Generalizing Rate Control Strategies for Real-time Video Streaming via *Learning from Deep Learning*

Tianchi Huang[1], Rui-Xiao Zhang[1], Chenglei Wu[1], Xin Yao[1], Chao Zhou[2], Bing Yu[2], Lifeng Sun[1,3]
[1]Dept. of Computer Science and Technology, Tsinghua University
[2]Beijing Kuaishou Technology Co., Ltd., China
[3]BNRist, Dept. of Computer Science and Technology, Tsinghua University
{htc19@mails.,sunlf@}tsinghua.edu.cn

ABSTRACT

The leading learning-based rate control method, i.e., QARC, achieves state-of-the-art performances but fails to interpret the fundamental principles, and thus lacks the abilities to further improve itself efficiently. In this paper, we propose EQARC (Explainable QARC) via reconstructing QARC's modules, aiming to demystify how QARC works. In details, we first utilize a novel hybrid attention-based CNN+GRU model to re-characterize the original quality prediction network and reasonably replace the QARC's 1D-CNN layers with 2D-CNN layers. Using trace-driven experiment, we demonstrate the superiority of EQARC over existing state-of-the-art approaches. Next, we collect several useful information from each interpretable modules and learn the insight of EQARC. Following this step, we further propose AQARC (Advanced QARC), which is the light-weighted version of QARC. Experimental results show that AQARC achieves the same performances as the QARC with an overhead reduction of 90%. In short, through learning from deep learning, we generalize a rate control method which can both reach high performance and reduce computation cost.

ACM Reference Format:
Author. 2019. Generalizing Rate Control Strategies for Real-time Video Streaming via *Learning from Deep Learning*. In *ACM Multimedia Asia (MMAsia '19), December 15–18, 2019, Beijing, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3338533.3366606

1 INTRODUCTION

In recent years, we have witnessed a rapid increase on the demand of real-time video streaming. Due to the complicated environment and stochastic property in various network environments, how to transmit video stream with high quality of experience (QoE) has become the fundamental challenge. Most rate control approaches [2, 3, 5, 7, 12, 13] often pick future bitrate as high as possible with the permission of network conditions. However, due to the inequality between the following video qualities and video bitrates, such strategies may cause a large waste of bandwidth resources. Thus, QARC [6] adopts deep reinforcement learning (DRL) method to tackle the problem: it trains the neural network and picks the future video bitrate without any presumptions. In details (§2), QARC consists of two neural network (NN) modules: 1) *Video Quality Prediction Network (VQPN)* predicts the future video quality via previous video frames; 2) *Video Quality Reinforcement Learning (VQRL)* uses A2C [11] to train a model to pick the future bitrate encoded w.r.t the network status observed and the video quality for the future frames estimated by VQPN.

Nevertheless, QARC adopts high-performance neural network as a *black box*, and seldom considers whether it is explainable or not. Meanwhile, the ignorance of interpretation may seriously constrains its further improvements. In this paper, motivated by the increasing development of explainable artificial intelligence(XAI) [15, 17], we propose EQARC(Explainable QARC), aiming to demystifying the underlying reasons that why QARC works well, and then, attempting to grope a way for improving the proposed algorithm. Specifically, our work consists of several steps.

First, on the premise of maintaining the performance, we reconstruct all the neural network modules in QARC (§3), which involves: 1) *EVQPN* (Explainable VQPN): we design a novel hybrid attention-based CNN+GRU model to re-characterize VQPN (§3.1. Results demonstrate that EVQPN improves the prediction quality by 8.5% - 36.76% compared with the baseline methods (§4.2.1). Besides, we consider the best NN architecture for EVQPN. (§4.3) 2) *EVQRL* (Explainable VQRL): we reasonably replace the 1D-CNN layers with 2D-CNN ones in VQRL and use the global average pooling layer to take the place of the original fully-connected layer (§3.2). Using trace-driven experiments we find that the proposed method builds a visual heatmap representation that can explain the implicit attention of VQRL for each input state (§5.2). Results also demonstrate that EQARC generalizes well, which improves QARC on average QoE by 3.6% (§4.3). Next, by observing the features from EVQPN's attention heatmap, we realize the main function of VQPN can be effectively replaced by computing the average value of the gray-scale image (§5.1). Furthermore, we show that the Fast Fourier Transform (FFT) features play a vital part in the VQRL's input, especially, under the non-stationary network environments (§5.2). Finally, based on the aforementioned insights, we further implement AQARC, which combines EVQRL and EVQPN into one single NN model to determine the rate control policy (§5.3). Extensive results show that AQARC obtains the close performance as previous approaches while reducing about 90% of the model size.

In general, our contributions are multi-folds.

- To the best of our knowledge, we are the first to investigate the fundamental insights of the AI-based rate control method. Motivated by the recent success of XAI, we propose EQARC

MMAsia '19, December 15–18, 2019, Beijing, China

ACM ISBN 978-1-4503-6841-4/19/12…$15.00
https://doi.org/10.1145/3338533.3366606

for better understanding the key principles of QARC. Results indicate that EAQRC betters recent schemes (§4.2.1, §4.3).

- *Learning from deep learning* is effective. Many helpful and useful information can be found in all of the modules of QARC. Through *learning* from VQPN, we observe that extracting features by computing the average value of the gray-scale image w.r.t. each video frame are able to reflect the essence. (§5.1) Meanwhile, through *learning* from VQRL, we also find that the features extracted by FFT play a vital role in determining the next bitrate (§5.2).
- We further propose AQARC, which achieves the similar performance with QARC while reducing the overhead by about 90% (§5.3).

2 RELATED WORK

Real-time rate control methods have been proposed and applied about two decades. These schemes are mainly classified into three types: loss-based bitrate approach, delay-based bitrate approach and model-based bitrate approach. Loss-based approaches such as TFRC [5] and rate adaptation protocol (RAP) [12], have been widely used in TCP congestion control. Delay-based approaches, aiming to adjust sending rate to control the transmission delay, can be further divided into the end-to-end delay (RTT) approaches, such as TCP Vegas [2]; one-way delay approaches, such as LEDBAT (Over UDP) and TCP-LP [8, 13]; and delay gradient approaches [3]. Meanwhile, Model-based bitrate control method, such as Rebera [7] and GCC [3], has been proposed to control the sending bitrate via network status observed, including end-to-end latency gradient, receiving rate estimated, past bitrate sent as well as loss ratio.

On the basis of conventional real-time video streaming system architecture, Huang et. al. [6] propose QARC, a rate control approach that is deployed on the sender side. QARC uses reinforcement learning to *learn* the correlation among the previous video frame, network status, and the best future bitrate. In details, QARC composes of two feasible and useful models, which involves: **Video Quality Prediction Network (VQPN)**, proposed by the end-to-end deep learning method, predicts the future video quality metrics based on historical video frames; **Video Quality Reinforcement Learning (VQRL)** uses A2C to train the neural network to select bitrates for future video frames based on network status observations and the future video quality metrics predicted by VQPN.

3 EXPLAINABLE QARC'S MECHANISM

Previous work [6] implemented a complete rate control approach and performed well in various network conditions [6]. However, due to the complex NN architecture and uninterpretability of deep learning, we still can't find out why QARC can finish such a complicated task, which makes it difficult for us to further enhance the performance of QARC. To this end, we reconstruct QARC based on explainable artificial intelligence (XAI) and turned each of these modules into an interpretable machine learning architecture. Meanwhile, a vital challenge for us is to preserve the performance of QARC as much as possible during the reconstruction. In this section, we start by introducing the EVQPN's NN architecture. We then describe the implementation details of EVQRL. Finally, we explain the offline network simulator and how it helps QARC to train the neural network models.

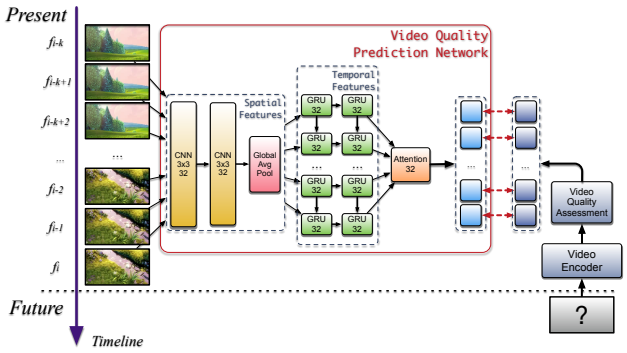


Figure 1: VQPN Architecture Overview

3.1 Explainable-VQPN (EVQPN)

Based on the original QARC [6], we implement explainable video quality prediction network(EVQPN) to help the RL model understand the future frame's perceptual video quality. Figure 1 describes the EVQPN's neural network architecture, which is composed of three modules: one Convolutional Neural Network (CNN) module extracts the spatial information; one Recurrent Neural Network (RNN) module captures the temporarily features; and a third attention layer comes after the RNN module to demonstrate the implicit attention of the temporal information. Details are shown as follows.

Input. EVQPN takes $F_i = [f_{i-k}, f_{i-k+1}, \cdots, f_i]$ as the state inputs, where f_i denotes the i-th sampled video frame.

Spatial information extraction. EVQPN uses several 2D-Conv layers to extract frame features, which can obtain the spatial information of each video frame in inputs F_i. Furthermore, we introduce a global average pooling layer (GAP) after the last Conv layer instead of the fully-connected layer aiming to encourage EVQPN to identify all the discriminative parts of feature maps [17] and to make EVQPN interpretable. Note that by removing the fully-connected layers, the number of neurons and the overhead of the network are both significantly reduced, while the performance is slightly improved compared with the previous work [6].

Capturing temporal features. Upon extracting frame features via CNN+GAP, VQPN leverages a double-layer recurrent module [4] to further extract temporal characteristics of the video frames F_i in past k sequences. For illustrating the weights of each recurrent unit clearly, we present a deterministic "soft" attention [1, 10] layer after the recurrent module. In this work, we use the self-attention model [9]. The idea is to calculate the output V_{t+1} based on a context vector c_i with consideration of all the current hidden states h_i of the recurrent layer, and c_i is formulated as: $c_i = \sum_{j=1}^{k}\{Softmax(f_{att}(h_j))h_i\}$. In which $Softmax(f_{att}(h_j))$ is a variable length alignment vector symbols the weight of each hidden state h_i. The attention function f_{att} is defined as $f_{att}(h_i) = \mu_\theta^T \tanh(w_\theta h_i + b_\theta)$, Where μ_θ, w_θ and b_θ are all learnable parameters. In practice, we implement the attention function as a single fully-connected layer with the tanh activation function.

Output: The output of VQPN is V_{t+1}, a vector indicating the prediction of the video quality assessment in the next time slot $t+1$ encoded in several bitrates.

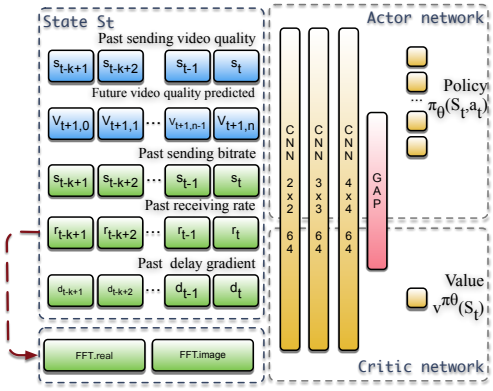


Figure 2: The Actor-Critic algorithm that VQRL uses to generate sending bitrate selection policies

Loss function: We use mean square error(MSE) as the loss function. Besides, we consider adding regularization to the loss function to avoid over-fitting on the training set. To encourage the diversity of the attention vectors and to overcome the aforementioned shortcomings [16], we add another penalization formulated by Frobenius norm. Let $\hat{V}_t$ denote the real vector of video quality score of the video in time t, the loss function can be written as (Eq. 1):

$$L_t(V;\theta) = \frac{1}{N}\sum |V_t - \hat{V}_t|^2 + \lambda_0||\theta||^2 + \lambda_1||\alpha\alpha^T - 1||_F^2 \qquad (1)$$

where λ_0 and λ_1 are the regularization coefficients.

3.2 Explainable Video Quality Reinforcement Learning (EVQRL)

Inspired by the original QARC, we formulate real-time video streaming problem with the A2C framework (See in Figure 2), named as explainable video quality reinforcement learning (EVQRL). Detailing our system components, which include:

State: We consider the metrics that can be obtained by both the sender and receiver in a feedback message session. VQRL's learning agent pushes the input state of time-slot t $s_t = \{p, v, s, r, d\}$ into the neural network, where p means the past sending video quality; v is the future video quality predicted by VQPN; s is the video sending rate of the past k sequences that is equal to the throughput measurement from the sender; r represents the receiving bitrate of past k sequences measured by the receiver; d is the delay gradient measured between a sender and receiver of the recent k sequences. Besides, we also add the additional features into input which decomposed the receive rate sequence through FFT.

Action: The agent needs to take an action when receiving the state, and the policy tells the agent which action should be selected in the RL problem. In general, the action space is discrete, and the output of the policy network is defined as a probability distribution: $f(s_t, a_t)$, denoting the probability of the selection action a_t under the state s_t. In this paper, the action space contains the candidates of sending bitrate in the next time-slot.

Reward: The reward is set to maximum the Quality of experience (QoE) perceived by the user. In this work, QoE is influenced by video quality, latency, and smoothness (§4.1).

Network Architecture Representation: The output of the actor network is a 5-dimension vector and the final output of the critic network is a linear value. In particular, they use the same network architecture consisting of two 2D-Conv layers with 64 filters.

4 EVALUATION

In this section, we evaluate EQARC with several experiments. We first show the datasets and metrics. Then we compare EVQPN with previously proposed methods and discuss the best NN architecture for EVQPN. Finally, we compare the average QoE of EVQRL with the original scheme.

4.1 Implementation

▹ **Testbed:** Considering the main contribution of this paper is to improve the NN architecture rather than performance, we experience the evaluation on the existing QARC's testbed [6] with same videos and network traces.

▹ **Video dataset:** We leverage QARC's video quality dataset as the video set. For each video in datasets, the VMAF metric is given with the bitrate in range of {300kbps, 500kbps, 800kbps, 1100kbps, 1400kbps}, and the reference resolution is configured as 800×480, which is the same size as the default resolution observed by the receiver during the real-time live streaming.

▹ **Network traces:** We use several network traces to evaluate EQARC, including packet-level network traces, chunk-level network traces, and synthetic network traces.

▹ **QoE metrics:** We adopt QARC's QoE metric (Eq. 2), which is commonly used in recent work [6], to evaluate EQARC. Ideally, how to evaluate QoE is complicated but not unique. In this paper, we only focus on proposing a rate-control method rather than a reliable QoE function.

$$QoE = \sum_{n=1}^{N}(V_n - \alpha B_n - \beta D_n) - \gamma \sum_{n=1}^{N-1}|V_n - V_{n-1}| \qquad (2)$$

Where V_n means the video quality of time n, B_n is the video bitrate that the sender selects, and D_n represents the *delay gradient* measured by the receiver. The last term is the smoothness of video quality. Coefficients α, β and γ are the weight to describe their aggressiveness. In this paper, followed by recent QARC[6] paper, we set $\alpha = 0.2$, $\beta = 10.0$, and $\gamma = 1.0$.

4.2 EVQPN Evaluation

4.2.1 EVQPN vs. Proposed Methods. We compare EVQPN to the following previously proposed methods which collectively represent the conventional prediction method:

(1) *Artificial neural networks (ANNs)*: leverages a single layer with 64 neurons to predict future video quality;
(2) *Fully-connected neural networks (FC)*: uses 3 traditional fully-connected layers with {128, 64, 64} hidden units;
(3) *Recurrent Neural Network (RNN)*: employs a conventional single-layer recurrent neural network with 128 hidden units;

Table 1: Comparing performance (RMSE) of VQPN with different regression algorithms.

Method	RMSE	Improvement(%)
ANN	0.181	-
FC(Baseline)	0.068	-
RNN	0.056	17.65
GRU	0.060	11.76
SI,TI	0.064	5.88
VQPN [6]	0.047	30.88
EVQPN	**0.043**	**36.76**

Table 2: Comparing the performance (RMSE) and the overhead of VQPN with different channel / hidden units. Results are collected under learning rate=1e-4.

Channel / Hidden Units	RMSE	Model Size(MB)
2	0.054	0.022
4	0.055	0.086
8	0.059	0.33
16	0.046	1.4
32	**0.043**	**5.3**
64	0.052	21.0
128	0.052	84.0

(4) *Gated Recurrent Unit (GRU)*: uses an advanced double-layer recurrent neural network with 128 hidden units.
(5) *SI&TI*: leverages specific input features including Spatial perceptual Information (SI) and Temporal Information (TI)[14].
(6) VQPN [6]: Previously proposed VQPN architecture passes k = 25 previous frames as the input to the NN architecture.

The comparison of the Root Mean Square Error (RMSE) between EVQPN against other regression algorithms is shown in Table 1. Note that we compare the performance of EVQPN with previously proposed work on the same test datasets. As expected, the results demonstrate that EVQPN imporves the prediction quality by 36.76% compared to the baseline and by 8.5% compared to VQPN.

4.2.2 EVQPN Deep Dive. Table 2 shows our results with different settings of channel number / hidden units. Results are summarized with RMSE metric. Empirically, channel number / hidden units = 32 yields the best performance. Meanwhile, its overhead and the model size are acceptable and guarantee the deploy ability in the real-world scenarios.

▹ **Generalization.** To sum up, EVQPN passes totally $k = 25$ previous frames ($t = 5s$ past time video and 5 frames for each second) as input to the neural network architecture, and the size of frames is defined as 64×36 with 3 channels. The features of the input frames are extracted as 32-dimension vectors via the spatial information extraction module. The spatial information extraction module is composed of 2 Conv layers, both of which consist of 32 filters with size 3 and stride 1. There is no max pooling layer in the spatial information extraction module. Finally, these feature maps are passed into a global average pooling layer. After extracting the spatial information of each video frame, a recurrent network is proposed

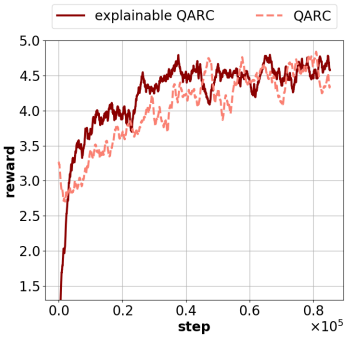


Figure 3: Comparsion of EQARC and QARC [6].

to estimate future video quality. VQPN passes $k = 25$ feature maps to a gated recurrent unit(GRU) layer with 32 hidden units, then the states of that layer are passed to another GRU layer with the same hidden units. A self-attention layer is performed after the GRU layer. Finally, EVQPN outputs a 5-dimension vector for each video frame, and each value in the vector represents the video quality score w.r.t. video bitrate {300, 500, 800, 1100, 1400} kbps. In short, the architecture of EVQPN is defined as $Conv1a_{32} \rightarrow Conv1b_{32} \rightarrow GAP \rightarrow GRU2a_{32} \rightarrow GRU2b_{32} \rightarrow Attention3_{32} \rightarrow Fc4$. We utilize the Adam optimizer to optimize EVQPN with the learning rate $\alpha = 10^{-4}$. We use TensorFlow to implement this architecture, and leverage the TFLearn deep learning library's TensorFlow API to declare EVQPN.

4.3 EVQRL vs. VQRL

For evaluating the performance of EVQRL architecture, we compare the previous proposed VQRL and the explainable VQRL on the experimental settings described in §3.2 and §4.1 respectively. Experimental results is shown in Figure 3, and we observe that EQARC performs better than QARC with improvements in average reward of 3.6%.

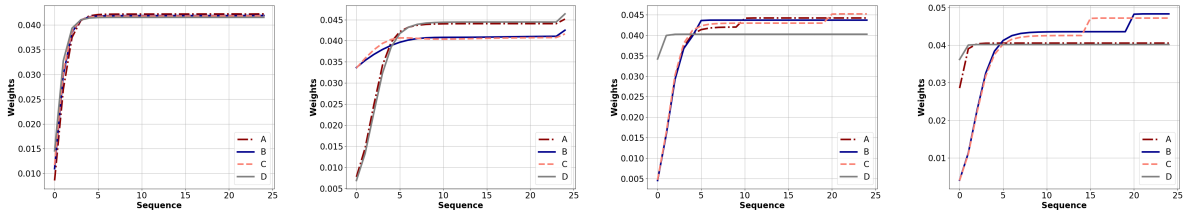
5 LEARNING FROM EQARC

In this section, we try to explore the insight of EQARC under the guidance of the attention heatmaps of EVQPN and EVQRL respectively. The section is mainly composed of three parts, with the first two representing the learning process of each module in EQARC. Then We finally propose an advanced QARC (AQARC) approach based on the learned insights.

5.1 Learning From EVQPN

In this subsection, we first introduce the method of how to inspect the attention heatmap. We then discover temporal information insight through the attention heatmap and further find that extracting the feature as the average value of a gray-scale can replace the original CNN-based spatial information extraction model.

Methodology: Unlike the previous work (e.g. [17]), EVQPN extracts its attention features with jointly consideration of both spatial information and temporal information. Thus, the attention



(a) A music video in static scenario. (b) A music video with variable scene. (c) A comic video clip. (d) A soccer game video clip.

Figure 4: The sample attention weight $\alpha_t = Softmax(f_{att}(h_t))$ of several video clips.

(a) Be yourself. (b) Tsubasa wa Iranai. (c) Big Buck Bunny. (d) World Cup Live.

Figure 5: The visualization heatmap captured from different video scenarios.

heatmap $\mathcal{A}_{i,k}$ for each frame f_i in VQPN's input is computed as

$$\mathcal{A}_{i,k} = \mathcal{F}^k(f_i) \cdot [\alpha_i w_k]^T. \quad (3)$$

Here, let $\mathcal{F}^k(f_i)$ represent the activation feature map of the last Conv layer, and w_k denotes the weight corresponding to class k for output bitrates.

Visualization Setup: To better understand the reason why EVQPN performs well, we apply our EVQPN model to predict video quality on several video clips in different scenarios including a live cast scenario, a sports game video clip and a music video shot. Meanwhile, the attention feature map $\mathcal{F}(f_i)$ and the alignment vector α_i will be recorded during the test. Figure 5 summarizes the results.

Results and Analyze: As expected, the temporal information can be explained clearly through the results. We observe that the attention weight α performs quite different in various video scenes. In Figure 4(a), we find that when the scene is not frequently switched, EVQPN stably refers to the spatial information of the past video frames, and vice versa (See Figure 4(d)). Due to the same video scene switching attribute, the remaining two video clips, i.e., Figure 4(b) and Figure 4(c) illustrate almost similar results.

The raw video frame and its attention heatmap are illustrated in Figure 5(a-d). However, we observe that there are some isomorphic features among the spatial information of video frames. No matter how complex the video frame is, the neural network still seems like to output the heatmap as a $gray - scale$ image rather than highlights the specific discriminative regions. Figure 6 shows the same conclusion clearly.

Table 3: Comparing performance (RMSE) of gray-scale approach with VQPN.

Video Clip Name	EVQPN	gray-scale + GRU
Be yourself	**0.013**	0.026
Big Buck Bunny	**0.116**	0.116
Tsubasa wa Iranai	0.042	**0.040**
World Cup Live	**0.121**	0.124

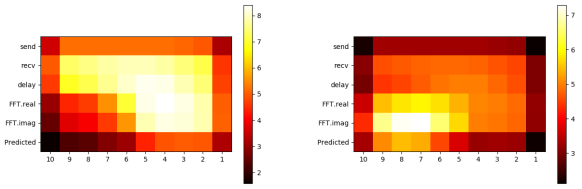
(a) Raw Video Frame. (b) Attention heatmap.

Figure 6: The raw video frames and its attention heatmap are processed in gray-scale image. The video frame is captured from a soccer video clip named *World Cup Live.*

Rethinking EVQPN's architecture: Our key idea is to identify the performance of the model using a group of gray-scale video frames and a one-layer RNN. In detail, we use the average value of the gray-scale image as the feature for each video frame, then we perform a one-layer GRU with a self-attentional model. The input and the output are similar to those of EVQPN. In Table 3, we report the results of this approach with different video clips and we also provide comparisons on RMSE between the gray-scale approach and EVQPN. We observe that the results of two approaches are close, especially in the *Tsubasa wa Iranai* task, the gray-scale approach performs better than EVQPN with the improvement of 4.76%. Nevertheless, there is still a wide distance in the static video scenes.

5.2 Learning from EVQRL

Methodology: As is already described in 3.2, we use a network architecture similar to class activation maps(CAM) [17] to solve the bitrate selection task and to further understand its action. We, therefore, formulate the attention heatmap of EVQRL as follows:



(a) stationary network environments. (b) non-stationary network.

Figure 7: Attention heatmap of EVQRL which are performed in both stationary and non-stationary network.

$\mathcal{A}_c(S_t) = \sum_k w_k^c \mathcal{F}^k(S_t)$, where $\mathcal{A}_c(S_t)$ is defined as the attention heatmap for action c; $\mathcal{F}^k(S_t)$ represents the activation feature map of the last Conv layer; w_k^c is the weight corresponding to action c for unit k and k means the filter number.

Experiments and Visualization Results: We visualize the attention heatmap of EVQRL under different network conditions. Results are summarized in Figure 7. We notice that in stationary network conditions, EVQRL focuses on almost all the network features including receive data and queuing delay gradient. By contrast, it neglects the video quality metric feature because of the feasible network conditions. However, while in the non-stationary network environments, EVQRL pays more attention to the FFT features and the video quality metrics instead of the raw network status.

5.3 Advanced QARC

On the basis of the results of EVQPN and EVQRL, we move forward to investigate how to improve or *refine* the proposed EQARC architecture. Inspired by the EVQPN's insight, we refine the EVQRL's network architecture by replacing the prediction video quality features with the average value of a gray-scale image and disable EVQPN module. In detail, the state of VQRL has been changed to $s_t = \{g, s, r, d, f_r^r, f_r^i\}$, in which g represents the average value of gray-scale images in time-slot t (See Eq. 4, where $\mathcal{G} = [0.30, 0.59, 0.11]$), and the rest of them are same as the state of previously proposed EVQRL.

$$g = \frac{1}{W \times H \times N} \sum_{k=0}^{N} \sum_{x=0}^{W} \sum_{y=0}^{H} \mathcal{G}^T f_k(x, y) \tag{4}$$

We then implement the AQARC approach referring to the same action and reward as illustrated in §4.1. Next, we evaluate them on the same test dataset. Results are summarized in Table 4. By means of results, we find that the AQARC obtains the close performance as the previous approaches by reducing about 90% of the model size overhead, thus we can prove that the conclusion learning from deep learning is effective and impressive.

6 CONCLUSION

Previous learning-based method, i.e., QARC lacks interpretability, resulting in the the difficulties for future improvement. In this paper, we attempt to use XAI to reconstruct QARC for finding out the fundamental principles. Following this idea, we propose EQARC, which not only outperforms recent work but also shows its interpretablity. Moreover, based on the insights observed from EQARC, we further propose AQARC, which significantly reduces QARC's overhead while preserving its original performance. Results indicate that the proposed method is useful and effective.

Table 4: Comparisons with different *QARC approaches* on the same test set. We evaluate the average QoE and the overhead for each approach receptively.

Scheme	Average QoE	Model Size (MB)
QARC [6]	5.55 ± **0.34**	8.9
EQARC (§3.2)	5.65 ± 0.74	6.5
AQARC (§5.3)	**5.70** ± 0.92	**0.88**

Acknowledgements. We thank the anonymous reviewer for the valuable feedback. This work was supported by the National Key R&D Program of China (No. 2018YFB1003703), NSFC under Grant 61936011, Beijing Key Lab of Networked Multimedia, and Kuaishou-Tsinghua Joint Project (No. 20192000456).

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014). arXiv:1409.0473 http://arxiv.org/abs/1409.0473
[2] Lawrence S. Brakmo and Larry L. Peterson. 1995. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on selected Areas in communications* 13, 8 (1995), 1465–1480.
[3] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 13.
[4] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv: Neural and Evolutionary Computing* (2014).
[5] Mark Handley, Sally Floyd, Jitendra Padhye, and Jörg Widmer. 2002. *TCP friendly rate control (TFRC): Protocol specification.* Technical Report.
[6] Tianchi Huang, Rui-Xiao Zhang, Chao Zhou, and Lifeng Sun. 2018. Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning. In *2018 ACM Multimedia Conference*. ACM, 1208–1216.
[7] Eymen Kurdoglu, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu. 2016. Real-time bandwidth prediction and rate adaptation for video calls over cellular networks. In *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 12.
[8] Aleksandar Kuzmanovic and Edward W Knightly. 2006. TCP-LP: low-priority service via end-point congestion control. *IEEE/ACM Transactions on Networking (TON)* 14, 4 (2006), 739–752.
[9] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).
[10] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *CoRR* abs/1508.04025 (2015). arXiv:1508.04025 http://arxiv.org/abs/1508.04025
[11] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. 1928–1937.
[12] R. Rejaie, M. Handley, and D. Estrin. 1999. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *INFOCOM 99, IEEE*, Vol. 3. 1337–1345 vol.3. https://doi.org/10.1109/INFCOM.1999.752152
[13] Dario Rossi, Claudio Testa, Silvio Valenti, and Luca Muscariello. 2010. LEDBAT: The New BitTorrent Congestion Control Protocol.. In *ICCCN*. 1–6.
[14] International Telecommunications. 2007. R-REC-BT.1788. https://www.itu.int/rec/R-REC-BT.1788-0-200701-I/en. (2007).
[15] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. 2048–2057.
[16] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. 2018. Deep Learning in Mobile and Wireless Networking: A Survey. *arXiv preprint arXiv:1803.04311* (2018).
[17] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *CVPR*.