or automation control method to make decisions for the next chunk. Nevertheless, such methods are built in pre-assumptions that it is hard to keep its performance in all considered network scenarios [13]. To this end, learning-based ABR algorithms [13], [14], [16] are proposed to solve the problem from another perspective: it adopts deep reinforcement learning (DRL) to train a neural network (NN) from scratch towards the better QoE objective. Despite the outstanding results that recent work has obtained, learning-based ABR methods suffer from several key issues:

A. Challenges for Perceptual Video Quality-aware ABRs

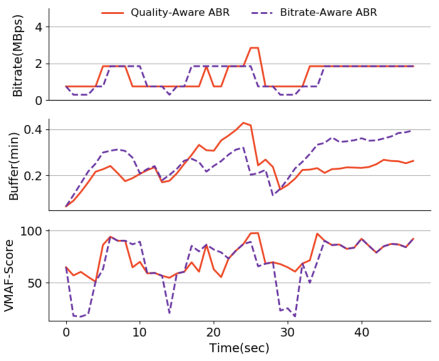
What's more, previous learning-based ABR schemes [11], [13] are evaluated by typical QoE objectives that use the combination of video bitrates, rebuffering times and video smoothness. However, such QoE metrics are short-handed because these forms of parameters neglect the quality of video presentations [31]. Meanwhile, recent work [32], [33] has found that perceptual video quality features play a vital part in evaluating the performance of VBR-encoded ABR streaming services. To prove this, we plot the trajectory generated by the quality-aware ABR and bitrate-aware algorithm on



(a) Trace No.**11** in HSDPA Dataset [28]

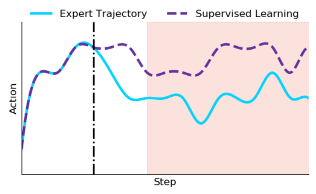


(b) Trace No.**115** in Oboe Dataset [17]

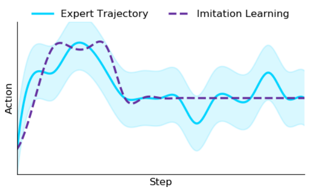


(c) Trace No.**24** in FCC Dataset [17]

Fig. 6. We evaluate quality-aware ABR algorithm and bitrate-aware ABR algorithm with the same video on Norway network traces respectively. Results are plotted as the curves of selected bitrate, buffer occupancy and the selected chunk's VMAF (§VI-A,[21]) for entire sessions.



(a) Supervised learning



(b) Imitation learning

Fig. 7. The real trajectory on the ABR task given by imitation learning and supervised learning, where the red background means the player occurs the rebuffering event.

Figure 6. As shown, the bitrate-aware algorithm selects the video chunk with higher bitrate but neglects the corresponding video quality, resulting in a large fluctuation in the perceptual video qualities. What's more, bitrate-aware algorithm often wastes the buffer on achieving a slight increase in video quality, which may cause unnecessary stalling event. On the contrast, the quality-aware algorithm picks the chunk with high and stable perceptual video quality and preserves the buffer occupancy within an allowable range. To this end, one of the better solutions is to add video bitrates as another metric to describe the perceptual video quality. We, therefore, encounter the second challenge of our work: *How to construct a video quality-aware ABR system?*

Our Solution. Our next challenge is to set up a video quality-aware ABR system. The work is generally composed of three tasks: 1) We construct Comyco's NN architecture with jointly considering several underlying metrics, i.e, past network features and video content features as well as video playback features (§IV-A). 2) We propose a quality-based QoE metric (§VI-A). 3) We collect a video quality DASH dataset which includes various types of videos (§VI-B).

B. Challenges for Learning-based ABRs

Recent learning-based ABR schemes often adopt RL methods to maximize the average QoE objectives. During the training, the agent rollouts a trajectory and updates the NN with policy gradients. However, the effect of calculated gradients heavily depends on the amount and quality of collected experiences. In most cases, the collected samples seldom stand for the optimal policy of the corresponding states, which leads to a long time to converge to the sub-optimal policy [23], [34]. Thus, we are facing the first challenge: *Considering the characteristic of ABR tasks, can we precisely estimate the optimal direction of gradients to guide the model for better updating?*

Our solution. Recall that the key principle of RL-based method is to maximize *reward* of each *action* taken by the agent in given *states* per step, since the agent doesn't really know the optimal strategy [35]. However, recent work [11], [13], [12], [17], [36], [16] has demonstrated that the ABR process can be precisely emulated by an offline virtual player (§VII-A2) with complete future network information. What's more, by taking several steps ahead, we can further accurately *estimate* the near-optimal expert policy of any ABR state within an acceptable time (§IV-B). To this end, the intuitive idea is to leverage supervised learning methods to minimize the loss between the predicted and the expert policy. Nevertheless, it's impractical because the off-policy method [35] suffers from *compounding error* when the algorithm executes its policy, leading it to drift to new and unexpected states [37]. For example, as shown in Figure 7[a], in the beginning, supervised learning-based ABR algorithm fetches the bitrate that is consistent with the expert policy, but when it selects a bitrate with a minor error (after the black line), the state may be transitted to the situation not included in the dataset, so the algorithm would select another wrong bitrate. Such compounding errors eventually lead to a continuous rebuffering event. As a result, supervised-learning methods lack the ablities to learn to recover from failures.

In this paper, we aim to leverage imitation learning, a method that closely related to RL and supervised learning, to learn the strategy from the expert policy samples. Imitation learning method reproduces desired behavior according to expert demonstrations [23]. The key idea of imitation learning is to allow the NN to explore environments and collect samples (just like RL) and learn the policy based on the expert policy (just as supervised learning). In detail, at step t, the algorithm infers a policy π_t at ABR state S_t. It then

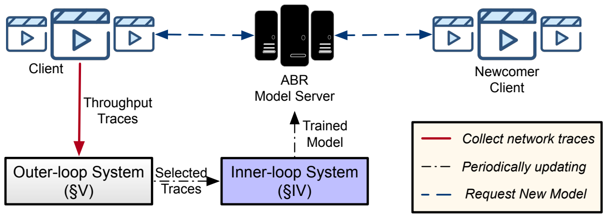


Fig. 8. Comyco System Overview. The system is composed of inner-loop system and outer-loop system.

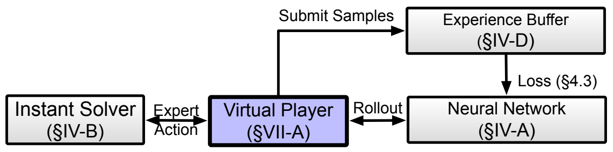


Fig. 9. Inner-loop Training System Work-flow Overview. Training methodologies are available in §IV-E.

computes a loss $l_t(\pi_t, \pi_t^*)$ w.r.t the expert policy π_t^*. After observing the next state S_{t+1}, the algorithm further provides a different policy π_{t+1} for the next step $t+1$ that will incur another loss $l_t(\pi_{t+1}, \pi_{t+1}^*)$. Thus, for each π_t in the class of policies $T \in \{\pi_0, \ldots, \pi_t\}$, we can find the policy $\hat{\pi}$ through any supervised learning algorithms (Eq. 1).

$$\hat{\pi} = \underset{\pi \in T}{arg\,min}\, \mathbb{E}_{s\sim d_\pi}\left[l_t(\pi_t, \pi_t^*)\right] \tag{1}$$

Figure 7[b] elaborates the principle of imitation learning-based ABR schemes: the algorithm attempts to explore the strategy in a range near the expert trajectory to avoid compounding errors.

C. Challenges for Lifelong Updating

Moreover, previous observation show that recent learning-based ABRs lack the abilities to deploy in the real-world scenarios since such methods are required to online updating effeciently for overcoming the time-varies of network conditions. At the same time, although we've already attempted to leverage imitation learning rather than reinforcement learning for fast updating, such methods still suffer from the large corpus of network traces on each period, which finally fail to converge in an acceptable time. Hence, list the third challenge: *Based on previously trained NN, is there any possibility of incrementally train it with a smaller yet efficient network traces?*

Our Solution. We consider the problem as a standard *catastrophic forgetting issue*, a phenomenon which can be observed as a dramatic performance degradation when some new tasks are added to an existing NN model. To tackle this fundamental problem, lifelong learning is one of the solutions which aims to preserve the performance on previous tasks while adapting to new data. To this end, we employ lifelong learning, also sometimes namely continual learning or incremental learning, on the proposed ABR system for continuously train the NN to fit the changes of network conditions. Furthermore, aiming to further reduce the training set, we implement a module which can dynamically filter the useful network traces from the real-world traces which instantly measured and submitted from the clients.

D. Comyco System Overview

To avoid these aforementioned limitations of previous ABR systems, we propose *Comyco*, an ABR system which uses lifelong imitation learning method to update the NN continuously. In detail, as illustrated in Figure 8, Comyco consists of two sub-systems, i.e., *inner-loop training system* and *outer-loop updating system*. Comyco's system workflow is shown as follow: Before the video starts, the video player, placed on the client-side, downloads latest NN model from *ABR model server* for making further decisions. Once the video session ends, the player collects the available throughput trace via past download chunk size and download time. At the same time, the collected trace will be submitted to the outer-loop updating system, which is placed on the server-side. Then the outer-loop system will instantly compute the gap between the current policy and the optimal strategy of the submitted trace and determine whether the trace should be *learned* by current NN during the next training loop. Next, for each time duration, the inner-loop training system, also placed on the server, will be enabled by the outer-loop system. It then update the NN w.r.t the selected traces efficiently via lifelong imitation learning training method. Finally, the trained model will be frozen and submitted to the ABR model server. The server then start waiting for the request of newcomer players.

IV. Inner-loop System Overview

In this section, we describe the proposed system in detail. Comyco's inner-loop system work-flow is illustrated in Figure 9. The sub-system is mainly composed of a NN, an ABR virtual player, an instant solver, and an experience replay buffer. We start by introducing the Comyco's NN architecture. Then we explain the basic training methodology. Finally, we further illustrate Comyco with a multi-agent framework.

A. NN Architecture Overview

Motivated by the recent success of on-policy RL-based methods, Comyco's learning agent is allowed to explore the environment via traditional rollout methods. For each epoch t, the agent aims to select next bitrate via a neural network (NN). We now explain the details of the agent's NN including its inputs, outputs, network architecture, and implementation.

Inputs. We categorize the NN into three parts, network features, video content features and video playback features ($S_k = \{C_k, M_k, F_k\}$). Details are described as follows.

▷ ***Past Network features.*** The agent takes past t chunks' network status vector $C_k = \{c_{k-t-1}, \ldots, c_k\}$ into NN, where c_i represents the throughput measured for video chunk i. Specifically, c_i is computed by $c_i = n_{r,i}/d_i$, in which $n_{r,i}$ is the downloaded video size of chunk i with selected bitrates r, and d_i means download time for video chunk $n_{r,i}$.

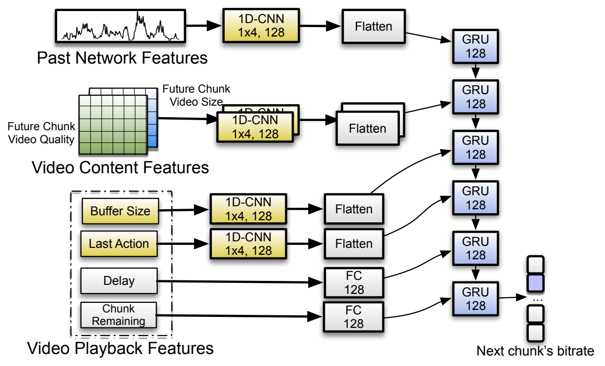


Fig. 10. Comyco's NN architecture Overview.

- ▷ ***Video content features.*** Besides that, we also consider adding video content features into NN's inputs for improving its abilities on detecting the diversity of video contents. In details, the learning agent leverages $M_k = \{N_{k+1}, V_{k+1}\}$ to represent video content features. Here N_{k+1} is a vector that reflects the video size for each bitrate of the next chunk $k+1$, and V_{k+1} is a vector which stands for the perceptual video quality metrics for each bitrate of the next chunk.
- ▷ ***Video playback features.*** The last essential feature for describing the ABR's state is the current video playback status. The status is represented as $F_k = \{v_{k-1}, B_k, D_k, m_k\}$, where v_{k-1} is the perceptual video quality metric for the past video chunk selected, B_k, D_k are vectors which stand for past t chunks' buffer occupancy and download time, and m_k means the normalized video chunk remaining.

Outputs. Same as previous work, we consider using discrete action space to describe the output. Note that the output is an n-dim vector indicating the probability of the bitrate being selected under the current ABR state S_k.

Implementation. As shown in Figure 10, for each input type, we use a proper and specific method to extract the underlying features. In details, we first leverage a single 1D-CNN layer with kernel=4, channels=128, stride=1 to extract network features to a 128-dim layer. We then use two 1D-CNN layers with kernel=1x4, channels=128 to fetch the hidden features from the future chunk's video content matrix. Meanwhile, we utilize 1D-CNN or fully connected layer to extract the useful characteristics from each metric upon the video playback inputs. The selected features are passed into a GRU layer and outputs as a 128-dims vector. Finally, the output of the NN is a *6-dims* vector, which represents the probabilities for each bitrate selected. We utilize *RelU* as the active function for each feature extraction layer and leverage *softmax* for the last layer.

B. Instant Solver

Once the sampling module rolls out an action a_t, we aim to design an algorithm to fetch all the *optimal* actions $\hat{a}_t$ with respect to current state s_t. Followed by these thoughts, we further propose *Instant Solver*. The key idea is to choose future chunk k's bitrate R_k by taking N steps ahead via an offline *virtual player*, and solves a specific *QoE maximization problem* with future network throughput measured C_t, in which the future real throughput can be successfully collected under both offline environments and real-world network scenarios. Inspired by recent model-based ABR work [11], we formulate the problem as demonstrated in Eq. 2, denoted as $QoE^N_{max}K$. In detail, the virtual player consists of a virtual time, a real-world network trace and a video description. At virtual time t_k, we first calculate download time for chunk k via $d_k(R_k)/C_k$, where d_k is the video chunk size for bitrate R_k, and C_k is average throughput measured. We then update B_{k+1} buffer occupancy for chunk $k+1$, in which δt_k reflects the waiting time such as Round-Trip-Time (RTT) and video render time, and B_{max} is the max buffer size. Finally, we refresh the virtual time t_{k+1} for the next computation. Note that the problem can be solved with any optimization algorithms, such as memoization, dynamic programming as well as Hindsight [38]. Ideally, there exists a trade-off between the computation overhead and the performance. We list the performance comparison of instant solver with different N in §VII-D. In this work, we set $N = 8$.

$$\max_{R_1,\dots,R_k,T_s} QoE^N, \qquad s.t.$$
$$\begin{cases} t_{k+1} = t_k + \dfrac{d_k(R_k)}{C_k} + \delta t_k, \\ C_k = \dfrac{1}{t_{k+1} - t_k - \delta t_k} \displaystyle\int_{t_k}^{t_{k+1}-\delta t_k} C_t dt, \\ B_{k+1} = \left[\left(B_k - \dfrac{d_k(R_k)}{C_k} \right)_+ + L - \delta t_k \right]_+, \\ B_1 = T_s, \\ B_k \in [0, B_{max}], R_k \in R, \forall k = 1, \dots, N. \end{cases} \tag{2}$$

C. Choice of Loss Functions for Comyco

In this section, we start with designing the loss function from the fundamental RL training methodologies. The goal of the RL-based method is to maximize the Bellman Equation, which is equivalent to maximize the value function $q_\pi(s,a)$ [35]. The equation is listed in Eq. 3, where $q_*(s,a)$ stands for the maximum action value function on all policies, $V_\pi(s)$ is the value function, $\pi(s,a;\theta)$ is the rollout policy. Thus, given an expert action $q_\pi(s,\hat{a}) = q_*(s,a)$, we can update the model via minimizing the gap between the true action probability $\hat{A}$ and π, where $\hat{A}$ is an one hot encoding in terms of $\hat{a}$. In this paper, we use cross entropy error as the loss function. Recall that the function can be represented as any traditional behavioral cloning loss methods [23], such as Quadratic, LI-loss and Hinge loss function. In addition, we find that the other goal of the loss function is to maximize the probabilities of the selected action, while the goal significantly reduces the aggressiveness of exploration, and finally, resulting in obtaining the sub-optimal performance. Thus, motivated by the recent work on RL [39], we add the entropy of the policy π to the loss function. It can encourage the algorithm to increase the exploration rate in the early stage and discourage it in the later stage. The loss function for Comyco is described in Eq 4.

$$\begin{aligned}\max_{\pi} V_{\pi}(s) &= \sum_{a\in A} \pi(a|s;\theta) q_{\pi}(s,a) \\ &= \max_{\pi}\max_{a} q_{\pi}(s,a) \\ &= \max_{a} q_{*}(s,a)\end{aligned} \tag{3}$$

$$L_{comyco} = -\sum \hat{A}\log \pi(s,a;\theta) + \alpha H(\pi(s;\theta)). \tag{4}$$

Here $\pi(s,a;\theta)$ is the rollout policy selected by the NN, $\hat{A}$ is the real action probability vector generated by the expert actor $\hat{a}$, $H(\pi(s;\theta)$ represents the entropy of the policy, α is a hyperparameter which controls the encouragement of exploration. In this paper, we set $alpha = 0.001$ and discuss L_{comyco} with different α in §VII-D.

D. Training Comyco with Experience Replay

Recent off-policy RL-based methods [40] leverage experience replay buffer to achieve better convergence behavior when training a function approximator. Inspired by the success of these approaches, we also create a sample buffer which can store the past expert strategies and allow the algorithm to randomly picks the sample from the buffer during the training process. We will discuss the effect of utilizing experience replay on Comyco in §VII-D.

E. Methodology

We summarize the Comyco's training methodology in Alg. 1. As shown, Comyco's workflow mainly consists of four parts:

1) *Samples an action via NN for the state observed.*
2) *Computes the expert action via instant solver according to the current state.*
3) *Updates the gradient by using loss function (Eq. 4).*
4) *Uses virtual player to emulate the next state.*

Algorithm 1 Inner-loop Overall Training Procedure

```
Require: Training model $\theta$, Instant Solver(§IV-B).
 1: procedure INNER-LOOP TRAINING
 2:     Initialize $\pi$.
 3:     Sample Training Batch $B = \{\}$.
 4:     Randomly pick trace, video from Network Dataset (§VII-A3).
 5:     Get State ABR state $s_t$.
 6:     repeat
 7:         Picks $a_t$ according to policy $\pi(s_t;\theta)$.
 8:         Expert action $\hat{a}_t$ = Instant Solver($s_t$).
 9:         $B \leftarrow B \bigcup \{s_t, \hat{a}_t\}$.
10:         Samples a batch $\hat{B} \in B$.
11:         Updates network $\theta$ with $\hat{B}$ using Eq.4;
12:         Produces next ABR state $S_{t+1}$ according to $s_t$ and $a_t$.
13:         if done then                          ▷ End of the video.
14:             Randomly pick trace, video from Network Dataset.
15:             Get State ABR state $s_t$.
16:         $t \leftarrow t + 1$
17:     until Converged
```

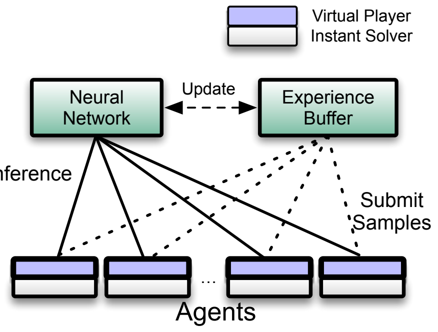


Fig. 11. Comyco's Multi-Agent Framework Overview.

F. Parallel Training

It's notable that the training process can be designed asynchronously, which is quite suitable for multi-agent parallel training framework. Inspired by the multi-agent training method [39], [19], we modify Comyco's framework from single-agent training to multi-agent training. As illustrated in Figure 11, Comyco's multi-agent training consists of three parts, a central agent with a NN, an experience replay buffer, and a group of agents with a virtual player and an instant solver. For any ABR state s, the agents use virtual player to emulate the ABR process w.r.t current states and actions given by the NN which placed on the central agent, and collect the expert action $\hat{a}$ through the instant solver; they then submit the information containing $\{s, \hat{a}\}$ to the experience replay buffer. The central agent trains the NN by picking the sample batch from the buffer. Note that this can happen asynchronously among all agents. By default, Comyco uses **12** agents, which is the same number of CPU cores of our PC, to accelerate the training process.

G. Implementation

We now explain how to implement Comyco. We use TensorFlow [41] to implement the training workflow and utilizing TFlearn [42] to construct the NN architecture. Besides, we use C++ to implement instant solver and the virtual player. Then we leverage Swig [43] to compile them as a python class. Next, we will show more details: Comyco takes the past sequence length $k = 8$ (as suggested by [13]) and future 1 video chunk features (as suggested by [11]) into the NN. We set learning rate $\alpha = 10^{-4}$ and use Adam optimizer [44] to optimize the model. For more details, please refer to our repository [3].

V. Outer-Loop System Overview

The key idea of the outer-loop sub-system is to reduce the number of training set with guaranteeing the training performance as much as possible. Thus, we propose outer-loop lifelong learning system as demonstrated in Figure . The sub-system includes several modules, such as optimal estimator (§V-A) and trace collector (§V-B). In this section, we introduce the modules and illustrate the training methodologies (§V-C, §V-D).

[3]https://github.com/thu-media/Comyco

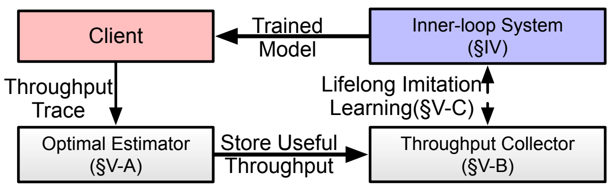


Fig. 12. Outer-loop Training System Work-flow Overview. Training methodologies are available in §V-D.

A. Optimal Estimator

Technically, the optimal estimator is a module which can compute the normalized QoE E_{tr} on a network throughput trace tr, where the trace is collected and reported from the client. As suggested by prior work, E_{tr} is defined as the ratio between the QoE performed by the current policy $Q_{tr,\theta}$ and the optimal QoE Q_{opt}. The formulation is listed in Eq. 5. Meanwhile, we leverage an instant solver (§IV-B) for estimating the optimal strategy. Specifically, we roll out the the best bitrate for each step via maximize the QoE objective (Eq. 2). In this work, we also set future horizon $N = 8$ since the near-optimal policy is well enough for this task [13], [11]. Note that we can also employ Hindsight [38] or another optimal ABR estimator [29] to replace the instant solver instead.

$$E_{tr} = \frac{Q_{tr,\theta}}{Q_{opt}}. \tag{5}$$

B. Trace Collector

Having computed the normalized QoE metric from the optimal estimator, we focus on learning a proper NN for the current network conditions efficiently. Ideally, the intuitive idea is to use whole collected trace to fine-tune the old NN. However, it's impractical since 1) the number of collected trace is so huge that we cannot train the NN to convergence in an allowable time. 2) most of the network conditions are similar to previous ones.(§III-B). Hence, our key idea is to pick proper traces into the trace collector, in which the normalized QoE of the trace is lower than the given threshold `Thres`. The inner-loop system then trains the NN from the network trace in the trace collector. Finally, Comyco generalize a strategy for the network conditions under next training duration. In this work, we set `Thres` as 0.8, and the training duration as 1 hour. We further investigate the influence of `Thres` on the porposed method in §VIII-F.

C. Loss function for lifelong learning method

Recall that the goal of the lifelong learning [45] is to avoid Catastrophic Forgetting, that means, the NN works well on the latest task but suffers from a bad performance on previous tasks. In this work, we pick Learning without Forgetting (LwF) [24], which uses outputs of the old models as soft targets on old tasks, as the learning algorithm. Note that LwF is a regularization-based method, which enables the lowest computation cost among all the previously proposed schemes, and works in the comparable performance in terms of the state-of-the-art approach [45]. The equation is listed in Eq. 6, where L_{Comyco} represents the loss function of Comyco (listed in Eq. 4), $\pi(s, a; \theta old)$ is roll out policy via the old NN (previous network), and $\pi(s, a; \theta)$ is the roll out policy for current NN. It's notable that we refer the old policy $\pi(s, a; \theta old)$ as a value, which means, it doesn't provide any gradients for the loss function. As suggested by previous work [24], we set λ=1.

$$L_{lifelong} = L_{Comyco} + \lambda \sum \pi(s; \theta_{old}) \log \pi(s; \theta). \tag{6}$$

D. Methodology

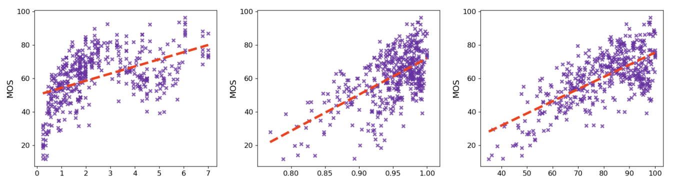
In general, we demonstrate the training methodology of outer-loop system in Alg. 2.

Algorithm 2 Outer-loop Overall Training Procedure

```
Require: Old model $\theta_{old}$, Training model $\theta$
Require: Trace Collector(§V-C), Threshold Thres.
Require: Optimal Estimator(§V-A).
 1: procedure OUTER-LOOP TRAINING
 2:     Receive trace from the client.
 3:     $E_{tr} \leftarrow$ Optimal Estimator(trace).
 4:     if $E_{tr}$ <Thres then
 5:         Store trace into Trace Collector.
 6:     if Training duration = 1 hour then                ▷
        Inner-loop training (§IV-E). We train $\theta$ on the trace set
        Trace Collector and $L_{Lifelong}$ using Alg.1.
 7:         Sample Training Batch $B = \{\}$.
 8:         repeat
 9:             Randomly pick trace, video from Trace
        Collector.
10:             Get State ABR state $s_t$.
11:             while not done do
12:                 Picks $a_t$ according to $\pi(s_t; \theta)$.
13:                 Get $\pi(s_t; \theta_{old})$ w.r.t $s_t$ and $\theta$.
14:                 Expert action $\hat{a_t}$ =InstantSolver ($s_t$).
15:                 $B \leftarrow B \bigcup \{s_t, \pi(s_t; \theta_{old}), \hat{a_t}\}$.
16:                 Samples a batch $\hat{B} \in B$.
17:                 Updates network $\theta$ with $\hat{B}$ using Eq.6;
18:                 Produces next ABR state $S_{t+1}$ w.r.t $s_t$, $a_t$.
19:                 if done then              ▷ End of the video.
20:                     Randomly pick trace and video again.
21:                     Get State ABR state $s_t$.
22:                 $t \leftarrow t + 1$
23:         until Converged
```

VI. QoE Metrics & Video Datasets

Upon constructing the Comyco's NN architecture with considering video content features, we have yet discussed how to train the NN. Indeed, we lack a video quality-aware QoE model and an ABR video dataset with video quality assessment. In this section, we use VMAF to describe the perceptual video quality of our work. We then propose a video quality-aware QoE metric under the guidance of real-world ABR QoE dataset [22]. Finally, we collect and publish a DASH video dataset with different VMAF assessments.



(a) Video Bitrate: 0.480 (b) SSIM: 0.592 (c) VMAF: 0.689

Fig. 13. Correlation comparison of video presentation quality metrics on the SQoE-III dataset [22]. Results are summarized by Pearson correlation coefficient [46].

A. QoE Model Setup

Motivated by the linear-based QoE metric that widely used to evaluate several ABR schemes [36], [11], [17], [13], [30], [32], we concluded our QoE metric $\texttt{QoE}_v$ as:

$$\texttt{QoE}_v = \alpha \sum_{n=1}^{N} q(R_n) - \beta \sum_{n=1}^{N} T_n + \gamma \sum_{n=1}^{N-1} [q(R_{n+1}) - q(R_n)]_+ - \delta \sum_{n=1}^{N-1} [q(R_{n+1}) - q(R_n)]_- , \quad (7)$$

where N is the total number of chunks during the session, R_n represents the each chunk's video bitrate, T_n reflects the rebuffering time for each chunk n, $q(R_n)$ is a function that maps the bitrate R_n to the video quality perceived by the user, $[q(R_{n+1}) - q(R_n)]_+$ denotes positive video bitrate smoothness, meaning switch the video chunk from low bitrate to high bitrate and $[q(R_{n+1}) - q(R_n)]_-$ is negative smoothness. Note that α, β, γ, δ are the parameters to describe their aggressiveness.

Choice of $q(R_n)$. Estimating QoE via handcrafted features from the client-side has lasted a long history [33], as several schemes seldom yield a reliable result. Revisiting these schemes, we find that Video quality assessment (VQA) plays a crucial part of QoE models. Nevertheless, most studies pick video bitrate, SSIM or PSNR as the inputs, which fail to characterize the video qualities of the entire video sessions. To better understand the correlation between video presentation quality and QoE metric, we test the correlation between mean opinion score (MOS) and video quality assessment (VQA) metrics, including video bitrate, SSIM [47] and Video Multi-method Assessment Fusion (VMAF) [21], under the Waterloo Streaming QoE Database III (SQoE-III)[4] [22], where SSIM is a image quality metric which used by D-DASH [14] and VMAF is an objective full-reference video quality metric which is formulated by Netflix to estimate subjective video quality. Results are collected with Pearson correlation coefficient [46] as suggested by [48]. Experimental results (Fig. 13) show that VMAF achieves the highest correlation among all candidates, with the improvements in the coefficient of 16.39%-43.54%. Besides, VMAF are also a popular scheme with great potential on both academia and industry [49]. We, therefore, set $q(R_n) = \texttt{VMAF}(R_n)$.

▷ ***DNN-based QoE Model.*** Moreover, we ask whether machine learning can help taming the complexity of optimal QoE and various video quality metrics or not. Thus, we develop a deep NN (DNN)-based Comyco to generalize a high-precision QoE from scratch. Results are listed in Table I. Unsurprisingly, Comyco's DNN model achieves state-of-the-art performance among all the QoE models, and improves over 12.43% in terms of Comyco's linear-based QoE model. Considering linear-based QoE model is more interpretable than DNN-based model, in this work, we treate linear-based QoE model as QoE_v instead. And we believe that using VMAF metric is more capable of characterizing perpetual video quality than previously scheme uses.

[4]SQoE-III is the *largest and most realistic dataset* for dynamic adaptive streaming over HTTP [22], which consists of a total of 450 streaming videos created from diverse source content and diverse distortion patterns.

TABLE I
PERFOMANCE COMPARISON OF QOE MODELS ON WATERLOO STREAMING SQOE-III [22]

QoE model	Type	VQA	SRCC
Pensieve's [13]	linear	-	0.6256
MPC's [11]	linear	-	0.7143
Bentaleb's [30]	linear	SSIMplus [50]	0.6322
Duanmu's [22]	linear	-	0.7743
Comyco's	linear	VMAF [21]	**0.7870**
P.NATS [51]	non-linear	O.21, O.22 [51]	0.8454
Comyco's DNN	non-linear	VMAF	**0.8834**

QoE Parameters Setup. Recall that main goal of our paper is to propose a feasible ABR system instead of a convincing QoE metric. In this work, we attempt to leverage linear-regression methods to find the proper parameters. Specifically, we randomly divide the SQoE-III database into two parts, 80% of the database for training and 20% testing. We follow the idea by [22] and run the training process for 1,000 times to mitigate any bias caused by the division of data. As a result, we set $\alpha = 0.8469$, $\beta = 28.7959$, $\gamma = 0.2979$, $\delta = 1.0610$. We leverage spearman correlation coefficient (SRCC), as suggested by [22], to evaluate the performance of our QoE model with existing proposed models and the median correlation and its corresponding regression model are demonstrated in Table I. As shown, QoE_v model outperforms recent work. In conclusion, the proposed QoE model is well enough to evaluate ABR schemes.

B. Video Datasets

To better improve the Comyco's generalization ability, we propose a video quality DASH dataset involves movies, sports, TV-shows, games, news and MVs. Specifially, we first collect video clips with highest resolution from Youtube, then leverage FFmpeg [52] to encode the video by H.264 codec and MP4Box [53] to *dashify* videos according to the encoding ladder of video sequences [22], [26]. Each chunk is encoded as 4 seconds. During the trans-coding process, for each video, we measure VMAF, VMAF-4K and VMAF-phone metric with the reference resolution of 1920×1080 respectively. In general, the dataset contains 86 complete videos, with 394,551 video chunks and 1,578,204 video quality assessments.

VII. EVALUATION FOR INNER-LOOP SYSTEM

In this section, we evaluate Comyco with inner-loop system under various network conditions and compare it with previously proposed ABR approaches.

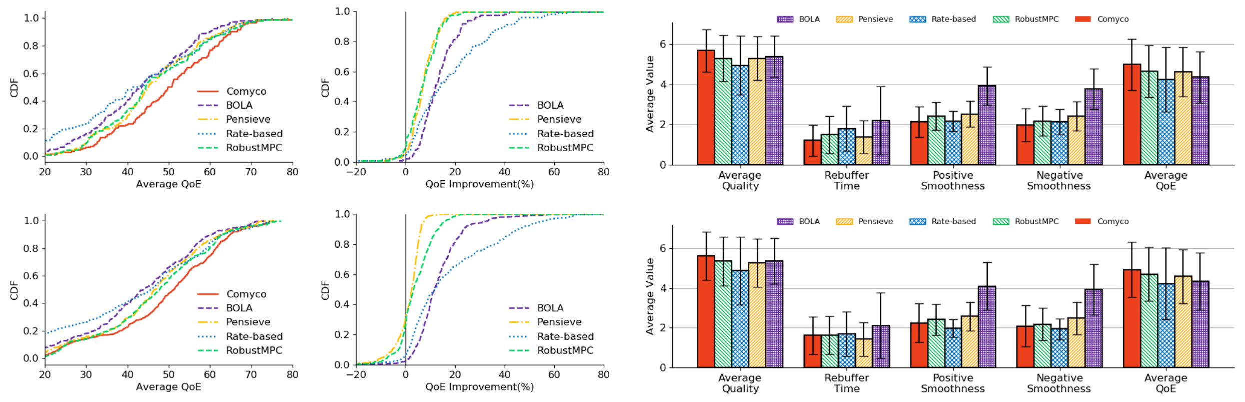


Fig. 14. Comparing Comyco with existing ABR approaches under the HSDPA and FCC network traces. Results are illustrated with CDF distributions, QoE improvement curves and the comparison of several underlying metrics (§VI-A).

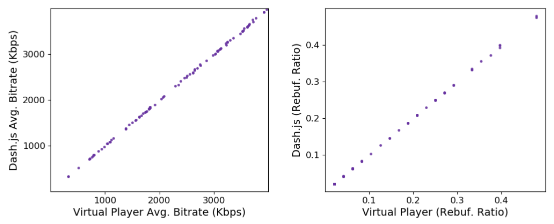


Fig. 15. Comparing the strategy between the virtual player and real `Dash.js` player. Results show the close correlation of strategy between two players.

A. Experimental Setup

In this experiment,we treat Comyco with inner-loop system as *Comyco*. Detailing the modules, that includes:

1) Virtual Player.: We design a faithful ABR offline virtual player to train Comyco via network traces and video descriptions. The player is written in C++ and Python3.6 and is closely refers to several state-of-the-art open-sourced ABR simulators including Pensieve, Oboe and Sabre [12]. We verify the accurateness of the proposed virtual player in Figure 15.

2) Testbed.: Our work consists of two testbeds. Both server and client run on the 12-core, Intel i7 3.7 GHz CPUs with 32GB RAM running Windows 10. Comyco can be trained efficiently on both GPU and CPU. Detailing the testbed, that includes:

▷ **Trace-driven emulation.** Following the instructions of recent work [13], [17], we utilize Mahimahi [54] to emulate the network conditions between the client (ChromeV73) and ABR server (SimpleHTTPServer by Python2.7) via collected network traces.

▷ **Real world Deployment.** Details are illustrated in §VII-E.

3) Network Trace Datasets.: We collect about 3,000 network traces, totally 47 hours, from public datasets for training and testing:

▷ **Chunk-level network traces:** including HSDPA [28]: a well-known 3G/HSDPA network trace dataset, we use a slide-window to upsampling the traces as mentioned by Pensieve (1000 traces, 1s granularity); FCC [27]: a broadband dataset (1000 traces, 1s granularity); Oboe [55] (428 traces, 1-5s granularity): a trace dataset collected from wired, WiFi and cellular network connections (Only for validation.)

▷ **Synthetic network traces:** uses a Markovian model where each state represented an average throughput in the aforementioned range[13]. We create network traces in over 1000 traces with 1s granularity.

4) ABR Baselines.: In this paper, we select several representational ABR algorithms from various type of fundamental principles:

▷ **Rate-based Approach (RB) [8]:** uses harmonic mean of past five throughput measured as future bandwidth.

▷ **BOLA [10]:** turns the ABR problem into a utility maximization problem and solve it by using the Lyapunov function. It's a buffer-based approach. We use BOLA provided by the authors [12].

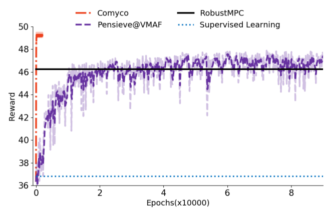
▷ **Robust MPC [11]:** inputs the buffer occupancy and throughput predictions and then maximizes the QoE by solving an optimization problem. We use C++ to implement *RobustMPC* and leverage QoE_v (§VI-A) to optimize the strategy.

▷ **Pensieve [13]:** the state-of-the-art ABR scheme which utilizes Deep Reinforcement Learning (DRL) to pick bitrate for next video chunks. We use the scheme implemented by the authors [56] but retrain the model for our work (§VII-B).

B. Comyco vs. ABR schemes

In this part, we attempt to compare the performance of Comyco with the recent ABR schemes under several network traces via the trace-driven virtual player. The details of selected ABR baselines are described in §VII-A4. We use *Envivo-Dash3*, a widely used [13], [11], [36], [17] reference video clip [26] and QoE_v to measure the ABR performance.

▷ ***Pensieve Re-training.*** We retrain Pensieve via our datasets (§VII-A3), NN architectures (§IV-A) and QoE metrics (§VI-A). Followed by recent work [17], our experiments use different entropy weights in the range of 5.0 to 0.1 and dynamically decrease the weight every 1000 iterations. Training time takes about 8 hours and we show that Pensieve outperforms RobustMPC, with an overall average QoE improvement of 3.5% across all sessions. Note that same experiments can improve the QoE_{lin} [11] by 10.5%. It indicates that QoE_v


(a) Epochs

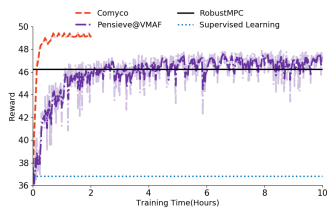

(b) Training Time

Fig. 16. Comparing the performance of Comyco with Pensieve and Supervised learning-based method under the HSDPA dataset. Comyco is able to achieve the highest performance with significant gains in sample efficiency.

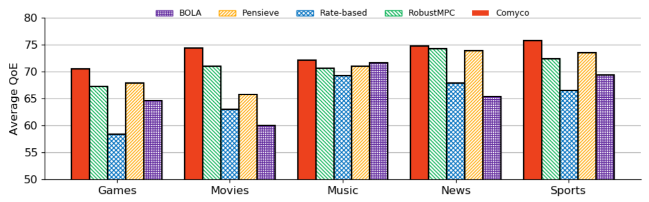


Fig. 17. Comparing Comyco with existing ABR approaches under the Oboe network traces and various types of videos.

cannot be easily improved since the metric reflects the real world's MOS score.

Comparison of Learning-based ABR schemes. Figure 16 illustrates the average QoE of learning-based ABR schemes on HSDPA datasets. We validate the performance of two schemes respectively during the training. Results are shown with two perspectives including Epoch-Average QoE and Training time-Average QoE and we see about **1700x** improvement in terms of the number of samples required and about **16x** improvement in terms of training time required. As expected (§III-B), we observe that supervised learning-based method fails to find a strategy, which thereby leads to the poor performance.

Comyco vs. Existing ABRs. Figure 14 shows the comparison of QoE metrics for existing ABR schemes (§VII-A4). Comyco outperforms recent ABRs, with the improvements on average QoE of 7.5% - 17.99% across the HSDPA dataset and 4.85%-16.79% across the FCC dataset. Especially, Besides, we also show the CDF of the percentage of improvent in QoE for Comyco over existing schemes. Comyco surpasses state-of-the-art ABR approach Pensieve for 91% of the sessions across the HSDPA dataset and 78% of the sessions across the FCC dataset. What's more, we also report the performance of underlying metrics including average video quality (VMAF), rebuffering time, positive and negative smoothness, as well as QoE. We find that Comyco is well behaved on the average quality metric, which improves 6.84%-15.64% compared with other ABRs. Moreover, Comyco is able to avoid rebuffering and bitrate changes, which performs as same as state-of-art schemes.

C. Comyco with Multiple Videos

To better understand how does Comyco perform on various videos, we randomly pick videos from different video types and utilize Oboe network traces to evaluate the QoE_v performances of the proposed methods. Oboe network traces have diversity network conditions, which brings more challenges for us to improve the performance. Figure 17 illustrates the comparison of QoE metrics for state-of-the-art ABR schemes under various video types. We find that Comyco generalizes well under all considered video scenarios, with the improvements on average QoE of 2.7%-23.3% compared with model-based ABR schemes and 2.8%-13.85% compared with Pensieve. Specifically, Comyco can provide high quality ABR services under movies, news, and sports, which are all the scenarios with frequent scene switches. We also find that Comyco fails to demonstrate overwhelming performance in serving music videos. It's really an interesting topic and we'll discuss it in future work.

TABLE II
COMYCO WITH DIFFERENT N AND REPLAY STRATEGIES.

$\alpha = 0.001/N$	5	6	7	8	9
Replay Off	0.883	0.893	0.917	0.932	0.942
Replay On	0.911	0.921	0.937	0.946	**0.960**
TimeSpan(Opt. Off)(ms)	1.56	8.74	58.44	389.68	2604.46

TABLE III
COMYCO WITH DIFFERENT α.

α	0.1	0.01	0.001	0.0001	0
k=4	0.883	0.895	**0.904**	0.881	0.867

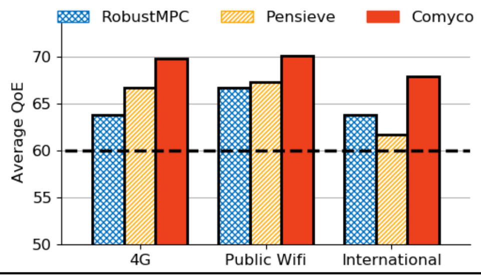
D. Ablation Study

In this section, we set up several experiments that aim to provide a thorough understanding of Comyco, including its hyperparameters and overhead. Note that, we have computed the offline-optimal results via dynamic programming and complete network status [13] before the experiment and treated it as a baseline.

Comparison of different future step N. We report normalized QoE and raw time span of Comyco with different N and replay experience strategy in Table II. Results are collected under the Oboe dataset. As shown, we find that experience replay can help Comyco learn better. Despite the outstanding performance of Comyco with N=9, this scheme lacks the algorithmic efficiency and can hardly be deployed in practice. Thus, we choose k=8 for harmonizing the performance and the cost.

Comyco with different α. Further, we compare the normalized QoE of Comyco with different α under the Oboe dataset. As listed in Table III, we confirm that $\alpha = 0.001$ represents the best parameters for our work. Meanwhile, results also prove the effective of utilizing entropy loss (§IV-C).

Comyco Overhead. We calculate [57] the number of floating-point operations (FLOPs) of Comyco and find that Comyco has the computation of 229 Kflops, which is only 0.15% of the light-weighted neural network ShuffleNet V2 [58] (146 Mflops). In short, we believe that Comyco can be successfully deployed on the PC and laptop, or even, on the mobile.



Network	RTT (ms)	Average Throughput (KB/s)	Std Throughput
4G	65.91	325.23	53.72
WiFi	15.58	292.98	27.65
Inter.	193.3	420.15	266.9

Fig. 18. Comparing Comyco with Pensieve and RobustMPC under the real-world network conditions. We take $QoE = 60$ as baselines.

E. Comyco in the Real World

We establish a full-system implementation to evaluate Comyco in the wild. The system mainly consists of a video player, an ABR server and an HTTP content server. On the server-side, we deploy an HTTP video content Server. On the client-side, we modify `Dash.js` [26] to implement our video player client and we use Chrome to watch the video. Moreover, we implement Comyco as a service on the ABR server. We evaluate the performance of proposed schemes under various network conditions including 4G/LTE network, WiFi network and international link (from Singapore to Beijing). Figure 18 illustrates network status, where μ is the average throughput measured and σ represents standard deviation from the average. For each round, we randomly picks a scheme from candidates and summarize the bitrate selected and rebuffering time for each chunk. Each experiment takes about 2 hours. Figure 18 shows the average QoE results for each scheme under different network conditions. It's clear that Comyco also outperforms previous state-of-the-art ABR schemes and it improves the average QoE of 4.57%-9.93% compared with Pensieve and of 6.43%-9.46% compared with RobustMPC.

VIII. Evaluation for Outer-loop System

In this section, we aim to evaluate the outer-loop system under various network conditions and compare it with several schemes, including previously proposed ABR schemes and outer-loop system with different updating strategies.

A. Experimental Setup

Considering the goal is to evaluate the effectiveness on lifelong learning rather than the performance of ABR streaming, we adopt virtual player (§VII-A2 to validate the outer-loop system via trace-driven emulation. Technically, unlike inner-loop system evaluation, we list network trace dataset and baselines as follows:

1) Network Trace Datasets: As described before, the sub-system is required to evaluate on continuous network throughput dataset. To this end, we utilize the large-scale network bandwidth dataset Kwai, which is collected by ourselves. The dataset is collected in the number over 860,000 traces from about 10 thousand users, totally 7 days from various network conditions, including wired, WiFi, cellular network, and so forth (§III-B).

2) Baselines: In this work, we pick several representative outer-loop system in different strategies as baselines.

- ▷ **Fine-tuning for 50 epochs (*Fine-tune-50*):** for each period t, we tune the trained model on the t-th hours' network traces for 50 epochs, lasting about 5 minutes on training time.
- ▷ **Fine-tuning for 300 epochs (*Fine-tune-300*):** meanwhile, we also fine-tune the trained model on network traces for 300 epochs for each time period. Note that the training time lasts over 30 minutes, which is impractical in practice.
- ▷ **Re-train Comyco in 50 epochs (*Retrain-50*):** for each duration t, we train Comyco on network traces in the range of t-th hours from scratch. Training time will take about 5 minutes.
- ▷ **Re-train Comyco in 300 epochs (*Retrain-300*):** we retrain Comyco for about 300 epochs since Comyco will be efficiently converged with an acceptable results. Recall that the training time lasts over half of the time duration.
- ▷ **RobustMPC [11]:** picks the bitrate by model predictive control method. As mentioned before, we also adopt C++ to implement *RobustMPC* and leverage QoE_v (§VI-A) to optimize the strategy.
- ▷ **Comyco Offline Training (*Comyco-offline*):** offline trains Comyco with inner-loop system's network trace dataset (§VII-A3). Note that we didn't further tune the NN model once Comyco's model has been frozen (§VII-D).

3) Implementation Details: In this experiment, we use QoE_v (§VI-A) to evaluate each scheme. For each duration t, we train the baselines on the network throughput traces with the range of t-th to t+1-th hour in the Kwai dataset, and validate them on the network dataset within t+1-th to t+2-th hour. In order to evaluate the fast convergence, we set the default training epoch as 50. The training time lasts about 5 minutes on our device (§VII-A2), and we evaluate the traces of 12 hours in one day. Furthermore, we also add *training models in 300 epochs (Fine-tine-300 and Retrain-300)* as strong baselines to better understand the gap between the proposed strategies and online-optimal policies. In this experiment, we treat the Comyco with outer-loop system as *lifelong Comyco*.

B. Comparison of Different Outer-loop Strategies

Figure 19 shows the average QoE curves of Lifelong Comyco and other baselines on Kwai dataset. We can see that Lifelong Comyco always rivals or outperforms other approaches. Specifically, Lifelong Comyco performs better than Comyco-offline scheme, with the improvements on average

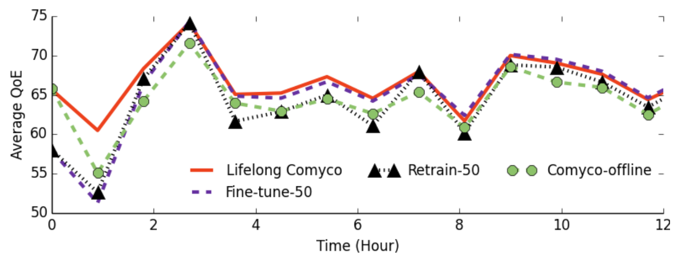


Fig. 19. Comparing Comyco with several baselines under Kwai dataset. Results are reported with QoE curves on each duration.

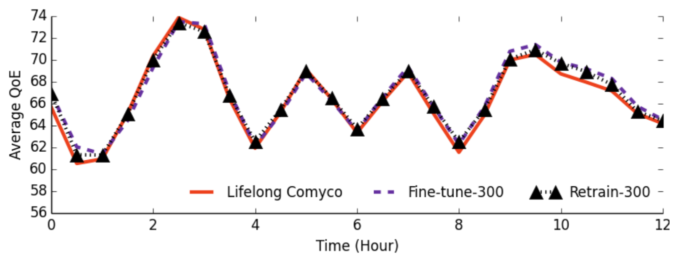


Fig. 20. Comparison of Comyco and online-optimal under Kwai dataset. Results are reported with QoE curves on each duration.

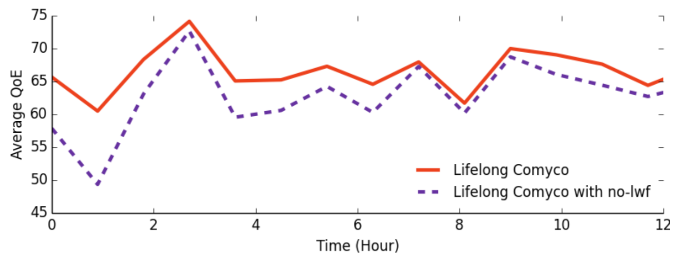


Fig. 21. Comparing the QoE of Comyco with the one without using LwF method. Results are collected under the Kwai dataset.

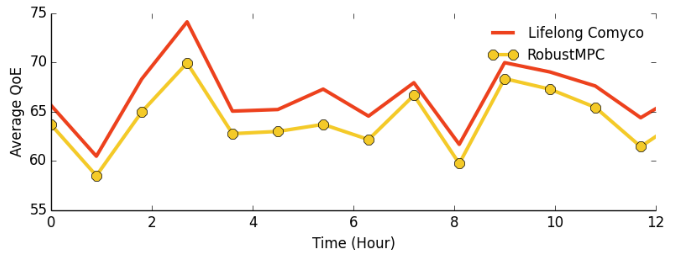


Fig. 22. Comparison of Comyco and state-of-the-art model-based algorithm RobustMPC under Kwai dataset. Results are also reported with QoE curves on each duration.

QoE of 1.07% - 9.81%. Another observation of this experiment demonstrates the lack of retrain and fine-tune approach: if the network situation changes dramatically (see time 0 to time 2 on Figure 19), such algorithms will not be able to provide reliable QoE to the users. Besides, fine-tune-50 works well when the network distribution changes steadily (from time 4-7) since the training set and the validation set are almost in the same distribution.

C. Lifelong Comyco vs. Comyco without LwF

In this experiment, we validate the effectiveness of LwF method. As shown in Figure 21, comparing the overall QoE performance of lifelong Comyco and Comyco without LwF, we observe that lifelong Comyco surpass the one without using LwF method on average increasing QoE of 1.51% - 21.41%. It makes sense since lifelong Comyco trains the NN with joint considering the previous network status and current network observed, whereas the Comyco without LwF algorithm diverges.

D. Lifelong Comyco vs. Online Optimal

To better understand the gap between lifelong Comyco and online optimal, we setup an experiment to evaluate the performance of lifelong Comyco, Fine-tune-300 and Retrain-300 (§VIII-A2) under the same network traces. As illustrated in Figure 20, we show that lifelong Comyco almost achieves online optimal on the entire session, with the decreasing of only 0.02%-3.34% compared with Fine-tune-300, and of 0.12%-3.33% in terms of Retrain-300. In particular, we also find that lifelong Comyco performs better than 16% of the sessions of Fine-tune-300 and Retrain-300 (improves from 0.17%-1.53%), which also proves the effectiveness of lifelong learning method.

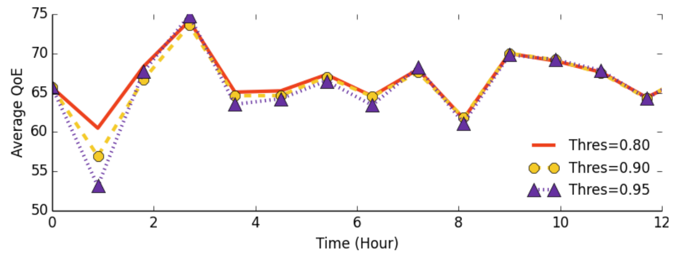


Fig. 23. Comparing the QoE of Comyco with the one without using LwF method. Results are collected under the Kwai dataset.

E. Lifelong Comyco vs. RobustMPC

Besides, we also compare the performance of lifelong Comyco with current state-of-the-art model-based approach RobustMPC. Results are illustrated as QoE curves in Figure 22. As expected, we can find that lifelong Comyco stands for the better scheme, outperforming RobustMPC on average QoE of 0.12% - 5.70%. Such observation also proves that a good ABR algorithm is required to update dynamically for fitting the changes of real-world network conditions [17].

F. Lifelong Comyco with Different Threshold `Thres`

In this experiment, we aim to understand the influence of threshold `Thres` for Comyco. In detail, we use three threshold candidates, involving $\{0.8, 0.9, 0.95\}$. We evaluate the lifelong Comyco with the proposed threshold on the same network environments respectively. Results are plotted in Figure 23. As shown, we see that `Thres`=0.8 represents the best parameter of lifelong Comyco. Especially, `Thres`=0.8 works well in time 2, while the other scheme fail to achieve a good result. Recall that the value is strongly depend on current task.

IX. Discussion

A. Why Comyco achieves better than Pensieve?

The disadvantage of RL has been discussed in (§III-B). Specifically, the key issue of Pensieve is to neglect exogenous inputs z_t (e.g. future throughput measured) when estimating policy gradient, since the advantage function may be overestimated or underestimated (value function $V(s_t;\theta_v) \neq V(s_t, z_t; \theta_v)$). On the contrary, in our work, Comyco consider z_t as the future throughput and outputs $a_t = \max_a QoE_v(s_t, a_t, z_t)$ (See Eq. 1). Same conclusions and proofs please refer to RL-based input variance algorithms [34].

B. Comyco vs. Oboe

We didn't compare Comyco with Oboe [17] since Oboe is an auto-tuning method, that means, it cannot be performed solely. Moreover, Oboe has no conflict with Comyco since they can perfectly fuse each other (just like Pensieve+Oboe).

C. Significance of accelerating training process

Fast efficiency training will bring several advantages, such as low overhead, fast model update, and easy deployment. Because the equilibria of RL method is not always Pareto efficient in offline-training tasks, e.g., fast generating ABR algorithms for personalized QoE or specific videos, new NN architecture exploration. In this paper, we only demonstrate one of use cases on fast training technologies.

D. Practical implementation

Learning-based ABR algorithms are struggling with its deployability. Specifically, Pensieve [13] is deployed on the server to avoid high computational costs on the client-side. However, in practice, most ABR algorithms are executed in the front-end to avert the extra latency connecting to the back-end [59], [60]. Thus, such ABR policy frameworks ([13], [17]) are theoretically effective but impractical [61].

As much as this work is NOT focused on the deployable problem of learning-based ABR algorithms, we still give some practical idea for implementation.

- We argue that deploying the model on the client is impractical since the computational cost is rather small for today's mobile (§VII-D). What's more, the user will optionally download the small-sized model, where the model size is even 50% smaller than the lowest video chunk size (by Tensorflow.js [62]).
- Several practical ABR schemes (i.e., PiTree [63], LIME [64] and ABRL [61]) have been proposed to distill the NN to a practical decision tree, a interpretable tabular, or a linear-based formula. Such schemes are also acceptable for appending into the Comyco system.
- Recent years have also seen several schemes who deploy the ABR algorithm as a service on the cloud. For example, Sun et al. [29] assume that throughput factors can be efficiently captured by Hidden-Markov-Model (HMM), then they optimize the model on the cloud with huge amounts of data. Oboe[17] attempts to place a dictionary, which mapping the throughput status $\{\mu, \sigma\}$ to the optimized traditional ABRs' ([11], [10]) parameters, on the cloud for assisting traditional algorithms to achieve higher performances in different network conditions. Meanwhile, deploying ABRs on the cloud has also been considered in industry. Thomas et al. [65], [66] proposed the Server and Network-assisted DASH (SAND) architecture to overcome the fact that the client-driven approach of DASH left less control to the network and service providers. RESA [67] employ an learning-based ABR proxy to make the suitable decision for each client. To this end, we believe that deploying an ABR service on the server or edge is also practical way for today's device and network environments.

X. Related Work

A. ABR schemes

Client-based ABR algorithms [2] are mainly organized into two types: model-based and learning-based.

Model-based. The development of ABR algorithms begins with the idea of predicting throughput. PANDA [7] predicts the future throughput for eliminating the ON-OFF steady issue. FESTIVE [8] estimates future throughput via the harmonic mean of the throughput measured for the past chunk downloads. However, due to the lack of throughput estimation method currently, these approaches still result in poor ABR performance. Meanwhile Most video client leverages a playback buffer to store the video content downloaded from the server temporarily. Thus, many approaches are designed to select the appropriate high bitrate next video chunk and avoid rebuffering events based on playback buffer size observed. BBA [9] proposes a linear criterion threshold to control the available playback buffer size. BOLA [10] turns the ABR problem into a utility maximization problem and solve it by using the Lyapunov function. However, the buffer-based approach fails to tackle the long-term bandwidth fluctuation problem. Hence, mixed model-based approaches, e.g., MPC [11], select bitrate for the next chunk by adjusting its throughput discount factor based on past prediction errors and estimating its playback buffer size. Nevertheless, these approaches require careful tuning because they rely on parameters that are quite sensitive to network conditions, resulting in poor performance in unexpected network environments. What's more, Akhtar et al. [17] propose an auto-tuning method to improve the model-based ABR's performance.

Learning-based: Several attempts have been made to optimize the ABR algorithm based on RL method due to the difficulty of tuning mixed approaches for handling different network conditions. Pensieve [13] is a system that leverages RL to select bitrate for future video chunks. D-DASH [14] uses Deep Q-learning method to perform a comprehensive evaluation based on state-of-the-art algorithms. Tiyuntsong optimizes itself towards a rule or a specific reward via the competition with two agents under the same network condition [16].

B. Imitation Learning meets Networking

Imitation learning [68] has been widely used in the various fields including networking scheduling and network congestion control schemes. Tang et al. [69] propose real-time deep learning based intelligent network traffic control method to represent the considered Wireless Mesh Network (WMN) backbone via imitation learning. Indigo [70] uses DAgger [71] to train a congestion-control NN scheme in the offline network emulator.

C. Lifelong learning methods

Lifelong learning (or namely continual learning and incremental learning) has become one of the research hotspots for tackling catastrophic forgetting problem. We categorize lifelong learning strategies into several types [72], including architectural and rehearsal as well as regularization strategies [45].

Architectural-based. the strategies train separated models for different tasks, and there often exists a selector to determine which model to launch during the validating process. E.g., Expert Gate [73] pick expert model to launch during inference, and determine which training strategy to apply for the new task via auto-encoders. Other strategies include but not limited to Incremental Learning through Deep Adaptation (DAN) [74], Copy Weight with Re-init (CWR) [75], etc.

Rehearsal-based. Rehearsal-based lifelong learning method is an experience replay-based method which picks past information periodically when adapting to the new data for avoiding catastrophic forgetting. For instance, Gradient Episodic Memory (GEM) [76] uses a fixed memory to store a subset of old patterns. Besides, motivated by the recent success of Generative Adversarial Networks (GAN) [77], several GAN-based method, i.e., Deep Generative Replay (DGR) [78] and Replay through Feedback (RtF) [79], have been proposed to train generative models for generating adversarial samples based on previous task distributions.

Regularization-based. Several approaches have been proposed to extend the loss function with additional terms for guaranteeing the performance on previous tasks. Learning without Forgetting (LwF) [24] uses outputs of the old models as soft targets of old tasks. These soft targets are considered as a substitute for the data of previous tasks, which cannot be accessed in lifelong learning settings. Elastic Weights Consolidation (EWC) [80] estimates the importance of parameters by the diagonal of the Fisher information matrix and uses individual penalty for each previous task. Other approach are listed in similar functions, e.g., Synaptic Intelligence (SI) and Memory Aware Synaptic (MAS) [81].

In general, such lifelong learning methods always suffer from a key issue: one method performs well in some experimental settings may totally fail in others [82].

XI. Conclusion

In this work, we propose Comyco, a learning-based ABR system which aim to thoroughly improve the performance of learning-based algorithm. In general, Comyco makes the contributions as follows: First, we construct Comyco as a video quality-based ABR system, including its NN architectures, datasets and QoE metrics. With trace-driven emulation and real-world deployment. Second, to overcome the sample inefficiency problem, we leverage imitation learning method to *guide* the algorithm to explore and exploit the *better* policy rather than stochastic sampling. Third, through data-driven analysis we find Comyco should be updated continually over time. Thus, we present lifelong learning-based Comyco, aiming to improve its adaption on network status. Massive of experimental results show that Comyco significantly improves the performance, effectively accelerates the training process, and achieves lifelong training on the entire session.

Acknowledgement.

This paper extends [83] by adding lifelong learning methods which significantly improves learning-based ABR algorithms to work in practice.

References

[1] Cisco, "Cisco visual networking index: Forecast and methodology, 2016-2021," 2017. [Online]. Available: https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[2] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over http," *IEEE Communications Surveys & Tutorials*, 2018.

[3] "Youtube," 2019. [Online]. Available: https://www.youtube.com

[4] "Hulu," 2019. [Online]. Available: https://www.hulu.com

[5] "Youku," 2019. [Online]. Available: https://www.youku.com

[6] "iqiyi," 2019. [Online]. Available: https://www.iqiyi.com

[7] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[8] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *TON*, vol. 22, no. 1, pp. 326–340, 2014.

[9] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2015.

[10] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *INFOCOM 2016, IEEE*. IEEE, 2016, pp. 1–9.

[11] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *ACM SIGCOMM Computer Communication Review*. ACM, 2015, pp. 325–338.

[12] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: improving bitrate adaptation in the dash reference player," in *Proceedings of the 9th MMSys*. ACM, 2018, pp. 123–137.

[13] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the 2017 ACM SIGCOMM Conference*. ACM, 2017, pp. 197–210.

[14] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-dash: A deep q-learning framework for dash video streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, Dec 2017.

[15] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "Hotdash: Hotspot aware adaptive video streaming using deep reinforcement learning," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 165–175.

[16] T. Huang, X. Yao, C. Wu, R.-X. Zhang, and L. Sun, "Tiyuntsong: A self-play reinforcement learning approach for abr video streaming," *arXiv preprint arXiv:1811.06166*, 2018.

[17] Z. Akhtar and et al., "Oboe: auto-tuning video abr algorithms to network conditions," in *SIGCOMM 2018*. ACM, 2018, pp. 44–58.

[18] "Kuaishou," 2019. [Online]. Available: https://www.kuaishou.com

[19] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun, "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, 2018, pp. 1208–1216.

[20] R. Mendonca, A. Gupta, R. Kralev, P. Abbeel, S. Levine, and C. Finn, "Guided meta-policy search," *arXiv preprint arXiv:1904.00956*, 2019.

[21] R. Rassool, "Vmaf reproducibility: Validating a perceptual practical video quality metric," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on*. IEEE, 2017, pp. 1–2.

[22] Z. Duanmu, A. Rehman, and Z. Wang, "A quality-of-experience database for adaptive video streaming," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 474–487, June 2018.

[23] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[24] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[25] "Http live streaming," https://developer.apple.com/streaming/, 2019.

[26] "Dash industry forum — catalyzing the adoption of mpeg-dash," 2019. [Online]. Available: https://dashif.org/

[27] M. F. B. Report, "Raw data measuring broadband america 2016," https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016, 2016, [Online; accessed 19-July-2016].

[28] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 2013, pp. 114–118.

[29] Y. Sun and et al., "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *SIGCOMM 2016*. ACM, 2016, pp. 272–285.

[30] A. Bentaleb, A. C. Begen, and R. Zimmermann, "Sdndash: Improving qoe of http adaptive streaming using software defined networking," in *Proceedings of ACM MultiMedia 2016*. ACM, 2016, pp. 1296–1305.

[31] Z. Wang, "Video qoe: Presentation quality vs. playback smoothness," Jul 2017. [Online]. Available: https://www.ssimwave.com/science-of-seeing/video-quality-of-experience-presentation-quality-vs-playback-smoothness/

[32] Y. Qin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue, "Abr streaming of vbr-encoded videos: characterization, challenges, and solutions," in *Proceedings of CoNeXT 2018*. ACM, 2018, pp. 366–378.

[33] Z. Duanmu, K. Zeng, K. Ma, A. Rehman, and Z. Wang, "A quality-of-experience index for streaming video," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 1, pp. 154–166, 2017.

[34] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, "Variance reduction for reinforcement learning in input-driven environments," *international conference on learning representations*, 2019.

[35] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[36] P. G. Pereira, A. Schmidt, and T. Herfet, "Cross-layer effects on training neural algorithms for video streaming," in *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2018, pp. 43–48.

[37] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," *arXiv preprint arXiv:1703.09327*, 2017.

[38] T.-Y. Huang, C. Ekanadham, A. J. Berglund, and Z. Li, "Hindsight: Evaluate video bitrate adaptation at scale," in *Proceedings of the 10th ACM Multimedia Systems Conference*, ser. MMSys '19. New York, NY, USA: ACM, 2019, pp. 86–97. [Online]. Available: http://doi.acm.org/10.1145/3304109.3306219

[39] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[41] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[42] Y. Tang, "Tf. learn: Tensorflow's high-level module for distributed machine learning," *arXiv preprint arXiv:1612.04251*, 2016.

[43] D. M. Beazley *et al.*, "Swig: An easy to use tool for integrating scripting languages with c and c++." in *Tcl/Tk Workshop*, 1996, p. 43.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[45] X. Yao, T. Huang, C. Wu, R. Zhang, and L. Sun, "Adversarial feature alignment: Avoid catastrophic forgetting in incremental task lifelong learning," *Neural computation*, pp. 1–26.

[46] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[47] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," pp. 2366–2369, 2010.

[48] T. Abar, A. B. Letaifa, and S. El Asmi, "Machine learning based qoe prediction in sdn networks," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2017, pp. 1395–1400.

[49] A. Aaron, Z. Li, M. Manohara, J. Y. Lin, E. C.-H. Wu, and C.-C. J. Kuo, "Challenges in cloud based ingest and encoding for high quality streaming media," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 1732–1736.

[50] A. Rehman, K. Zeng, and Z. Wang, "Display device-adapted video quality-of-experience assessment," in *Human Vision and Electronic Imaging XX*, vol. 9394. International Society for Optics and Photonics, 2015, p. 939406.

[51] W. Robitza, M.-N. Garcia, and A. Raake, "A modular http adaptive streaming qoe model—candidate for itu-t p. 1203 ("p. nats")," in *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2017, pp. 1–6.

[52] FFmpeg, "Ffmpeg." [Online]. Available: http://ffmpeg.org/

[53] GPAC, "Mp4box." [Online]. Available: https://gpac.wp.imt.fr/mp4box/

[54] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: accurate record-and-replay for http," pp. 417–429, 2015.

[55] Usc-Nsl, "Usc-nsl/oboe," Oct 2018. [Online]. Available: https://github.com/USC-NSL/Oboe

[56] Mao, "hongzimao/pensieve," Jul 2017. [Online]. Available: https://github.com/hongzimao/pensieve

[57] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[58] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 116–131.

[59] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 157–168.

[60] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE multimedia*, vol. 18, no. 4, pp. 62–67, 2011.

[61] H. Mao, S. Chen, D. Dimmery, S. Singh, D. Blaisdell, Y. Tian, M. Alizadeh, and E. Bakshy, "Real-world video adaptation with reinforcement learning," in *ICML 2019 Workshop*, 2019.

[62] D. Smilkov, N. Thorat, Y. Assogba, A. Yuan, N. Kreeger, P. Yu, K. Zhang, S. Cai, E. Nielsen, D. Soergel *et al.*, "Tensorflow. js: Machine learning for the web and beyond," *arXiv preprint arXiv:1901.05350*, 2019.

[63] Z. Meng, J. Chen, Y. Guo, and M. Xu, "Pitree: Practical implementation of abr algorithms using decision trees," in *2019 ACM Multimedia Conference on Multimedia Conference*. ACM, 2019.

[64] A. Dethise, M. Canini, and S. Kandula, "Cracking open the black box: What observations can tell us about reinforcement learning agents," in *Proceedings of the 2019 Workshop on Network Meets AI & ML*. ACM, 2019, pp. 29–36.

[65] E. Thomas, M. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey, "Enhancing mpeg dash performance via server and network assistance," 2015.

[66] E. Thomas, M. van Deventer, T. Stockhammer, A. C. Begen, M.-L. Champel, and O. Oyman, "Applications and deployments of server and network assisted dash (sand)," 2016.

[67] Y. Wang, H. Wang, J. Shang, and H. Tuo, "Resa: A real-time evaluation system for abr," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 1846–1851.

[68] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, p. 21, 2017.

[69] F. Tang, B. Mao, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 154–160, February 2018.

[70] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 731–743.

[71] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[72] D. Maltoni and V. Lomonaco, “Continuous learning in single-incremental-task scenarios,” *arXiv preprint arXiv:1806.08568*, 2018.

[73] R. Aljundi, P. Chakravarty, and T. Tuytelaars, “Expert gate: Lifelong learning with a network of experts.” in *CVPR*, 2017, pp. 7120–7129.

[74] A. Rosenfeld and J. K. Tsotsos, “Incremental learning through deep adaptation,” *arXiv preprint arXiv:1705.04228*, 2017.

[75] V. Lomonaco and D. Maltoni, “Core50: a new dataset and benchmark for continuous object recognition,” in *Conference on Robot Learning*, 2017, pp. 17–26.

[76] D. Lopez-Paz *et al.*, “Gradient episodic memory for continual learning,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6467–6476.

[77] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[78] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2017, pp. 2994–3003.

[79] G. M. van de Ven and A. S. Tolias, “Generative replay with feedback connections as a general strategy for continual learning,” *arXiv preprint arXiv:1809.10635*, 2018.

[80] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, p. 201611835, 2017.

[81] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, “Memory aware synapses: Learning what (not) to forget,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[82] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, “Measuring catastrophic forgetting in neural networks,” in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[83] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, “Comyco: Quality-aware adaptive video streaming via imitation learning,” *arXiv preprint arXiv:1908.02270*, 2019.

PLACE PHOTO HERE

Tianchi Huang received his M.E degree in the Department of Computer Science and Technology in Guizhou University in 2018. Currently he is a Ph.D student in the Department of Computer Science and Technology at Tsinghua University, advised by Prof. Lifeng Sun.

His research work focuses on the multimedia network streaming, including transmitting streams, overlay path construction and edge-assisted content delivery. He received Best Student Paper Awards presented by ACM Multimedia System 2019 Workshop.

PLACE PHOTO HERE

Xin Yao is currently a Ph.D. candidate in the Department of Computer Science and Technology at Tsinghua University, advised by Prof. Lifeng Sun. He received his bachelor’s degree from the Department of Computer Science and Technology at Tsinghua University in 2016. His research interests focus on federated learning, lifelong/continual learning, and transfer learning.

PLACE PHOTO HERE

Rui-Xiao Zhang received his B.E degree in Electronic Engineering Department in Tsinghua University in 2017. Currently, he is pursuing his Ph.D candidate in Department of Computer Science and Technology, Tsinghua University, China.

His research interests lie in the area of content delivery networks, the optimization of multimedia streaming and cloud computing. He received Best Student Paper Awards presented by ACM Multimedia System 2019 Workshop.

PLACE PHOTO HERE

Chao Zhou Chao Zhou received his Ph.D. degree from the Institute of Computer Science and Technology, Peking University, Beijing, China, in 2014. He has been with Beijing Kuaishou Technology Co., Ltd. as an Algorithm Scientist. Before joining Kuaishou, he was a Senior Research Engineer with the Media Technology Lab, CRI, Huawei Technologies CO., LTD, Beijing, China. Dr. Zhou’s research interests include HTTP video streaming, joint source-channel coding, and multimedia communications and processing. He has been the reviewer for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY , IEEE TRANSACTIONS ON MULTIMEDIA , IEEE TRANSACTIONS ON WIRELESS COMMUNICATION and so on. He received Best Paper Award presented by IEEE VCIP 2015, and Best Student Paper Awards presented by IEEE VCIP 2012.

PLACE PHOTO HERE

Chenglei Wu received the Master degrees in Tsinghua University. He is currently a Ph.D with the Computer Science and Technology Department of Tsinghua University. His research interests focus on 360 video streaming, adaptive video streaming and routing.

PLACE PHOTO HERE

Bing Yu received the B.E degree from Tsinghua University, Beijing, China. He has been with Beijing Kuaishou Technology.

PLACE PHOTO HERE

Lifeng Sun received the B.S and Ph.D degrees in system engineering from National University of Defense Technology, Changsha, Hunan, China, in 1995 and 2000, respectively. He joined Tsinghua University since 2001. He is currently a Professor with the Computer Science and Technology Department of Tsinghua University, Beijing.

Dr.Sun’s research interests include the area of networked multimedia, video streaming, 3D/multiview video coding, multimedia cloud computing, and social media.