Improving Mobile Interactive Video QoE via Two-Level Online Cooperative Learning

Huanhuan Zhang, Anfu Zhou, *Member, IEEE*, and Huadong Ma, *Fellow, IEEE*

Abstract—Machine learning models, particularly reinforcement learning (RL), have demonstrated great potential in optimizing video streaming applications. However, the state-of-the-art solutions are limited to an *"offline learning"* paradigm, i.e., the RL models are trained in simulators and then are operated in real networks. As a result, they inevitably suffer from the simulation-to-reality gap, showing far less satisfactory performance under real conditions compared with simulated environment. In this article, we close the gap by proposing Legato, an online RL framework for real-time mobile interactive video systems. Legato puts many individual RL agents directly into the video system, which make video bitrate decisions in real-time and evolve their models over time. Legato then employs a two-level cooperative learning mechanism to enhance video QoE. First, Legato proposes a score-based robust learning algorithm to eliminate risks of quality degradation caused by the RL model's exploration attempts. Then, Legato adaptively aggregates agents following a network condition-aware manner to form its corresponding high-level RL model that can help each individual to react to unseen network conditions. We implement Legato on an interactive real-time video system. Based on the exhaustive evaluations, we find that Legato outperforms the state-of-the-art algorithms significantly across a wide range of QoE metrics.

Index Terms—Interactive video, online reinforcement learning, cooperative learning, learning aggregation

1 Introduction

Real-time interactive video communication carries a dominant fraction of today's Internet traffic [1], largely due to many mainstream interactive video applications, such as Facetime, Skype, Google Hangouts, WeChat, *etc.* With the upgrade of the LTE-Advanced and 5G network infrastructures, new use cases are also rapidly emerging, such as live VR broadcasting [2], [3], 3D volumetric video [4], cloud gaming [5], [6], tele-operation of surgery robots or vehicles [7], [8]. Such interactive video applications impose the toughest demand on the network in terms of bandwidth and latency. Although the telecom infrastructure strives to match the demands, it only promises best effort service, and has to rely on the application itself to adapt to the highly dynamic network conditions.

Different from the classic video-on-demand (VoD) services, whose video contents are generated in advance, based on chunk-level HTTP transmission and usually suffer from second-level latency [9], the contents in interactive video are generated in real-time, and need to be transmitted and received in millisecond-level [10]. Such low-latency transmission is controlled by the RTP/RTCP protocols in real-time. To maintain high quality of experience (QoE), traditional interactive video applications adopt rule-based RTP/RTCP protocols, e.g., congestion control at the transport layer and video bitrate adaption on the application. However, the pre-programmed rules fall short of accommodating the highly heterogeneous modern Internet consisting of cellular/WiFi wireless links, cross-continent fiber links, intra-cloud data-center links, *etc.*, all with diverse bandwidth, latency, and buffer capacities. In recent years, data-driven approaches [11], [12], [13], [14], [15] have been explored to improve video QoE. For instance, Pensieve [12] employs reinforcement learning (RL) in video streaming systems to adapt the video bitrate, aiming to mitigate the risk of frame stalling while maximizing bandwidth utilization. Concerto [15] customizes imitation learning to harmonize the transport layer and video codec, so as to reduce stalling in mobile real-time video.

While showing potential, these solutions commonly adopt a *"learning offline, running online"* strategy. The learning models are trained in simulators or emulators, and then the trained models are directly deployed and tested in real applications. Unfortunately, such offline learning models lead to far less satisfactory performance in the real world, despite the high performance achieved in simulator/emulator [16]. Moreover, they may exhibit even the opposite performance characteristics from those observed in simulations, as validated in [15]. The root cause lies in two interrelated factors: *(i)* It is very challenging to faithfully simulate the complicated real-world Internet dynamics [17]. Individual routers along a network path can bear various capabilities and states, e.g., multiple-flow competition, packet drop strategy, load-induced quality fluctuation, let alone the complicated interaction among multiple protocol layers on end devices. *(ii)* In essence, the capacity of a data-driven algorithm is strictly bounded by its learning environment. Its

• *The authors are with the Beijing Key Lab of Intelligent Telecommunication Software and Multimedia, Beijing University of Posts and Telecommunication, Beijing 100876, China. E-mail: {zhanghuanhuan, zhouanfu, mhd}@bupt.edu.cn.*

Manuscript received 28 Aug. 2021; revised 3 May 2022; accepted 20 May 2022. Date of publication 2 June 2022; date of current version 31 Aug. 2023.
This work was supported in part by the National Key R&D Program of China under Grant 2019YFB2102202, in part by the Innovation Research Group Project of NSFC under Grant 61921003, in part by NSFC under Grants 61720106007 and 61832010, in part by 111 Project under Grant B18008, and in part by Youth Top Talent Support Program.
(Corresponding author: Huadong Ma.)
Digital Object Identifier no. 10.1109/TMC.2022.3179782

experience acquired via trial-and-error in a simulator may become stale when coping with the real-world Internet.

In this work, we seek to close the simulation-to-reality gap [18], [19] by proposing Legato, an *online reinforcement learning based adaptation framework for mobile interactive video*. Legato runs and keeps training directly on an interactive video system, so as to learn and respond to the real network conditions. Comparing with conventional offline learning, one can collect and concatenate network traces from individual user, and feed them into a simulator/emulator to train an RL model [12], [15]. In this way, the RL model can explore diverse environments to enrich its experience, and converge to a universal model with swarm intelligence learned from all users. However, for the proposed online learning, a massive number of real-time video sessions run simultaneously, during which the learning algorithm needs to evolve with each session in real time. Therefore, the key bottleneck lies in how to transform from serial offline learning to parallel online learning while still harnessing the swarm intelligence. Thus Legato is not merely a straightforward "offline-to-online" learning environment shift. Instead, we identify two unique design challenges.

(i) How to ensure highly accurate and robust online learning? In order to let each user effectively learn in its own environment, we design a deep reinforcement learning model based on the state-of-the-art PPO algorithm [20], and associate one individualized RL model instance with each user. In this way, each user leads to a different RL model with its own learning experience and can cope with its network dynamics. In essence, an RL algorithm learns by following a trial-and-error principle, which risks disrupting the system when applied directly to online learning. Specifically, the algorithm may cause the un-robustness problem, i.e., taking incorrect exploitation actions (e.g., selecting very high video bitrates under poor network conditions) that lead to catastrophic effects (e.g., severe congestion and thus low QoE), especially during the early learning phase when the RL model is not well trained yet. To handle the problem, we design a robust cooperative learning scheme in which PPO-based actor makes an in-depth blend with the legacy rule-based video bitrate algorithm. To guarantee such blended manner's effectiveness and robustness, we propose an *adaptive score mechanism* to determine which actor has a larger potential of generating higher QoE, thus has a larger score value in it, corresponding to better-optimized decision, and vice visa. To endow the scoring mechanism with such ability, we represent it as a learning framework that is controlled by a value-based RL algorithm named DDPG [21]. In such a cooperatively interactive process, robustness is improved during the online learning process, then the model will learn to evolve to be a self-dependent and robust video bitrate adaptation algorithm.

(ii) How to learn from massive concurrent real-time interactive video? To harness the swarm intelligence with each individual learning user, we propose an adaptive learning aggregation. Concretely, we aggregate all users' experiences following a *federated learning* principle, so as to form a high-level model that can react to any network dynamics unseen by individuals. To conduct effective aggregation, we have an in-depth clustering analysis to reveal the correlation of network condition and video QoE relying on thousands of real-world video sessions, then we formulate the effective aggregation standards and further conduct network condition-aware aggregation policy to meet the swarm intelligence requirements.

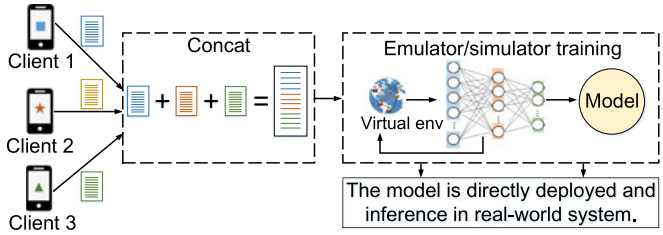


Fig. 1. The architecture of traditional offline learning.

To sum up, we name Legato's score-based cooperative learning as the *Level-1* optimization mechanism, and the adaptive learning aggregation as the *Level-2*. These two levels are coupled and cooperated iteratively, so as to strike a balance between individualized experience and robustly swarm intelligence.

We compare Legato with the state-of-the-art real-time video algorithms OnRL[1], and conduct a range of controlled experiments based on the practical video traces on our real-time video transmission prototype system. Experimental results demonstrate that Legato not only improves the robustness of video transmission, (e.g., decreasing the stall rate by 8.29% and low fps frequency by 43.90%), but also enhances the video quality (e.g., 6.60% gains in video throughput), compared with the baseline algorithms.

Contributions: Legato resolves general problems that arise when RL meets real-time applications over the highly dynamic and heterogeneous mobile Internet. The specific contributions are summarized as follows:

(i) We propose an iterative RL algorithm that enables online learning directly from massive concurrent interactive video sessions, instead of conventional simulation/emulation based offline learning (Section 3).

(ii) We design a two-level cooperative mechanism to ensure highly effective individual learning while still harnessing the swarm intelligence, mainly by proposing a novel score-based cooperative learning mechanism to boost the robustness of online learning (Section 4), and identifying an adaptive network condition-based learning aggregation algorithms (Section 5).

(iii) We implement and deploy Legato on a real-time interactive video system (Section 6) and validate its remarkable performance gains over the state-of-the-art solutions (Section 7).

2 Design Overview

Existing machine learning-driven real-time video transport systems mostly adopt the "offline learning" approach [11], [12], [15] as illustrated in Fig. 1. Such a system needs to first collect a plethora of simulated [11], [12], [15] or real network traces [16], [23] from different users. Then these traces are serialized, i.e., concatenated regardless of their inherent

1. OnRL is the primary conference version of Legato published in [22].

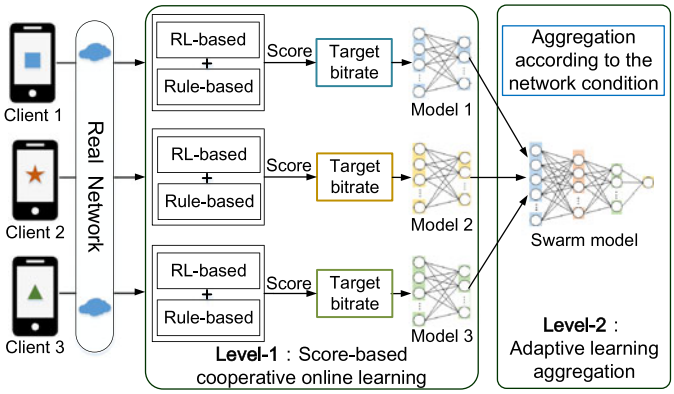


Fig. 2. An overview of *Legato's two-level online cooperative learning*.

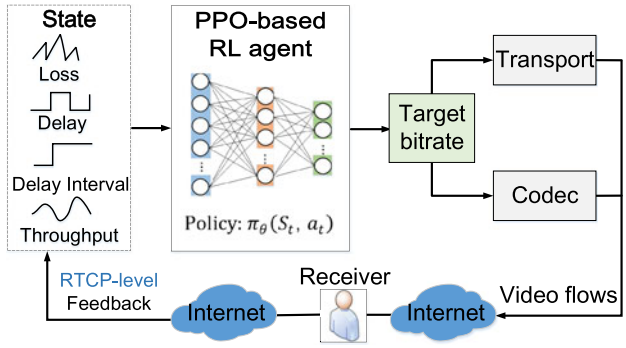


Fig. 3. The workflow of Legato's individual learning.

personal/temporal characteristics, and then replayed by a custom-built simulator/emulator to train a neural network model (usually an RL model). Finally, the trained model is distributed to each user, who executes the learned bitrate adaptation strategy at run time.

However, such offline learning cannot deliver expected performance improvement due to the simulation-to-reality gap as reported in [16], [17], [23] and also corroborated in Section 7.2. Moreover, the serialized learning aggravates the gap: *(i)* The uniform trace serialization blurs the users' individual characteristics, i.e., each user will get the same training experience, which loses the possibility of personalized optimization. *(ii)* Once the RL model is offline trained, it freezes and stops learning in the real world. Therefore, it is unable to deal with new network conditions unseen before.

In contrast, Legato adopts a fundamentally different design principle of learning at runtime by proposing *an iterative two-level cooperative learning architecture*. Fig. 2 illustrates Legato's design overview, which operates as follows, *(i) Level-1: individualized robust cooperative learning*. Each user separately learns its own RL model at run time, based on an RL algorithm [20] with customized design for real-time video transport (details in Section 3). In this way, each user possesses an individualized model incorporating its personalized learning experiences. Essentially, the RL's intrinsic trial-and-error mechanism is possible to cause inaccurate bitrate, thus results in severe QoE degradation. On the other hand, rule-based algorithms tend to make conservative decisions. To leverage the two policies' advantages, we propose a score-based cooperative fusion mechanism, so as to generate a co-determined bitrate that harnesses the two policies' behavior concurrently (details in Section 4). *(ii) Level-2: adaptive learning aggregation*. The individualized cooperative models will be fed to a backend server, and be classified into diverse network groups according to a customized network condition classification model. Then the models in the same group will be fused to generate a universal model following the principle of federated learning [24] (details in Section 5). From a high level, the aggregated model has experienced enough network variants from different users, and thus acquires the desired capability of coping with network dynamics.

We emphasize that the two stages operate iteratively. Specifically, the aggregated model will be distributed to its corresponding users, and each user, upon a new real-time video session, will start out with the latest specific aggregated model, and keep on learning from the new session. The procedure leads to a new individual model, which is fused at the back-end server for a new round of adaptive learning aggregation. The continuous iterations aim to strike a balance between each user's personalized experience and the swarm experience from all users. It is noteworthy that a mature aggregated model can be applied to a new user immediately, so as to eliminate the risk of low QoE due to random exploration at the bootstrapping phase. In the following sections, we proceed to elaborate on the individual online learning algorithm, the robust score-based cooperative learning, and the adaptive learning aggregation, respectively.

3 Individualized Online Learning

We design a PPO-based RL algorithm[2] for individualized online learning, as shown in Fig. 3. Specifically, the RL agent keeps observing the instantaneous network *state* S_t, i.e., packet loss, delay, delay interval (defined in Eq. (1)) and throughput at any continuous time-stamp t, and deciding an *action* a_t (i.e., a bitrate), which is expected to match currently available network bandwidth. Then the action a_t will be enforced on both the video codec and the transport layer protocol, which generates and consumes the video traffic at the rate a_t, respectively. The traffic, after going through the network path, will produce a new *state* S_{t+1} of next time-stamp, which initiates a new round of RL *action*.

The mapping of $S_t \rightarrow a_t$, the key function of RL agent, is decided by the RL's control policy π_{S_t,a_t}, which is learned in the above process. Intuitively, if the RL agent produces a bitrate exceeding the available bandwidth, it will incur network congestion. The consequence will be reflected in the next state S_{t+1}, most likely with high packet loss or large delay. By observing the change from S_t to S_{t+1}, the RL agent can realize that it has made an inappropriate action a_t, so it needs to update the policy π to generate a more conservative bitrate when observing S_t or similar state next time.

Underlying the workflow above, we identify two requirements: *(i)* The RL agent should be very agile, i.e., responding to network dynamics at video frame-level granularity, corresponding to a timescale of dozens of

2. PPO is a policy gradient optimization algorithm in the field of reinforcement learning, with the feature of easy convergence. Besides, it can guarantee that the policy update is not always too large, by controlling the latest policy is not very different from the old policy.

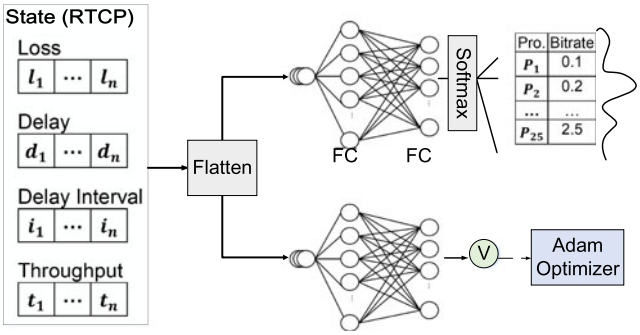


Fig. 4. Legato's RL model design.

milliseconds. *(ii)* The RL agent should refrain from jumpy actions which hurt video perception quality and QoE [25]. To meet the requirements, Legato first adopts a batch-level policy update running in the background at second level granularity, while executing agile response using the latest model at millisecond-level granularity. Second, in order to improve the smooth bitrate adaptation and avoid significant fluctuation between two consecutively updated bitrates, Legato employs a relatively stable policy adjustment mechanism and also designs to penalize large bitrate fluctuation in the reward design. In particular, the RL model will be saved together with its network condition information, and waiting for learning aggregation, which will be introduced in Section 5. We now present Legato's RL model and training methodology.

3.1 Legato's PPO Model

Fig. 4 depicts Legato's RL model, including state, action, and the neural network architecture.

State. The state, also the input of the RL agent, at any time t, is denoted with $S_t = (\vec{l_t}, \vec{d_t}, \vec{i_t}, t_t)$, which represents the packet loss, delay (represented by RTT, which is acquired on the WebRTC system), delay interval , and the receiver-side throughput. More specially, delay interval represents the difference between the arriving interval of two consecutive RTP packets at the receiver side and the corresponding departure interval at the sender side [26]. The formulation is represented as follows:

$$i_t = (T_{a_i} - T_{a_{i-1}}) - (T_{s_i} - T_{s_{i-1}}), \tag{1}$$

where T_{a_i} represents the arriving time of packet i, and T_{s_i} stands for the packet's departure time at the sender side.

These four inputs can be easily derived from the sender-side at a fine-grained timescale of around 50 ms, based on the periodic RTCP ACK message in WebRTC [27]–the transport/application layer protocol stack used in mainstream interactive video services.

Action. In each RTCP interval, Legato maps S_t to a compact action space $\mathcal{A} : \{0.5Mbps, 0.6Mbps, \cdots, 2.5Mbps\}$, representing the target output bitrate of video codec. After the video traffic is transmitted through the network, Legato will immediately transform to a new state S_{t+1}, and generate a new action a_{t+1}. Through such continuous iterations, Legato will learn to cope with network dynamics.

Reward. To ensure that Legato can learn from past experience, each action is associated with a reward, which is the video QoE of previous time interval defined as follows,

$$r_t = \alpha \times \sum_{n=1}^{N} q_n - \beta \times \sum_{n=1}^{N} l_n - \eta \times \sum_{n=1}^{N} d_n - \varphi \times \sum_{n=1}^{N-1} \left| q_n - q_{n-1} \right|, \tag{2}$$

where N represents the number of RTP packets in a state, q_n is the throughput measured at the receiver side which directly corresponds to received video resolution, e.g., 360P corresponds to the throughput region of 0.8Mbps∼1.2Mbps, 480P corresponds to the throughput region of 1.2Mbps∼1.5Mbps. l_n and d_n are the packet loss rate and delay at the transport layer. The final term is used to enforce video smoothness by penalizing large bitrate fluctuations. These metrics are weighted by different impacting factors $\alpha, \beta, \eta, \varphi$, and then summed together. These hyper-parameters have significant impacts on training effectiveness. Clearly, the four metrics have different magnitudes. The value of q_n and the final item usually falls within 0.1∼4.0 Mbps. l_n is commonly less than 10% in practical video systems, while d_n ranges from dozens of milliseconds to hundreds of milliseconds depending on the end-to-end path length. To obtain a meaningful r_t, these metrics need to be normalized to a consistent range by adjusting values of the parameters $\alpha, \beta, \eta, \varphi$. In Legato, the scope of the parameters is limited to 30∼100, 50∼100, 5∼25, and 10∼50, respectively. In the actual deployment of Legato, they are empirically set to 50, 50, 10, 30, respectively.

Neural Network (NN) Architecture. Legato uses a NN architecture to represent the policy π with a set of parameters θ. It adopts the simple fully-connected (FC) layer structures to distill implicit features hidden in different input elements. More specifically, Legato first flattens the sequences $(\vec{l_t}, \vec{d_t}, \vec{i_t}, t_t)$, and then feeds them into two similar NNs, one is used for feature extraction and then outputs the bitrate action, and the other is served to judge the overall objectives like a critic. Each NN contains two FC layers, with 64 and 32 elements, respectively. We have empirically verified that flattening the input sequences performs better in terms of bandwidth prediction accuracy than feeding each state to a separate NN structure. In addition, each layer employs the tanh activation function [28], which shows more accurate bandwidth prediction capabilities than traditional relu or softmax activation. Besides FC, we have also tried CNN and long-short-term-memory (LSTM) for feature extraction, but their performance is not as good. Further analysis shows that the CNN architecture specializes in extracting image features that consist of complex spatial information, which does not exist in Legato's state space. As for LSTM, it is more useful for reasoning the time-series data that takes into account long historic impact, but the performance of real-time video is more dependent on instantaneous network conditions.

3.2 Training Methodology

In the training stage of reinforcement learning, the cumulative reward is usually used to reflect the continuous action gains, which is represented as $\mathbb{R}_\tau = \sum_{t=t'}^{T_\vartheta} \gamma^{t-t'} r_t^\vartheta$, where T_ϑ stands for the batch size for gradient policy updates, t' represents the start time-stamp in certain batch-size, t stands for the followed continuous time-stamp, and we will compute each corresponding reward at time t as r_t^ϑ. γ serves as a discounting factor, usually customized as 0.99 or 0.9. The

summation means that more approaching reward r_t^{ϑ}, will have more critical impacts to the $\mathbb{R}_\tau$. The objective of Legato's training is to maximize $\mathbb{R}_\tau$. To achieve this, the gradient policy updates are used and can be formulated as follows:

$$\triangledown\bar{\mathbb{R}}_\tau = \frac{1}{\Theta}\sum_{\vartheta=1}^{\Theta}\sum_{t=1}^{T_\vartheta}(\hat{A}(a_t^{\vartheta}, s_t^{\vartheta}))\triangledown log p_\theta(a_t^{\vartheta}, s_t^{\vartheta}), \tag{3}$$

where Θ represents the number of batches. $\hat{A}(a_t^{\vartheta}, s_t^{\vartheta}) = \mathbb{R}_\tau - b$, (short as $\hat{A}$, $b \approx E[\mathbb{R}_\tau]$), represents the *advantage function* that expresses the difference in expected reward b decided by practical action from the state, compared with the averaged expected reward r from policy π. Intuitively, the advantage function can show the extra benefit of a certain action over the average action. $\triangledown log p_\theta(a_t^{\vartheta}, s_t^{\vartheta})$ stands for the policy updated gradients of the probabilistic mapping at $(a_t^{\vartheta}, s_t^{\vartheta})$. Recall that traditional policy gradient [29] may have large or little adjustment between contiguous policies, due to the uncertain learning rate value. This uncertainty can bring catastrophic QoE and difficult convergence in real-time video especially for the real-world learning of Legato. To avoid this, we design two customized modules to maintain the online learning performance:

(i) Batch-level updates instead of instance-level (a single input) updates. As noted before, Legato needs to respond to each input instance to adapt to the real-time bandwidth variation. Usually, the response also leads to an update of neural network parameters. However, such instance-level updates will largely increase the learning time and in turn slow down the response. We thus customize a *batch update policy*. In particular, the learning agent buffers the recent ⟨state, action, reward⟩ records. Only when the buffer is more than a batch-size, the agent will feed the buffer to the policy network to update the neural network parameters. In this way, the agent can execute fine-grained response in real time, while simultaneously running the online learning.

(ii) Smoothing-level updates. Legato utilizes the loss function $L(\theta)$ to compute policy gradient estimator, which is formulated as follows,

$$L(\theta) = \hat{\mathbb{E}}\left[min\left(\frac{p_{\theta_{new}}(s,a)}{p_{\theta_{old}}(s,a)}\hat{A}, clip(\frac{p_{\theta_{new}}(s,a)}{p_{\theta_{old}}(s,a)}, 1-\varepsilon, 1+\varepsilon)\hat{A}\right)\right]. \tag{4}$$

We bound the difference between a new policy and an old one, by taking into account their probability ratio $\frac{p_{\theta_{new}}(s,a)}{p_{\theta_{old}}(s,a)}$. Specifically, the first term of min is the gradients of $\frac{p_{\theta_{new}}(s,a)}{p_{\theta_{old}}(s,a)}\hat{A}$, and when $\frac{p_{\theta_{new}}(s,a)}{p_{\theta_{old}}(s,a)}$ is beyond the interval [1-ε, 1+ε], we clip the ratio and thus remove the incentive from the loss function $L(\theta)$. Then, the minimum of the clipped and unclipped objectives will be taken as the final loss objective. Here, ε is a hyper-parameter set to 0.2 [20]. In this way, Legato will update a learning policy smoothly, and in consequence avoids too jumpy bitrate decisions, so as to enhance bitrate smoothness for better QoE.

Neural Network Implementation Parameters. We use TensorFlow [30] to implement the training workflow and TFlearn [31] to construct the NN architecture. Legato takes the fine-grained exploration in RTCP-level, and the batch size for batch training is 32 in our design. It is noteworthy that we have tried the common practice of batch size adjustment, e.g., 16, 32 and 64, and did not observe substantial change on the QoE-related metric. We employ Adam optimizer [32] to update the stochastic gradient descent.

Enforcing RL Actions. A basic requirement for effective learning is that the learning algorithm's action should be faithfully executed. In our case, once the RL algorithm decides on a sending bitrate a_t, the video encoder should ideally produce the video stream at exactly the same rate. Though this is straightforward for offline simulation or VoD streaming, the video codec in real-time video transmission cannot generate a perfect bitrate of a_t particularly in short time-scales. Instead, the video codecs have their own processing logic depending on image scene dynamics, compression strategy and even device computing capacities, which results in inherent and substantial video bitrate fluctuation, and a gap between the RL's bitrate decision a_t and the actual video traffic output rate o_t, as demonstrated in [22]. Moreover, the score-based cooperative learning mechanism in Section 4 will also aggravate the gap. To handle the problem, we adapt Legato learning algorithm by feeding the gap $g_t = a_t - o_t$ into the RL neural model. In this way, Legato can learn the dynamics of the gap and then remedy the impacts by tuning its reward operation automatically. Once it detects a large gap (e.g., $|g_t| > 0.5$ Mbps), Legato will deem the previous RL action as corrupted, and reduce the action's impact by imposing a low weighting factor (default 0.5 in our design) in the cumulative reward (Section 3.2).

4 Score-Based Cooperative Learning

In this section, we first discuss the necessity of cooperative learning, and then describe the score-based cooperative learning policy to ensure the reliability of RL when it is trained online subject to real network dynamics.

4.1 The Necessity of Cooperative Learning

Essentially, typical RL models learn in a trial-and-error manner, whereas at run time, any erroneous action may lead to severe QoE degradation for the interactive video. We provide a showcase in Fig. 5. The video bitrate, packet delay, and ground truth network bandwidth are sampled from a controlled experiment on a local real-time video prototype system based on WebRTC framework (The system implementation will be introduced in Section 6). Within a period of 10 minutes, there exist 8 instances where the RL-generated bitrate overshoots the available bandwidth, causing latency spikes. Ideally, the RL algorithm should learn from these mistakes. But unlike simulation based training, accumulating experience through failures is unacceptable for real-time video. This is likely to happen particularly at the beginning of a video session, where the RL model may need to keep exploiting and failing, so as to materialize its policy.

On the other hand, traditional rule-based video transport control algorithms, such as Google Congestion Control (GCC) [26], have dramatically different behavior from learning-based controllers. In what follows, we give more details about GCC, which has been incorporated by Google Chrome and other well-known real-time communication applications[33].

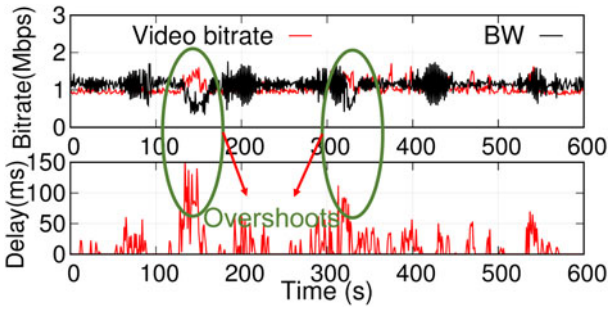


Fig. 5. Showcase of RL's bitrate overshooting.

GCC is built on UDP and RTP protocol, and compounded by two modules: The first is a delay-based controller at the receiver, which computes a delay-related bitrate based on an network over-use detector. The second one is a loss-based controller, with decisions made at the sender by tracing the loss variances. The final decision is co-determined by the hybrid loss-and-delay controllers, and tends to make conservative decisions [26]. So a straightforward solution that ensures the reliability of online learning can be combining the rule-based algorithm and the RL algorithm, i.e., switching to a conservative algorithm once RL's decision becomes too aggressive, as proposed in OnRL [22]. However, we find that such simple switching policy in OnRL algorithm is not safe enough in practical video system. In particular, we identify two key limitations: *(i)* Whereas the RL actor can detect when to switch to the safe policy, it is hard to determine when to roll back to the RL policy. Moreover, we find that the rollback often happens before RL has effectively learned the switch-penalty proposed in OnRL, so the aggressive behavior of RL still has a high possibility to occur, which will be validated in Section 7.4. *(ii)* The learning-based actor and rule-based actors follow an "either-or" principle. It is possible that neither of their decisions is optimal for underlying network conditions.

To solve the above limitations, we design a *score-based cooperative learning mechanism* that coordinates the wild RL and conservative rule-based algorithm in a coupled close-loop. Intuitively, we do not merely regard the rule-based algorithm (i.e., GCC in our design) as a protective backup, but treat it as the same important policy with the RL algorithms. The two policies work concurrently in the video system, and decide on their own bitrate separately. Then the two decisions will be associated by a dynamic score-based blend. At a high level, the score-based policy observes the two actors' behaviors, and enables judging which actor has a larger possibility to generate a more appropriate bitrate for the instantaneous network bandwidth, then will endow it a higher scoring value (e.g., 0.8), and give another a relatively lower value (e.g., 0.2), then generating a cooperatively blended bitrate. In this way, the final bitrate represents the optimal decision co-determined by the two actors, then it will be used by the system to best match the instantaneous network condition variation.

4.2 Score-Based Cooperative Learning Design

In this section, we first describe the overview of score-based cooperative learning design, then describe the detailed design and training methodology.

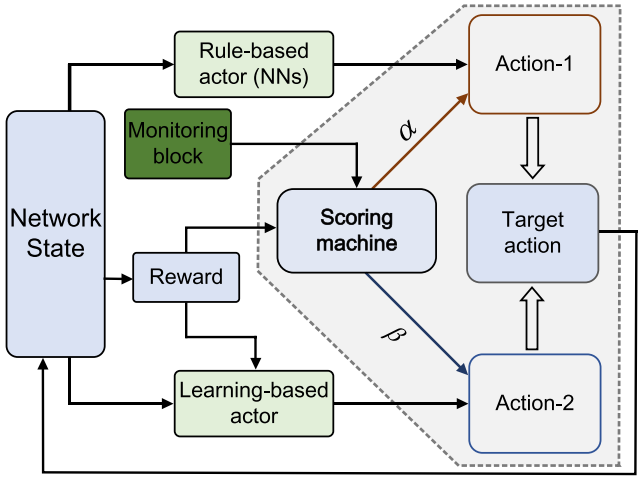


Fig. 6. Legato's score-based cooperative learning mechanism.

4.2.1 Score-Based Cooperative Learning Workflow

Fig. 6 illustrates the workflow of score-based cooperative learning mechanism. Specifically, *(i)* The rule-based GCC and the learning-based PPO work separately: they simultaneously observe the instantaneous network state and generate their own actions, i.e., bitrate decision. *(ii)* Neither of the two bitrate actions will be directly used by the video system. Instead, the two actions will be combined by a dynamic scoring machine, which is composed of a learning-based DDPG algorithm. The scoring machine will give its score decision to the two policies, e.g., α for rule-based GCC's action, and β for learning-based PPO's decision. After a simple linear integration between the two actions and their corresponding score value, it will generate a final bitrate that makes effect for the video system. The video traffic, after going through the network path, will produce a new network state, which initiates a new round of score-based blended action.

4.2.2 Learning-Based Dynamic Scoring Machine Design

To realize effective and robust scoring mechanisms, we model the scoring principle as a dynamic learning task, and the mechanism is built on top of a reinforcement learning algorithm named DDPG [21], a variant of deterministic policy gradient algorithm, which is designed for learning continuous action value efficiently. Different from the probability distribution-based decision-making, (e.g., PPO), DDPG aims at a direct action value, which suits better for our score-based tasks (e.g., generating a deterministic action). In what follows, we describe the design modules and details of Legato's scoring mechanism.

Monitoring Block. Note that the DDPG algorithm is based on the actor-critic principle, so we custom-design their inputs as a monitoring block to adapt with the domain-knowledge in our online learning design of real-time video transmission. More concretely, for the input of actor network, at any time t, is denoted with $I_{A_t} = (\bar{l}_t, \bar{d}_t, \bar{i}_t)$, which represents the averaged packet loss, averaged delay, and averaged receiver-side throughput in each monitoring block (e.g., an RTCP interval). For the input of critic network, at

any time t, is denoted with $I_{C_t} = (\bar{l}_t, \bar{d}_t, \bar{i}_t, O_{1t})$. The O_{1t} represents the output (i.e., action) of the actor network, as we show below. Similar to the state of RL actor model, these inputs can be easily derived at the sender side at a fine-grained timescale.

Output. The scoring machine outputs two values, represented as O_{1t} and O_{2t}, respectively, of which correspond to α, β in Fig. 6. O_{1t} and O_{2t} can be formulated into the range of [0, 1] as follows,

$$O_{1t} = 0.5 + 0.5 \times tanh(NNs(I_t)), O_{2t} = 1 - O_{1t}. \quad (5)$$

Note that both of O_{1t} and O_{2t} are float values. O_{1t} determines the score value to RL actor, and correspondingly, O_{2t} stands for the scoring to GCC actor.

Policy Model. The actor network represents the policy network, denoted as Θ^{μ}, which is composed of two consecutive FC layer, regulated by the tanh activation function. The critic network, denoted as Θ^{τ}, will first feed the list $\{\bar{l}_t, \bar{d}_t, \bar{i}_t\}$ into a FC layer, and O_{1t} into another FC layer, separately. Then their corresponding output layer will be concatenated, fed into a final FC layer, and output a vector as the critic value. We emphasize that both the actor Θ^{μ} and critic network Θ^{τ} have their structure counterparts for DDPG algorithm's neural function approximator design.

Memory Replay. The common challenge in training deep neural networks is catastrophic forgetting [34], i.e., forgetting previous knowledge with learning new knowledge from unseen environment. To solve such problem and fully utilize the history knowledge, we utilize the memory replay mechanism to train the scoring machine in an *off-policy* manner. Specifically, we maintain a global memory buffer M, which is composed of the history list of 4-tuple $< I_A, O_{1t}, r, I_{A_{next}} >$. During each training iteration, we randomly select these tuples from the buffer to form a mini-batch sample as training data of actor-critic network, and the newly generated tuples will be stored into the buffer waited for a new round of samplings. We empirically set that the memory buffer M maintains the length of $5e^4$, and the mini-batch size as 64.

Training Detail. It's noteworthy that the RL-model and the score-model are integrated into one entity following the online learning design with video sessions evolving. They share an overall reward function in Eq. (2). In particular, the training effectiveness of RL-model is guaranteed in Eq. (4), and the scoring machine is regulated by the classic TD loss function [35]. More training methodology formulations are detailed in [21].

5 Adaptive Learning Aggregation

The learning aggregation is inspired by two key observations from our analysis from real-world video sessions on e-commerce live video broadcasting: *(i)* Different users exhibit different levels of network dynamics. For instance, some users typically initialize video sessions at home (e.g., make-up tutors or in-store shopping guides), with stable WiFi connections. Correspondingly, Legato's learning algorithm tends to generate stable video bitrate decisions. In contrast, other client users (e.g., outdoor travelers) usually undergo fluctuant cellular network conditions, due to mobility or handoff. Accordingly, the learning algorithm will learn to change its video bitrates frequently to match the instantaneous network variations. *(ii)* While a user usually has relatively consistent network conditions, she may change the daily routines, thus encountering new network dynamics. In this case, an RL model purely trained from her own previous network conditions will react inappropriately.

To strike a balance between individual and swarm experience, we propose an adaptive learning aggregation method. In order to design an effective aggregation policy, we first present an elaborated analysis to demystify the correlation between various network conditions and video QoE (Section 5.1). Then we design a network condition discriminator and fine-grained network condition-aware learning aggregation algorithm based on federated learning principle (Section 5.2, Section 5.3).

5.1 Revealing the Correlation Between Network Conditions and Video QoE

We have conducted a trace-driven study to profile the relationships between network conditions and video QoE. To make the analysis comprehensive, we select about 2-thousand network traces from a commercial real-time video vendor. Each trace lasts about 35 minutes, and the overall traces are about 1166 hours in total. The traces contain concrete network conditions, and their corresponding multiple QoE metrics. We select 5 metrics that have explicit/implicit impacts on Legato model's inputs and QoE, e.g., the input metrics, loss rate, RTT, and video throughput, and the QoE-related metrics, e.g., fps and frame jitter delay. Based on these metrics, we cluster different network conditions that share with critical features, e.g.,

(i) The Internet Service Providers (ISP), e.g., the AT&T, Verizon in US, and CUCC, CTCC, in China. They are usually equipped with diverse network control modes and base stations (BS), and have the probabilities to impact user experience. In Legato, we select three widely-used ISPs existing in our video traces, and anonymize them into ISP-1, ISP-2, and ISP-3, due to collaborated issues.

(ii) The network types, currently dominated are WiFi (mostly indoor stable environments) and 4G (usually outdoor fluctuate environments).

(iii) The network quality, e.g., dynamic (denoted as D) versus stable (denoted as S) versus low-quality (denoted as L) versus high-quality (denoted as H) network. In particular, we classify them into 4 kinds of typical groups, i.e., H&S, H&D, L&S, L&D, separately (More descriptions in Section 5.2).

We plot the CDF of above the 5 metrics and their different clustering conditions over the 2-thousand traces in Fig. 7. According to these CDFs, we have three key findings: *(i)* The QoE metrics exhibit less sensitiveness to the ISP information, as shown in Fig. 7a. In particular, ISP-2 and ISP-3 have almost the same data distribution across all the 5 metrics. Compared with them, ISP-1 provider shows lower loss rate and RTT, but their averages are very close, e.g., 0.013% differences in loss rate and 0.12 seconds in RTT compared with ISP-2. *(ii)* We observe that the two mainstream

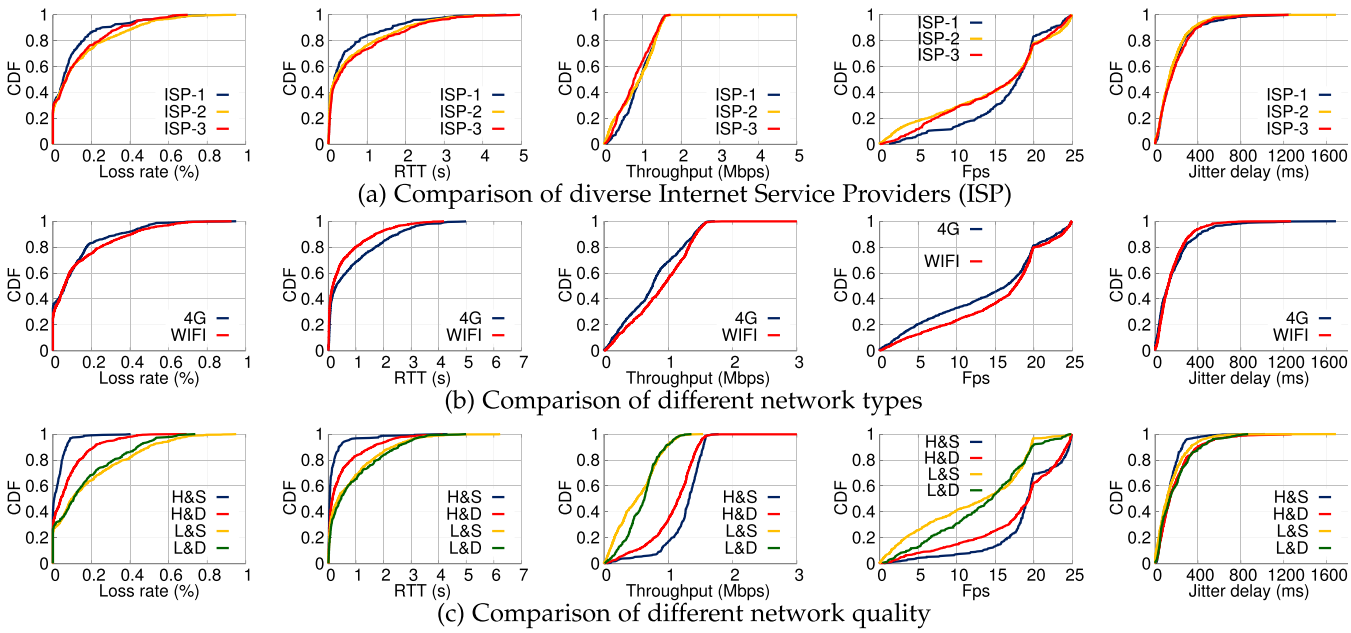


Fig. 7. The fine-grained QoE metrics comparisons over two thousand real-time video session traces.

network types, i.e., 4G and WiFi, show much difference in the video quality and stall events (Details in Fig. 7b). For instance, the averaged RTT and frame jitter of 4G are remarkably 53.40%, 13.36% higher than that of WiFi separately. Meanwhile, for the video quality, e.g., the averaged receiving throughput and fps of 4G are 12.64%, 9.92% lower than WiFi, separately. Accordingly, we conjecture that the network type is much related to the QoE. *(iii)* Moreover, as illustrated in Fig. 7c, we find that the QoE metrics are also significantly relevant to the instantaneous network quality. For instance, the network in high and stable (H&S) quality maintains the lowest loss rate (0.027%), RTT (0.18s), and frame jitter (133.47ms), while has the highest video throughput (1.25 Mbps) and fps (18.8). In contrast, the network equipped with the worst situation (e.g., L&D) undergoes the largest RTT (0.89s), and frame jitter (189.23ms), which are corresponding to the worst stalling events. These results clearly indicate that network quality is more likely to affect the user QoE.

Finding Summary: Overall, we observe that the network type and quality have key impacts on Legato model's input signal and video QoE. Thus we will design a fine-grained aggregation policy based on the two key network features.

5.2 Network Condition-Aware Aggregation Design

We illustrate the workflow of Legato's adaptive learning aggregation in Fig. 8. Building on these data-driven findings, we firstly design a *network condition discriminator* to identify each user's network conditions, including the network type (e.g., 4G, WiFi) and network quality (Stable or dynamic, low-quality or high-quality). Especially, the network type (e.g., 4G, WiFi, or the emerging 5G) can be directly recorded by the app provider. As we know, before the video session starts, we cannot know the practical network quality. Thus, in order to obtain each user session's such quality, we utilize the default rule-based algorithms (e.g., GCC) as a probing attempt. More specifically, we use the GCC for the first 1 minute, and log the millisecond-level sending throughput traces. Note that the 1 minute records are sufficient for network condition identifying. Then we divide the network quality into stable/dynamic/low-quality/high-quality according to the averaged throughput Th_{avg} and its variance Th_{std}, in comparison with the threshold margin detailed in Table 1. If one session's averaged throughput is larger than the mean of all the sessions, we regard the corresponding network quality as *high-quality*, and *low-quality* otherwise. Correspondingly, if one session's throughput std. is lower than the mean of all sessions, we deem it as *stable* network, and *dynamic* otherwise.

After such a classification scheme, we identity 8 groups of models in total, as detailed in Table 1. Then we will aggregate models in the same groups (labeled as G). Recent works [24], [36] have shown that, averaging the parameters of the same neuron across many NN models can achieve similar effects of aggregating these models' experience. We thus utilize the strong feature representation ability of NN parameters to realize the model aggregation, as shown in Fig. 8.

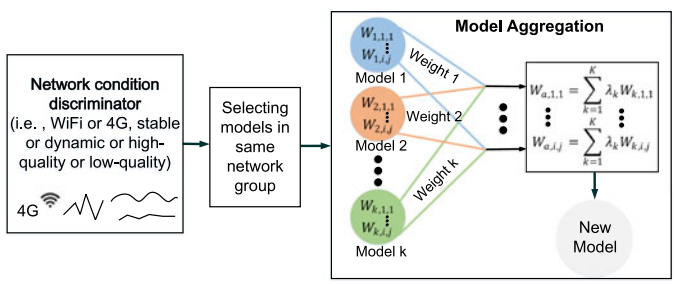


Fig. 8. The workflow of Legato's adaptive learning aggregation.

TABLE 1
Legato Classifies Network Conditions Into Eight Groups

Group-1:	4G & H ($th_{avg} > 1.27$ Mbps) & S ($th_{std} < 0.18$ Mbps)	Group-2:	WiFi & H ($th_{avg} > 1.27$ Mbps) & S ($th_{std} < 0.18$ Mbps)
Group-3:	4G & H ($th_{avg} > 1.27$ Mbps) & D ($th_{std} > 0.18$ Mbps)	Group-4:	WiFi & H ($th_{avg} > 1.27$ Mbps) & D ($th_{std} > 0.18$ Mbps)
Group-5:	4G & L ($th_{avg} < 1.27$ Mbps) & S ($th_{std} < 0.18$ Mbps)	Group-6:	WiFi & L ($th_{avg} < 1.27$ Mbps) & S ($th_{std} < 0.18$ Mbps)
Group-7:	4G & L ($th_{avg} < 1.27$ Mbps) & D ($th_{std} > 0.18$ Mbps)	Group-8:	WiFi & L ($th_{avg} < 1.27$ Mbps) & D ($th_{std} > 0.18$ Mbps)

Specifically, suppose we have a total number of K users in one kind of group G_{th}, each user k's neural model can be denoted with a matrix $W_{k,i,j}$, where each element of $W_{k,i,j}$ is the parameter of a neuron in the neural model (i represents the i-th layer of the NN, and j is the j_{th} neural cell in each layer). Note that the dimensions of all matrices $W_{k,i,j}(k \in [1, K])$ are the same, since the NN architecture across all users is the same. To fuse the individual models, we perform a weighted averaging operation following the principle of federated learning [24]:

$$\overline{W} = \sum_{k=1}^{K} \sum_{i=1}^{I} \sum_{j=1}^{J} \lambda_k W_{k,i,j} \quad w.r.t. \quad K \in G_{th}, \tag{6}$$

where $\overline{W}$ represents the aggregated model, and λ_k is the weight of user k's model. We provide an example in Fig. 9 to illustrate the model averaging process. In particular, the weights $\overline{W}_{2,2}$ of the dark neural cell is computed from the corresponding neuron position of two individual models: model-1 and model-2.

We have two ways to determine the value of each λ_k:

(i) Average aggregation. For a newly joined user (i.e., a user without any prior experience), we generate and apply a model with average experience from all users, i.e., we let $\lambda_k = \frac{1}{K}$ for $k \in [1, K]$.

(ii) Prioritized aggregation. For other users, we prioritize a user's own weight thus it can achieve optimal performance under its typical network conditions, while reacting appropriately when confronting unusual conditions. Specifically, for each user k, we let $\lambda_k = p$ ($p \in [0, 1]$), and $\lambda_m = \frac{1-p}{K-1}, \forall m \; / = k$. In Section 7, we experimentally evaluate the impact of weight parameter of p.

5.3 Model Identifying and Utilization

Now, we have 8 kinds of learning models after the learning aggregation, these models will be ready for distributing to their corresponding users. For a user who starts a new video session, we will first identify its network conditions based on the discriminator. Once the network condition is generalized to one of the 8 models, the user will start out with its specific model, and keep on learning from the new session. We summarize the overall aggregate algorithm and workflow in Algorithm 1.

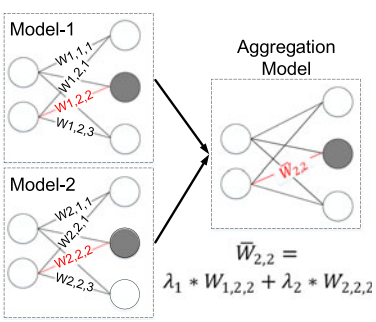


Fig. 9. A showcase of the weighted models.

6 Implementation

Real-Time Video System Implementation. We implement Legato based on WebRTC [27], a real-time video communication framework with built-in support of video codec and transport-layer protocol (i.e., GCC). WebRTC allows flexible re-implementation of the video control algorithms, and has been used in state-of-the-art transport and video application studies, such as BBR [37], Salsify [38], Concerto [15], and OnRL [22], *etc.*

Algorithm 1. Adaptive Learning Aggregation Design

Step 1: Clustering different user models into 8 groups:
(i) network type: WiFi or 4G;
(ii) network quality: stable or dynamic, low-quality or high-quality;
Step 2: Network condition-aware aggregation:
if *in the same group* G_i **then**
Aggregating all the models in the i_{th} groups via two ways:
(i) average aggregation
(ii) prioritized aggregation
end
Step 3: Model identifying and utilization:
Identifying the user's network condition by using the *Network condition discriminator*.
if *the same group* G_i **then**
Load the i_{th} model and keep on learning
end

Our Legato implementation essentially replaces the existing bitrate control module in WebRTC. Ideally, Legato's components should be implemented inside the mobile WebRTC-based app due to currently dominated mobile devices. However, due to the lack of API support for training RL neural networks on mobile devices [39], we implement Legato via a cloud-assisted framework, as shown in Fig. 10. Besides the pair of real-time video's sender and receiver, we introduce an RL server, on which we implement the three key design components of Legato (i.e., the individualized online learning, two-level iterative learning) based on Tensorflow [30].

During each real-time video session, the sender maintains a connection and exchanges information with the RL

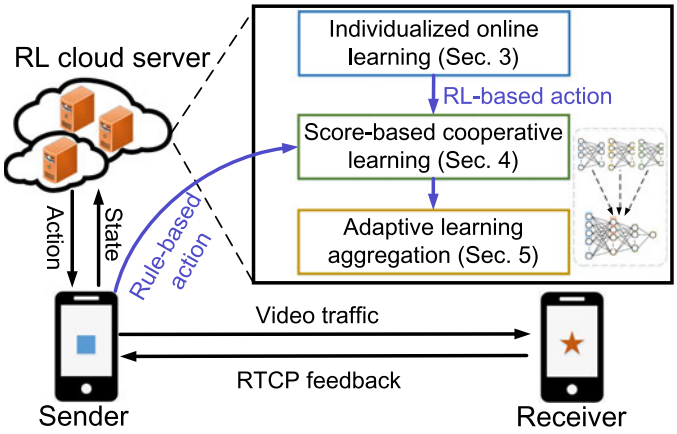


Fig. 10. System implementation of Legato.

server. It collects RTCP feedbacks (i.e., packet loss, delay, throughput, *etc.*) from the receiver and sends them to the RL server as the input of Legato. Then Legato processes the input and the rule-based actions, then returns an action (i.e., the target video bitrate determined by Legato's score-based cooperative mechanism) to the sender, which is further executed by the underlying video system. Meanwhile, the Legato module on the server periodically updates its control policy to realize online learning. To ensure the low delay between the sender and RL server, we aim to let the two as closely as possible, such as prioritizing to make the sender connect the RL server in the same city. Under such configuration, we found that the information exchange latency from the sender to the RL server is only in few to dozens of millisecond-level intervals, which has ignorable impacts on video communications as will be demonstrated in Section 7.3. The end-to-end network bandwidth between the sender and receiver follows the throughput traces[3] that we collect from a real-world commercial video sessions and are enforced by Linux tc [40]. Our platforms allow for controlled experiments with known bandwidth ground-truth, which enable facilitating deeper diagnosis of the experimental results.

Tackling Extreme Events. To handle some extreme events (e.g., possible connection failure between the sender and RL server, we also implement a fallback mechanism on the sender, i.e., the sender automatically downgrades to the default GCC controller. In addition, we deploy a separate back-end server to perform fine-grained network condition-aware learning aggregation at the frequency of once per week (typically in the early morning with the least user activity).

RL Server. Each RL server is a COTS PC equipped with 12 cores, 800-MHz CPU and 16-GB memory, and runs in Ubuntu Linux 16.04. We adopt the Tensorflow version 1.15.2 to host Legato's neural network, and each server can accommodate at least 40 concurrent users, due to efficient neural network structure of Legato with a small size of 495.2 KB and runtime memory requirement ranging of 0.256-0.272 GB. Currently, we have deployed 3 RL servers for our testing.

3. We note that network bandwidth is difficult to acquire in real-world video streaming application, thus we use the fine-grained (second-level) throughput traces as the varying available bandwidth in our testbed experiments. Although the throughput value may not be equal to the actual available bandwidth or usually lower than the actual bandwidth, it possesses general tendencies of network dynamic nature [15].

7 Evaluation

In this section, we evaluate Legato from 3 aspects: *(i)* We validate the necessity of online learning by comparing the performance of models under different algorithms and operating environments including simulation, testbed and operational interactive video application (Section 7.2). *(ii)* We evaluate Legato in the real-time interactive video prototype system through a corpus of real-world video traces (Section 7.3). *(iii)* We evaluate each design module inside Legato separately to gain an in-depth microscopic understanding of Legato's two-level online cooperative learning (Section 7.4).

7.1 Methodology

Evaluation Metrics. We employ both application-layer and transport-layer metrics for a comprehensive evaluation, including:

(i) Video stall-related metrics: stall rate to measure the fluency of the interactive video session. In practical real-time video system, a stalling happens whenever the packet RTT exceeds 300ms [22], [41], consistent with the evaluation metrics in real-time interactive video applications. Here we remark that the stall rate indicates how frequently video freezing occurs, which is different from stalling duration. In the simulator/testbed environments, the RTT hardly rises to 300ms, so a stalling event is considered to occur when the instantaneous video frame rate drops below 12 fps, as suggested by the DevOps engineers of an operational interactive video system. The metric has also been adopted in state-of-the-art works[22], [42].

(ii) Video quality-related metrics: video bitrate, received throughput and averaged fps (frames per second) that characterize the perceived quality of the video frames. Note that our experiments are performed in both controlled prototype testbed and the operational real-time video system.

(iii) We also examine the transport layer metrics including packet loss rate and RTT during video transmission, which helps to understand Legato's behaviors in-depth and explain the application-layer performance.

Baseline Algorithms. We compare Legato against the following algorithms which represent the state-of-the-art in real-time interactive video transmission: *(i)* Concerto [15], which utilizes deep imitation learning [43] to optimize video QoE. As a semi-supervised algorithm, Concerto is trained in a simulator with known ground-truth network bandwidth. The trained model is then deployed and tested in real applications. In this work, we run Concerto using its original model [15] which is trained over 1 million real-world real-time video sessions unless otherwise stated. *(ii)* OnRL [22], which proposes an online reinforcement learning method to optimize real-time video QoE. It makes the learning process in the real video system instead of simulation, so as to close the "simulation-to-reality" gap.

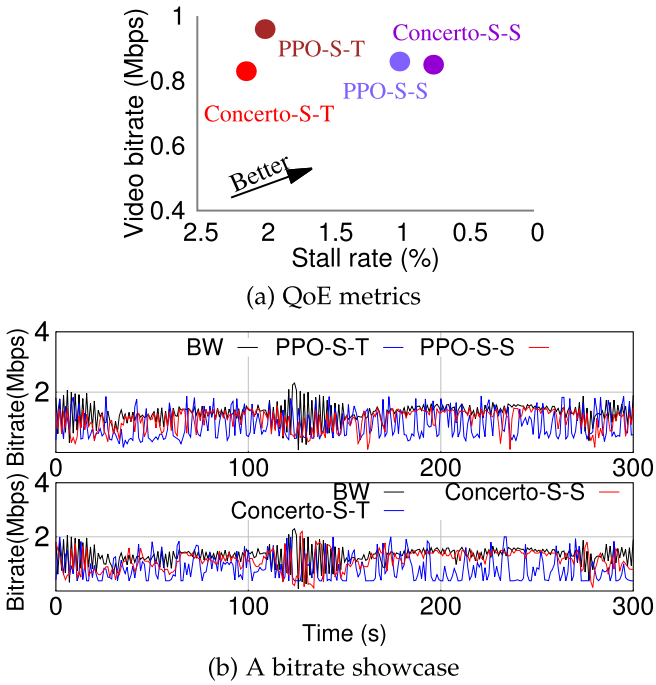


Fig. 11. Performance of simulation-trained models under simulator and testbed, respectively.

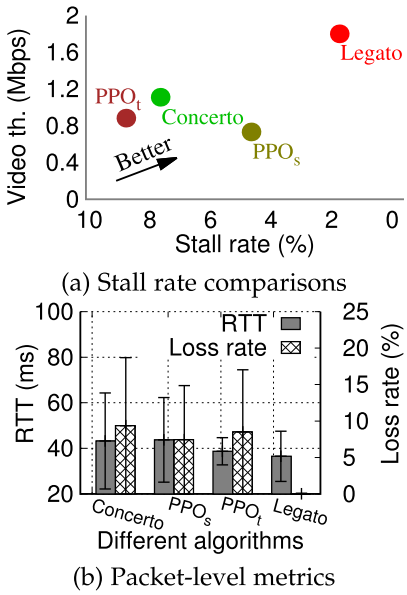


Fig. 12. Performance comparison of different models running over an operational interactive video app.

7.2 The Need for Online Learning

The Gap Between Simulation and Testbed. In order to validate the limitations of "offline learning", we first run a micro-benchmark to compare the performance of simulation-trained models, i.e., Concerto and the basic PPO model described in Section 3. To ensure fairness, we use the same 3-hour trace in two different testing environments: the simulator and local testbed. The resulting video bitrates and stall rates are presented in Fig. 11a, which confirms that Concerto and PPO work well when the testing is done in the same environment as the training (i.e., the cases marked as Concerto-S-S and PPO-S-S). However, the performance drops significantly after the testing environment is changed to be the testbed (i.e., Concerto-S-T and PPO-S-T). Specifically, their stall rates increase by 64.9% and 51.2% respectively, with minor video bitrate deviation (between 0.02 Mbps and 0.1 Mbps). These results clearly validate the necessity of online learning. To understand the performance drop, we plot a 300-second segment of the models' actions (i.e., video bitrates) against the ground-truth bandwidth in Fig. 11b. We observe that both Concerto and PPO in simulation environment can follow the bandwidth very closely despite high dynamics, with average deviations of only 17.4% and 13.6%, respectively. In contrast, the deviation increases to 30.3% and 37.12%, when coping with the same traces in the testbed environment. *To summarize, once the deployment environment differs from the training environment, the offline RL models' experience becomes stale, leading to unsatisfactory performance.*

The Gap Between Simulation/Testbed and Real-World Network Conditions. We further examine the limitations of "offline learning" under real-world network dynamics. In addition to Legato, we intentionally integrate three offline-trained models into a commercial real-time video system, including the Concerto and PPO models trained in simulator (referred to as PPO_s), and the PPO model trained in testbed (referred to as PPO_t). Note that the offline-trained models all showed satisfactory video QoE in their own training environment as validated above. We then run the 4 models on a randomly selected app user. In particular, each model is run over 10 video sessions in the practical system, under fluctuant network conditions. We plot the average video throughput, stall rate, packet RTT and loss rate in Fig. 12, over a random set of video sessions lasting 10 hours in total. We find that Legato achieves the most compelling video QoE. In particular, it outperforms the most competent offline-trained scheme Concerto by 31.9% in terms of video throughput, and leads to a remarkable 78.3% reduction in stall rate, which again corroborates the benefits of online learning mechanism. On the other hand, PPO_t and PPO_s exhibit large QoE gaps on both metrics. Accordingly, Legato also exhibits the smallest packet delays and loss rates, which confirms its ability to cope with network dynamics. *The result further validates the necessity of online learning for optimizing real-time interactive video applications under real network conditions.*

7.3 System-Level Evaluation in a Real-Time Video System

We distribute Legato to real-world users for a system-level evaluation. In particular, we have recruited 5 users, and collected over 200 real-time video traces from them. In each session trace, we log the network/application performance metrics at second-level granularity.

Overall Performance. We compare Legato with the state-of-the-art real-time video QoE optimization algorithm OnRL. Table 2 summarizes the QoE metrics averaged over long time video sessions (e.g., 5 hours). We observe that *(i)* Legato achieves a remarkable reduction in terms of video stall-related metrics, i.e., reducing stall rate by 8.29%, decreasing the low fps frequency by 43.90% than OnRL.

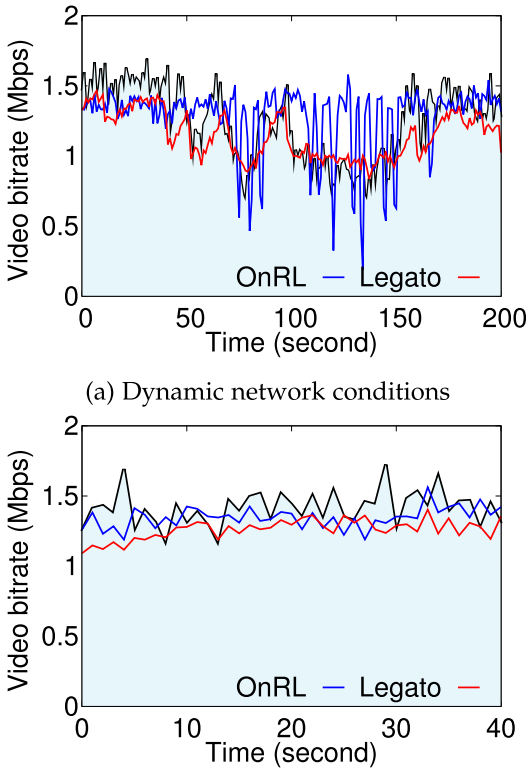
TABLE 2
Average QoE Statistics from Real-Time Interactive Video System

Algorithms	Video stall-related metrics				Video quality-related metrics	
	Stall rate (%)	Low fps frequency (%)	Frame Jitter (ms)	Frame delay (ms)	Fps	Video throughput (Mbps)
OnRL	3.98	2.05	117.11	39.70	27.40	1.06
Legato	3.65	1.15	101.02	36.44	28.09	1.13

Meanwhile, the frame delay, and frame jitter also drop 13.74%, 8.21%, separately. *(ii)* Meanwhile, Legato also achieves remarkable improvement in terms of video quality metrics: 2.52% higher fps, and 6.60% higher video throughput. These gains clearly validate the algorithm effectiveness of Legato. Below we provide a more in-depth analysis of the results.

Performance Breakdown and Showcase. We select two typical network conditions: the fluctuant and stable traces, and plot their showcases in Fig. 13. We find that *(i)* OnRL algorithm cannot deal well with the highly dynamic network conditions. For instance, at about 50s of Fig. 13a, OnRL can realize its overshooting behavior with its safety detection mechanism, and switch to the safe policy. However, OnRL rolls back to the RL policy earlier before learning well enough, so the overshooting events frequently happen during 50s-150s, and thus will cause a high loss rate and stalling events, which corroborates the limitations of OnRL's simple "either-or" switch operation. In contrast, Legato has the abilities to explore and closely track the available bandwidth under its effective score-based cooperative decision mechanism. *(ii)* In the stable network as illustrated in Fig. 13b, Legato shows a relatively safer and smoother bitrate estimation than OnRL, e.g., Legato's bitrate *std.* is 3.9% lower than OnRL. The bitrate smoothness will help to improve the video quality fluency, e.g., persistent 720P instead of continually switching between 720P and 480P.

Comparing Legato With Salsify. We further add an experiment to compare Legato with Salsify, a typical rule-based and fine-grained bitrate coding scheme for real-time video transmission [38]. We run both algorithms over 4 randomly selected traces from the real-time video provider, totally lasting about 4 hours. We plot their two QoE-related metrics (i.e., frame delay and video bitrate) in Fig. 14. We can observe that Legato achieves consistently better QoE than Salsify, not only the frame delay is decreased by 45.26%, but also the video bitrate is improved by 34.18%. The gains of Legato stem from it enables adaptively adjusting its bitrate to cope with the dynamic network conditions, thus seldom cause bitrate overshooting behaviors. In contrast, Salsify conducts fast bitrate adjustment to follow up the video codec, doesn't deeply consider the transport layer information, which will degrade the bitrate robustness and thus impact video QoE. These results also corroborate the effectiveness of Legato's online learning strategies.



(a) Dynamic network conditions

(b) Stable network conditions

Fig. 13. Showcase comparison of OnRL and Legato.

7.4 Detailed Analysis of Legato Pipeline

We now conduct micro-benchmark experiments to study the impact of each design module inside Legato separately.

Understanding of Legato's Scoring Mechanism. We now demystify the behavior of Legato's score-based cooperative mechanism, in order to deeply understand its robust decisions against various network conditions. We visualize Legato's blended decision, its two actors' decisions, and their corresponding score values, separately, in Fig. 15. We make the following observations: *(i)* When the RL-based actor aggressively explores the instantaneous bandwidth and causes continuously over-estimating, e.g., during the period M_1 (5s-30s), and M_3 (110s-130s) in the green line, the dynamic score machine will prioritize GCC (e.g., the yellow line), the one who has the under-use behavior. After the appropriate score integration, e.g., the score value of GCC is

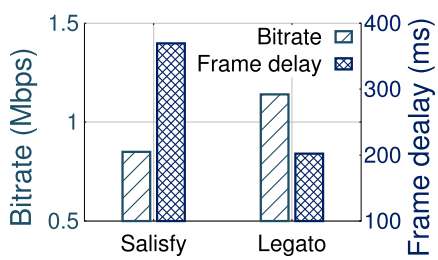


Fig. 14. Legato achieves better QoE than Salsify.

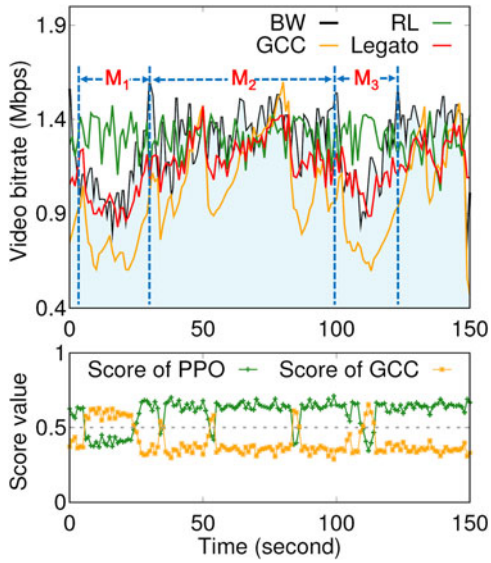


Fig. 15. Understanding of Legato's score mechanism.

relatively higher than PPO, the blended bitrate (the red line) can better follow the bandwidth variation, thus achieving compelling QoE. *(ii)* In contrast, during the period M_2 of 30s-110s, when the RL actor enables exploring the bandwidth appropriately, GCC-based actor still shows the "AIMD-like" cautious natures; whereas the scoring machine will automatically be inclined to the RL-based actor by generating larger score weights for itself. These adaptive score values are automatically controlled by the DDPG-based design mechanism as detailed in Section 4.

The Gain From Learning Aggregation. We first demonstrate the gain of exploiting prior learning experiences then continuously learning, by comparing the performance of an aggregated model (*agg-model*) against the one that starts from scratch (*scr-model*). More specifically, we train 8 separate models from randomly selected interactive video session traces, each lasting longer than 2 hours. Then we aggregate them through averaging as specified in Section 5.2. Fig. 16a plots the video throughput and fps, when running agg-model and scr-model over the same set of new traces. We find that, *agg-model* outperforms scr-model on the two important QoE-related metrics: throughput and fps gains are 18.2%, and 47.6%, respectively. To further understand above results, we analyze the two models' actions (i.e., the video sending bitrates) in Fig. 16b. We see

TABLE 3
Verification of Network Condition-Aware Learning Aggregation

Models	Metrics	Test (stable trace)	Test (dynamic trace)
Swarm-stable	Fps	28.91	27.55
	Stall (%)	0.77	1.83
Swarm-dynamic	Fps	26.28	27.85
	Stall (%)	3.01	1.84

that, *agg-model* enables following the network bandwidth at the beginning of the session because the aggregation model works to respond with the network, and then it can real-time probe the bandwidth with the model evolving within its own network experience, and generate adaptive bitrate in real-time. In contrast, *scr-model* usually sends more traffic than the available bandwidth, which incurs congestion and thus low QoE. Moreover, the actions of *scr-model* exhibit 47.6% higher variance than that of the *agg-model*, indicating it struggles to explore the available bandwidth at the cost of overshooting/underuse. Overall, the comparing results show that the learning aggregation mechanism can help Legato to adapt the network bandwidth at the beginning of this session, then the online learning will generate an individualized RL model with its own learning experience and can cope with its network dynamics effectively.

Verification of Network Condition-Aware Learning Aggregation. We train 4 models using 4 different traces, two are in diverse stable network conditions, and another two are relatively dynamic traces. Note that the network condition discrimination follows the criterion in Section 5.2. We aggregate the two stable models into a *swarm-stable* model, and aggregate the two dynamic ones to form a *swarm-dynamic* one, separately. Then we conduct cross-validation, i.e., using the *swarm-stable* model and *swarm-dynamic* model to test a randomly selected stable trace and a dynamic trace, separately. Afterwards, we summarize the key QoE results of the 4 cross-testing in Table 3, from which, we observe that the swarm model is capable of performing well when its testing environment is in the same network conditions with its aggregated traces. For instance, *swarm-stable* model shows 4.93% higher averaged fps and 57.92% decreased stall rate when testing in the stable traces, than the dynamic ones. Similarly, the *swarm-dynamic* model achieves 5.63% higher fps and 63.58% stalling reduction when testing in the dynamic traces, comparing with the stable ones. These

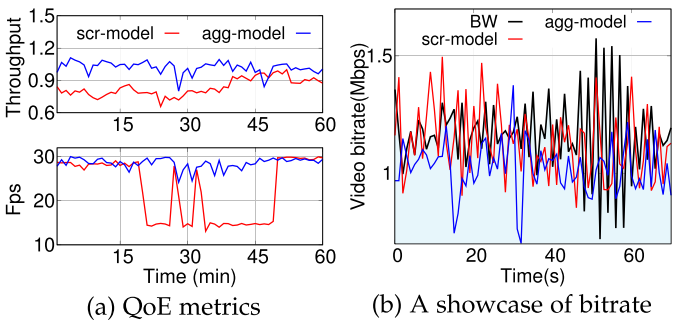


Fig. 16. Video performance with and without learning aggregation.

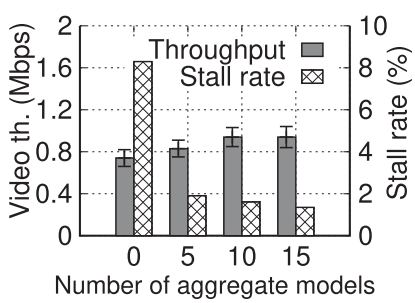


Fig. 17. QoE versus. number of models in aggregation.

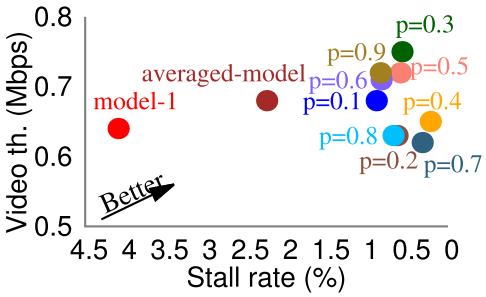


Fig. 18. Effect of prioritized aggregation.

results validate the necessity of aggregating models in similar network conditions.

The Impact of the Number of Models Involved in Aggregation. We train 15 different models using randomly 10 hour traces and select 0 (i.e., starting from scratch), 5, 10, and 15 out of them for aggregation. We examine the performance of the resulting 4 aggregated models under a same 1-hour test trace, as depicted in Fig. 17. We observe that: *(i)* Compared with the case with 0 aggregation, the aggregation of 5, 10, 15 models reduces the stall rate by 77.1%, 80.7%, 83.7%, respectively. *(ii)* While model aggregation also improves video throughput, the gain is relatively lower, i.e., 10.8%, 21.27% and 21.3%, respectively. Overall, *the more models involved in aggregation, the more swarm intelligence the aggregated model will have, and the better it can cope with the complex network dynamics.*

The Effect of Prioritized Aggregation. We aggregate 8 random individual models (Each model is trained for 1-hour) in 3 ways: averaged-model, a single model without averaging (named model-1), and prioritized-model (assigning a controlled weight of $p = \lambda_k$ to model-1, $(1 - \lambda_k)/7$ to the remaining models, and we vary the range of λ_k from 0.1 to 0.9.). Here we intentionally choose 3 traces from model-1's user, over which the eleven models run. We make an average over 3 traces to mitigate occasionality, and depict the results in Fig. 18. We observe that, *(i)* Even the simplest learning aggregation has 44.9%, 7.3% gains on stall rate and throughput than model-1 (individualized learning). The result indicates that Legato's learning aggregation mechanism indeed helps to improve QoE after leveraging swarm intelligence. *(ii)* While the throughput is relatively similar (the largest difference occurs under the two settings of $p = 0.3$ and $p = 0.7$, with a gap of 0.13 Mbps), the prioritized-models have a much lower stall rate than model-1 and averaged-model. In particular, $p = 0.4$ shows the lowest stall of 0.27%, while that of model-1 and averaged-model is 4.09% and 2.25%, respectively. *(iii)* Different weighted value of prioritized-models has certain impacts on the QoE metrics. Specially, we find that, the benefits of boundary weights (e.g., $p = 0.1, 0.2, 0.7, 0.9$) are not as good as the middle values (e.g., $p = 0.3, 0.5$). These findings indicate that *for an existing user who already has a trained model, it is better to balance its individual experience with others', by using medium aggregation weights (i.e., close to 0.5), so as to achieve the optimal QoE.*

The Effect of Thresholds in Network Condition-Aware Aggregation Mechanism. We select 3 kinds of th_{avg}, 1.0 Mbps (The representative bitrate of 360P), 1.27 Mbps (The representative bitrate of 480P), and 1.45 Mbps (The representative bitrate of 720P), and explore their impacts on video QoE. In each kinds of network condition categorization, we train 4 different models, and aggregate them following the design in Section 5.2. We compare them on two randomly selected traces, according to the aggregation utilization mechanism in Algorithm 1, and plot the averaged QoE-related metrics, as shown in Table 4, from which we observe that, with th_{avg} increased, the video bitrate also tends to slightly increase on average, meanwhile, it will cause higher loss rate and stall rate. Among them, th_{avg} of 1.27Mbps (corresponding to the most commonly used 480P video) can better balance thees matrices performance. Thus, we infer it's reasonable of existing thresholds set. Note that we will deploy such network condition-aware aggregation in practical real-time interactive applications, and we will report the results of large-scale deployments based on such thresholds in the future work.

TABLE 4
The Effects of Different Thresholds on Th_{Avg}

th_{avg}	Video bitrate (Mbps)	Loss rate (%)	Stall rate (%)
1.0 Mbps	0.9	3.02	2.29%
1.27 Mbps	0.89	1.84	2.14%
1.45 Mbps	0.99	3.26	2.3%

The Effect of Reward Function Design. We further add an experiment to evaluate the benefit of Legato's reward design (#1 for short) by comparing it against three classic designs. In order to conduct fair comparisons, we only modify the reward design, and keep Legato's other parts unchanged. The other three designs are: (i) product-based design following state-of-the-art Orca algorithm [44], which can be represented as $(throughput - \alpha * loss)/delay$ form, (#2 for short), (ii) Legato's reward w/o smoothness item (#3 for short), and (iii) Legato's reward with higher weights (e.g., 100) on smoothness item (represented by #4). We compare the QoE performance over above 4 methods under the same real-time video traces, and then plot the video throughput and stall rate metrics in Fig. 19. We find that, *(i)* Legato's original summation-based design shows similar QoE performance as the product-based design (#2). In particular, the throughput with product-based design drops by 8.06%. *(ii)* When Legato's reward is short of bitrate

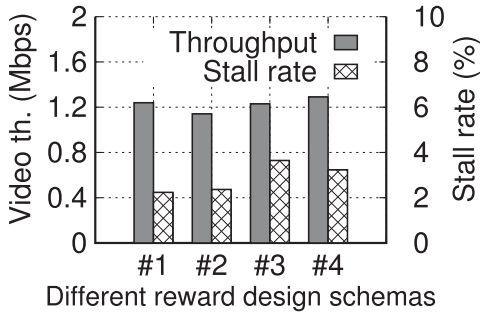


Fig. 19. Reward design comparing: #1 is Legato's original reward design in Eq. (2), #2 represents Orca's product-based reward, #3 stands for Legato's reward w/o bitrate smoothness item, #4 is equipped with higher weights on bitrate smoothness item.

smoothness item, the predicted bitrate will show larger fluctuation, thus is easier to generate overshooting behavior. So the stall rate will dramatically increase by 62.94% compared with Legato's original design. On the other hand, when the smoothness item is equipped with relatively higher weights, the bitrate will be inclined to gentle change thus less the adaption capabilities, and also shows more stalling events, i.e., 45.1% increase in stall rate.

8 Related Work

Interactive Video Transport. Real-time Internet Interactive video applications impose the toughest requirements on data transport protocols [1], [45]. Although the transport layer has been studied for decades, new network characteristics and applications are constantly emerging that motivate new designs. In contrast to the packet loss metrics commonly used in traditional congestion control protocols [46], [47], many emergent systems target low latency and have explored customized congestion indicators, including a mix of delay and loss [26], [48], [49], [50] as well low-layer KPIs of cellular networks [51], [52]. While these methods can better balance the tradeoff between delay and throughput, they rely on a set of hand-crafted controlling rules, and fall short of handling the increasing heterogeneity and dynamics especially for mobile networks [53].

Starting from Remy [11], machine learning models and particularly RL are utilized to generate more adaptive controlling rules automatically beyond the hand-crafted ones. Remy [11] explores a Markov model (i.e., a kind of tabular RL) to optimize congestion control algorithms. A feasible system, Concerto [15], designs an imitation learning (IL, a type of RL) based algorithm to better coordinate the codec and transport layers of video. While making remarkable progress, these models are all trained offline on simulators or emulators. Due to the simulation-to-reality gap, as reported in [19] and corroborated by our experiments (Section 7), these approaches often deliver marginal performance gain when tested at large-scale under real network conditions [16].

Learning-Based Transport for Video-on-Demand (VoD). Learning-based optimization has also been used in VoD services. To name a few, Pensieve [12] adopts the A3C RL algorithm [54] to predict the optimal VoD bitrate that fits the instantaneous network condition. Indigo [17] proposes an IL algorithm to improve VoD QoE. The most recent works [16], [23], [55] move forward to adapt and evaluate ML-based video streaming algorithms in real-world environment. In particular, ABRL [16] makes customized designs based on Pensieve algorithm to accommodate the unique challenges when running it on Facebook's VoD systems. QFLow [55] introduces a learning based network reconfiguration framework to achieve high QoE for YouTube streaming. Puffer [23] builds and operates a VoD open platform, and trains a supervised learning model "in situ", i.e., using the traces collected from the real deployment.

Compared with these systems, Legato differs in two major aspects: *(i)* Essentially, the ML models in these works are still trained offline (though using real-world traces), and thus detour unique challenges arising from online training, e.g., robust learning and swarm intelligence. *(ii)* They focus on VoD applications, which differ from the interactive video that imposes much harsher requirements atop today's best-effort Internet service [15]. In essence, VoD and real-time interactive applications are quite different for two reasons: *(i)* The VoD clients commonly maintain a playback buffer of dozens of seconds [9], so they are insensitive to short term (e.g., sub-second level) network dynamics. In contrast, real-time video is more sensitive to instantaneous network traffic variation, which is hard to reproduce in simulation. *(ii)* Most information of a VoD session is known in advance, e.g., the size of video chunks and buffers. In contrast, the real-time video content is always generated and consumed instantaneously. The algorithms need to responsively react to video dynamics under very tough low-latency intervals.

Online Learning. Classical online learning advocates training with data streams on the fly [56]. The salient feature is that it keeps refining its prediction model constantly in the presence of a new environment [57], in contrast to using a stationary post-trained model. Much research effort has been made on theoretical aspects of online learning in the AI community, ranging from the earliest online gradient update [58], [59], [60], [61] to the recent meta-learning [62]. Online learning has rarely been explored for practical network optimization. PCC [13] and PCC-Vivace [63] propose congestion control protocols sharing an online learning flavor. They leverage probing packets to continuously optimize a network utility objective. Park [64] proposes an open platform to facilitate experimenting with online RL for solving computer system problems, beyond RL's conventional application domains (e.g., gaming [65]). As we know, even though existing models can be fine-tuned, it's also based on offline updating after collecting lots of new training data. Examples include [17], [64]. Legato is dramatically different from such offline updating, it's trained online, the model is evolved with the data flow in an on the fly manner. Furthermore, Legato adapts online learning to address the unique challenges in interactive video transmission.

In Legato, we also propose a cooperative learning mechanism, which has shown its capability of preventing the catastrophic effect caused by RL's inherent but risky trial-and-error exploration. The cooperative learning component is inspired by the concept of safe learning, which has been adapted in other RL-based domains. For example, Shielding [66] detours RL's certain actions when a safety condition is triggered. A more recent work [67] proposes a guaranteed-safe policy to dynamically solve the load balancing problem. To our knowledge, Legato is the first framework to customize cooperative learning to improve the robustness of real-time interactive video.

Federated Learning is a distributed machine learning architecture, in which data is stored on distributed local devices instead of central servers [24]. Generally, federated learning aims to aggregate multiple users' experiences while preserving privacy, and also to reduce communication overhead [68], [69]. The learning aggregation component of Legato is inspired by the concept of federated learning. But Legato goes much further to solve practical problems in optimizing interactive video, including designing the specific RL neural network suitable with video transmission, enforcing RL's action despite video traffic dynamics and

enhancing video robustness by designing the robust cooperative learning mechanism.

9 Discussion

On-Device Online Learning. Our current deployment of Legato adopts a cloud-assisted architecture. The key design modules are located in remote cloud servers instead of the mobile devices, to accommodate the lack of mobile platform for RL training. Note that the RL cloud-server is unlikely to cause any privacy issue, as it only collects transport layer performance statistics, i.e., loss rate, RTT, without any personal information. In addition, it is also noteworthy that the mainstream Tensorflow-Lite [70] and Core-ML [71] only allow executing pre-trained RL models, i.e., they support the inference phase but not the training phase. The only RL-training-support platform in a recent work [39], as far as we know, is developed in Java and hard to be integrated with contemporary interactive video applications. However, as neural processing units (NPUs) rapidly become available on mobile devices, neural network training on mobile devices will be feasible. On-device training will eliminate the overhead and cost of deploying RL servers, which shall facilitate the widespread use of online learning algorithms like Legato.

Legato's Scalability. We note that Legato does not aim to improve reinforcement learning in general, but focuses on addressing the system-level challenges when adapting learning algorithms to optimize real-time video transport. The novelty of Legato lies in the online learning framework (including the neural network architecture and training methodology), the robust cooperative learning mechanisms, the fine-grained learning aggregation mechanism as well as in implementing, deploying and evaluating online RL on the interactive video prototype system. We believe these customized mechanisms of Legato may benefit future work on optimizing video transport and even other computer system design, such as resource scheduling and load balancing in practical networks. To extend the usage of Legato to other applications, one can reuse the architecture and mechanisms, but need to re-train RL models using the applications' own data. Besides, owing to Legato's cloud-assisted architecture, it can be applied to any device, not only the mobile device but also the PC, as long as the device is equipped with WebRTC framework.

10 Conclusion

In this work, we designed an online reinforcement learning based real-time interactive video system named Legato. We solve two unique challenges: avoiding the QoE damage from reckless exploitation of online RL by in-depth harnessing rule-based policy; adaptive learning aggregation mechanism to handle heterogeneous network environment. The real-world evaluation on an interactive video system and panoramic controlled testbed experiment demonstrate that Legato outperforms the state-of-the-art solutions. We believe Legato hints on a new direction that embraces online learning into more video communication applications, such as VoD, 360 panoramic video, virtual reality, volumetric video, *etc*.

Acknowledgments

The authors appreciate the insightful feedback from the editors and reviewers who helped improve this work.

References

[1] "Cisco visual networking index: Forecast and trends," 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11–741490.html?dtid=osscdc000283

[2] J. Clements, T. Gessesse, D. Sedani, and J. Klein, "Live video broadcasting mobile application for social sharing," U.S. Patent App. 14/821,519, Feb. 25 2016.

[3] M. Hopkins, "Live sports virtual reality broadcasts: Copyright and other protections," *Duke Law Technol. Rev.*, vol. 16, 2017, Art. no. 141.

[4] B. Han, Y. Liu, and F. Qian, "ViVo: Visibility-aware mobile volumetric video streaming," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 11:1–11:13.

[5] H. Wang *et al.*, "Toward cloud-based distributed interactive applications: Measurement, modeling, and analysis," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 3–16, 2018.

[6] I. Slivar, M. Suznjevic, and L. Skorin-Kapov, "Game categorization for deriving qoe-driven video encoding configuration strategies for cloud gaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 3s, pp. 56:1–56:24, 2018.

[7] Z. Wang, I. Reed, and A. M. Fey, "Toward intuitive teleoperation in surgery: Human-centric evaluation of teleoperation algorithms for robotic needle steering," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1–8.

[8] A. L. K. Lockwood *et al.*, "Interactions between vehicle and teleoperations system," U.S. Patent App. 15/644,310, Jan. 10 2019.

[9] S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptive video players over HTTP," *Signal Process. Image Commun.*, vol. 27, no. 4, pp. 271–287, 2012.

[10] "The future of ultra low-latency video streaming," 2021. [Online]. Available: https://api.video/blog/video-trends/the-future-of-ultra-low-latency-video-streaming

[11] K. Winstein and H. Balakrishnan, "TCP ex machina: Computer-generated congestion control," in *Proc. ACM SIGCOMM Conf.*, 2013, pp. 123–134.

[12] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Int. Group Data Commun.*, 2017, pp. 197–210.

[13] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *Proc. 12th USENIX Symp. Netw. Syst. Des. Implementation*, 2015, pp. 395–408.

[14] N. Jay, N. H. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on internet congestion control," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3050–3059.

[15] A. Zhou *et al.*, "Learning to coordinate video codec with transport protocol for mobile video telephony," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, 2019, pp. 29:1–29:16.

[16] H. Mao *et al.*, "Real-world video adaptation with reinforcement learning," in *Proc. Reinforcement Learn. Real Life Workshop*, 2019, pp. 1–10.

[17] F. Y. Yan *et al.*, "Pantheon: The training ground for internet congestion-control research," in *Proc. USENIX Annu. Tech. Conf.*, 2018, pp. 731–743.

[18] "Closing the simulation-to-reality gap for deep robotic learning," 2017. [Online]. Available: https://ai.googleblog.com/2017/10/closing-simulation-to-reality-gap-for.html

[19] K. Bousmalis *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4243–4250.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, pp. 1–12, 2017.

[21] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations*, 2016, pp. 1–14.

[22] H. Zhang *et al.*, "OnRL: Improving mobile video telephony via online reinforcement learning," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 29:1–29:14.
[23] F. Y. Yan *et al.*, "Learning in situ: A randomized experiment in video streaming," in *Proc. 17th USENIX Symp. Netw. Syst. Des. Implementation*, 2020, pp. 495–511.
[24] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[25] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-Based adaptive video streaming with FESTIVE," in *Proc. Int. Conf. Emerg. Netw. Experiments Technol.*, 2012, pp. 97–108.
[26] G. Carlucci, L. D. Cicco, S. Holmer, and S. Mascolo, "Congestion control for web real-time communication," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2629–2642, 2017.
[27] "WebRTC homepage," 2018. [Online]. Available: https://webrtc.org/
[28] Y. L. Cun, I. Kanter, and S. A. Solla, "Eigenvalues of covariance matrices: Application to neural-network learning," *Phys. Rev. Lett.*, vol. 66, no. 18, pp. 2396–2399.
[29] M. Pirotta, M. Restelli, and L. Bascetta, "Adaptive step-size for policy gradient methods," in *Proc. 27th Annu. Conf. Neural Informat. Process. Syst.*, 2013, pp. 1394–1402.
[30] "Tensorflow source code," 2020. [Online]. Available: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/tools
[31] "TFLearn: Deep learning library featuring a higher-level API for TensorFlow," 2020. [Online]. Available: http://tflearn.org/
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–15.
[33] "10 massive applications using webrtc," 2020. [Online]. Available: https://bloggeek.me/massive-applications-using-webrtc/
[34] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends Cogn. Sci.*, vol. 3, no. 4, pp. 128–135, 1999.
[35] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.
[36] I. J. Goodfellow and O. Vinyals, "Qualitatively characterizing neural network optimization problems," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015, pp. 1–20.
[37] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *ACM Queue*, vol. 14, no. 5, pp. 20–53, 2016.
[38] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein, "Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implementation*, 2018, pp. 267–282.
[39] C. Wang, Y. Xiao, X. Gao, L. Li, and J. Wang, "Close the gap between deep learning and mobile intelligence by incorporating training in the loop," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 1419–1427.
[40] "Linux traffic control," 2020. [Online]. Available: https://events.static.linuxfound.org/sites/events/files/slides/Linux_traffic_control.pdf
[41] "Video quality of service (QOS) tutorial," 2017. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-video/212134-Video-Quality-of-Service-QOS-Tutorial.html
[42] H. Zhang *et al.*, "Loki: Improving long tail performance of learning-based real-time video adaptation by fusing rule-based models," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 775–788.
[43] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.
[44] S. Abbasloo, C. Yen, and H. J. Chao, "Classic meets modern: A pragmatic learning-based congestion control for the internet," in *Proc. Annu. Conf. ACM Special Int. Group Data Commun. Appl. Technol. Archit. Protoc. Comput. Commun. Virtual Event*, 2020, pp. 632–647.
[45] J. Jiang *et al.*, "Via: Improving internet telephony call quality using predictive relay selection," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 286–299.
[46] Y. Jiang, J. Zhang, and Q. Guan, "Improvement of TCP reno congestion control protocol," *Sensors Transducers*, vol. 163, no. 1, 2014, Art. no. 308.
[47] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.
[48] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *Proc. 10th USENIX Conf. Netw.ed Syst. Des. Implementation*, 2013, pp. 459–471.
[49] Q. Xu, S. Mehrotra, Z. Mao, and J. Li, "Proteus: Network performance forecast for real-time, interactive mobile applications," in *Proc. 11th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2013, pp. 347–360.
[50] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *Proc. ACM Conf. Special Int. Group Data Commun.*, 2015, pp. 509–522.
[51] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "PiStream: Physical Layer Informed Adaptive Video Streaming over LTE," in *Proc. ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 427–439.
[52] X. Xie, X. Zhang, and S. Zhu, "Accelerating mobile web loading using cellular link information," in *Proc. Annu. Int. Conf. Mobile Syst. Appl. Serv.*, 2017, pp. 427–439.
[53] H. Zhang, A. Zhou, R. Ma, J. Lu, and H. Ma, "Arsenal: Understanding learning-based wireless video transport via in-depth evaluation," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10 832–10 844, Oct. 2021.
[54] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33nd Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[55] R. Bhattacharyya *et al.*, "QFlow: A reinforcement learning approach to high QoE video streaming over wireless networks," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2019, pp. 251–260.
[56] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, "Online deep learning: Learning deep neural networks on the fly," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2660–2666.
[57] A. Nagabandi, C. Finn, and S. Levine, "Deep online learning via meta-learning: Continual adaptation for model-based RL," in *Proc. 7th Int. Conf. Learn. Representations*, 2019, pp. 1–15.
[58] L. Bottou and Y. LeCun, "Large scale online learning," in *Proc. Adv. Neural Informat. Process. Syst.*, 2003, pp. 217–224.
[59] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.
[60] K. G. Vamvoudakis and F. L. Lewis, "Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem," in *Proc. Int. Joint Conf. Neural Netw.*, 2009, pp. 3180–3187.
[61] H. Bou-Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor, "Online multi-task learning for policy gradient methods," in *Proc. 31th Int. Conf. Mach. Learn.*, 2014, pp. 1206–1214.
[62] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. 5th Int. Conf. Learn. Representations*, 2017, pp. 1–11.
[63] M. Dong *et al.*, "PCC vivace: Online-learning congestion control," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implementation*, 2018, pp. 343–356.
[64] H. Mao *et al.*, "Park: An open platform for learning-augmented computer systems," in *Proc. Annu. Conf. Neural Informat. Process. Syst.*, 2019, pp. 2490–2502.
[65] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[66] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. 32nd AAAI Conf. Artif. Intell., 30th Innov. Appl. Artif. Intell., 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 2669–2678.
[67] H. Mao, M. Schwarzkopf, H. He, and M. Alizadeh, "Towards safe online reinforcement learning in computer systems," in *Proc. 33rd Conf. Neural Informat. Process. Syst.*, 2019, pp. 1–9.
[68] X. Yao, T. Huang, C. Wu, R. Zhang, and L. Sun, "Federated learning with additional mechanisms on clients to reduce communication costs," 2019, *arXiv:1908.05891*.
[69] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. 6th Int. Conf. Learn. Representations*, 2018, pp. 1–14.
[70] "Tensorflow lite," 2020. [Online]. Available: https://www.tensorflow.org/lite
[71] "Core ML framework," 2020. [Online]. Available: https://developer.apple.com/documentation/coreml

Huanhuan Zhang received the BE degree from the Beijing Forestry University, in 2017, and the PhD degree in computer science and technology from the Beijing University of Posts and Telecommunications (BUPT), in 2022. Her research interests include video streaming transport optimization, wireless network protocol, and machine learning.

Anfu Zhou (Member, IEEE) received the BS degree from the Renmin University of China, and the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy Sciences, in 2012. He is currently a professor with the school of Computer Science, Beijing University of Posts and Telecommunications. His research interests include mobile computing, wireless networking, and IoT systems.

Huadong Ma (Fellow, IEEE) received the BS degree in mathematics from Henan Normal University, Xinxiang, China, in 1984, the MS degree in computer science from the Shenyang Institute of Computing Technology, Chinese Academy of Science, Beijing, China, in 1990, and the PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Science, in 1995. He is currently a professor with the School of Computer Science, Beijing University of Posts and Telecommunications, China. From 1999 to 2000, he held a visiting position with the University of Michigan, Ann Arbor, MI. His current research interests include Internet of Things and sensor networks, multimedia computing, and he has published more than 300 papers in journals (such as ACM/IEEE Transactions) or conferences (such as ACM MobiCom, ACM SIGCOMM, and IEEE INFOCOM) and five books. He received the Natural Science Award of the Ministry of Education, China, in 2017. He received the 2019 Prize Paper Award of *IEEE Transactions on Multimedia*, 2018 Best Paper Award from IEEE Multimedia, Best Paper Award in IEEE ICPADS 2010, and Best Student Paper Award in IEEE ICME 2016 for his coauthored papers. He received the National Funds for Distinguished Young Scientists in 2009. He is an editorial board member of the *IEEE IoT Journal*, *ACM Transactions on Internet of Things*. He serves as the chair of the ACM SIGMOBILE China.