Delay-Constrained Rate Control for Real-Time Video Streaming with Bounded Neural Network

Tianchi Huang[†][*], Rui-Xiao Zhang[*], Chao Zhou[‡], and Lifeng Sun[*][§]
[*] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡] Beijing Kwai Technology Co., Ltd, China
[†] Department of Computer Science and Technology, Guizhou University, Guizhou, China
{htc17,zhangrx17}@mails.tsinghua.edu.cn
zhouchaoyf@gmail.com, sunlf@mail.tsinghua.edu.cn

ABSTRACT

Rate control is widely adopted during video streaming to provide both high video qualities and low latency under various network conditions. However, despite that many work have been proposed, they fail to tackle one major problem: previous methods determine a future transmission rate as a single for value which will be used in an entire time-slot, while real-world network conditions, unlike lab setup, often suffer from rapid and stochastic changes, resulting in the failures of predictions.

In this paper, we propose a delay-constrained rate control approach based on end-to-end deep learning. The proposed model predicts future bit rate not as a single value, but as possible bit rate ranges using target delay gradient, with which the transmission delay is guaranteed. We collect a large scale of real-world live streaming data to train our model, and as a result, it automatically learns the correlation between throughput and target delay gradient. We build a testbed to evaluate our approach. Compared with the state-of-the-art methods, our approach demonstrates a better performance in bandwidth utilization. In all considered scenarios, a range based rate control approach outperforms the one without range by 19% to 35% in average QoE improvement.

CCS CONCEPTS

• **Information systems** → *Multimedia streaming*; • **Computing methodologies** → *Neural networks*;

KEYWORDS

Real-time Video Streaming, Rate Control, Deep Learning, Delay-Constrained

ACM Reference Format:
Tianchi Huang, Rui-Xiao Zhang, Chao Zhou, Zeping Li, and Lifeng Sun. 2018. Delay-Constrained Rate Control for Real-Time Video Streaming with Bounded Neural Network. In *Proceedings of MMSys 2018 Nossdav Workshop (NOSSDAV'18).* Amsterdam, The Netherlands, June 15, 2018, Article 6, 6 pages. https://doi.org/10.475/123_4

§ Lifeng Sun is the correspondence author of this paper.
† This work was done by Tianchi Huang during his visit to Tsinghua University.

NOSSDAV'18, June, 2018, Amsterdam, The Netherlands
© 2018 Association for Computing Machinery.
ACM ISBN 123-4567-24-567/08/06...$15.00
https://doi.org/10.475/123_4

1 INTRODUCTION

Recent years have seen a rapid increase in the requirements of real-time video streaming. People publish and watch live video streaming smoothly from supported applications at any time, in any where, and under any network environments. Due to the complicated environment and stochastic property in various network environments, the abundance of rate control approaches have been proposed to solve the fundamental problem: how to transport video stream with higher video bitrate and lower latency. For conventional approaches, loss-based approaches[8, 15] use packet loss to control their session. Moreover, these methods will cause delay instability.[14]. To solve this problem, delay-based approaches[9, 16] are proposed to make the end-to-end delay converge to a target value by controlling sending rate. However, the final delay of these approaches is not constrained [6]. Model-based approaches[3, 4, 10] aim to design a model [8]to describe the underlying relationships by analyzing observed historical network status. However, forecasting future network throughput as a single value seems to be unreliable[13, 18, 19]. Meanwhile, many studies focus on describing future network throughput as a probability model[18]. Nevertheless, the fixed rules assumption continues to struggle with their performance.

In our study, we aim to use a suitable range to describe the future network conditions instead of using a single value. However, we cannot quantify the value of "suitable" clearly, and we only know the aim of "suitable" means "a range can cover the future observations but not too broad." Motivated by this, We try to explore the insight of network congestion via end-to-end deep learning[12]. Starting from this concept, we propose a delay-constrained rate control approach to constrain end-to-end delay during the session. Our approach is placed on the receiver. Upon receiving packets from the sender, the receiver feeds a message containing the next time sending rate back to the sender. The sender then changes encoder's parameters to fit it. Our method consists of two modules, the delay filter, and the rate estimator. The delay filter is a module that computes delay gradient required using previously delay gradient observations. The rate estimator is an end-to-end deep learning model which can estimate future throughput in a range by using historical observations. Due to this unique requirements, the rate estimator consists of two neural networks, the prediction network (PN) and the error estimation network (EEN). By comparing each candidate architecture, we finally propose the best architecture as the rate estimator.

After that, we evaluate our rate control approach using a full system implementation. We collect a large corpus of data from real-world network environments as a training dataset to optimize it. We perform several experiments to compare the performance with previously proposed approaches and evaluate the effectiveness of range factor. The key contributions of our paper are as follows:

- We design and train a novel neural network architecture to predict future network status in range instead of a single value, which can be more conducive to encode video in real-time video streaming scenario.
- We propose a delay-constrained rate control approach which outputs as a range to control the session in low latency and high bitrate under the premise of stability.

2 RELATED WORK

Rate control methods have been proposed and applied about two decades. These schemes are mainly classified into three types, loss-based, delay-based and model-based[17].

Loss-based: Loss-based approaches, for instance, TFRC[8] and rate adaptation protocol (RAP)[15], have been widely used in TCP congestion control, which increases bitrate till packet loss occurs. However, until packet loss occurs, latency also increases, vice versa. Thus, a bad experience will be given to the users because of the network jitter like sawtooth. Thus, using packet loss event as the control signal may cause its throughput to be unstable, especially in error-prone environments[6].

Delay-based: Some studies have also focused on delay-based approaches solving the problem of the loss-based approach. Delay-based approaches try to adjust sending rate to control the transmission delay. According to the way in which these approaches calculate delay, they can be divided into the end-to-end delay (RTT) approaches, such as TCP Vegas[2]; one-way delay approaches, for instance, LEDBAT (Over UDP) and TCP-LP[11, 16], and delay gradient approaches[4]. However, it is hard for the delay metrics to converge to the target value in some network conditions which its upper limit has always suffered a wide range of changes, such as WIFI[6, 14].

Model-based: Model-based approaches are also proposed in recent years to control congestion such as Rebera[10] and GCC (Google Congestion Control)[4]. These approaches are aimed to design a model or a filter to describe the underlying relationships by previously network status observations. These model forecast future throughput as a single value. Nevertheless, due to the unstable network environments, the single value may not fully describe the future network status, especially in real-time live video streaming scenario.

3 SYSTEM MECHANISM

We start with an overview of real-time live streaming scenario consisting of a sender and a receiver. Figure 1 illustrates the architecture. The sender generates the raw live streaming data using a camera and sends them to the receiver. The receiver receives the data stream and displays the view. A UDP-based protocol is

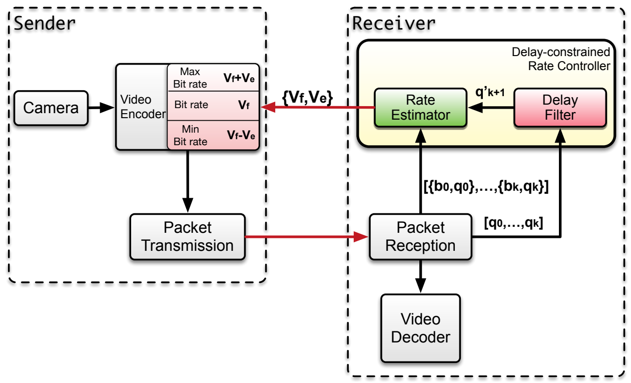

Figure 1: Delay-constrained Rate Control System Overview

established to send live video stream in MTU-sized packet[1] and receives feedback reports from the receiver. In general, our approach constrains the delay with the methods as follows:

Delay filter, placed on the receiver, which computes future delay gradient $\hat{q_t}$ on demand from historical delay gradient observations, collected that is fed back to the sender with the aim of constraining the delay;

Rate estimation with bounded neural network, placed on the receiver, which computes the sending bitrate range $[V_f - V_e, V_f + V_e]$, where V_f is described as a target sending bitrate value, and V_e is its error between V_f and future observations;

Rate control, placed on the sender, which controls the encoding bitrate of the video encoder.

4 METHODS

4.1 Delay Filter

Aiming to constrain the transmission delay by controlling the sending rate in the sender, we design a delay filter module that computes delay gradient on demand denoted as $\hat{q}_{k+1}$ according to delay gradient sequence $[q_0, q_1, \ldots, q_k]$ of past k time-slots.

At the same time, when we choose the bitrate , we try to optimize the following formulation, as is shown in Eq.1,

$$\min\left[\overline{q}^2 + \alpha\sigma^2\right] \tag{1}$$

where $\overline{q} = \frac{1}{k+1}(\sum_{i=1}^{k} q_i + \hat{q}_{k+1})$, and $\sigma^2 = \frac{1}{k+1}(\sum_{i=1}^{k}(q_i - \overline{q})^2 + (\hat{q}_{k+1} - \overline{q})^2)$. Coefficient α is the weight to describe its aggressiveness. T he first part of this object tries to make the average delay gradient approximate zero to obtain the constant queuing delay and the second part is to minimize the variation of the "change" of the delay gradient, aiming to avoid a fierce change of queuing delay. (Eq. 2). We will discuss the best α in Section 5.3.1.

$$\hat{q}_{k+1} = \frac{\alpha - 1}{\alpha + k}\sum_{i=1}^{k} q_i \tag{2}$$

Delay Gradient Measurement. In real-time video streaming control scenario, one of vital work for congestion control is to

[1]In this paper, the sender delivery of a single 1500-byte (MTU-sized) packet.

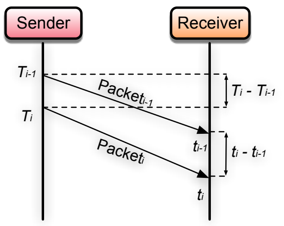

Figure 2: Delay Gradient Measurement

measure one-way delay or queuing delay of the session. However, the clocks on both sides are not synchronous which makes the delay measurement unreliable. Thus, delay gradient, defined as the difference on delay measurement on both side, has been proposed to measure the latency in unsynchronized clock environment.

Delay gradient (Figure 2) $q(t_i)$ is computed as follows[4] (Eq. 3) , which is actually used to describe the variety of the queuing delay. If the network is congested, its delay gradient will be greater than zero, vice versa. If the delay gradient equals to zero, we can only infer that network is not congested.

$$q(t_i) = \frac{1}{N} \sum_{i=1}^{N} [(t_i - t_{i-1}) - (T_i - T_{i-1})] \tag{3}$$

Where T_i is the time at which the i-th packet has been sent, and t_i is the time at which the i-th packet has been received, and N represents the packet count in a period. However, in real-world network environment, delay gradient is often observed along with burst noise. In our study, we filter the noise by a complementary filter.

4.2 Rate Estimation with Bounded Neural Network

In this section, we mainly set up the rate estimator module in several steps. We start by explaining its bounded neural network architecture including inputs, outputs, training methodology, and so forth. Compared the performance with four architecture candidates, we then confirm the best neural network architecture of rate estimator.

4.2.1 Bounded Neural Network Architecture. Our motivation is to build a model which uses previously historical observation to predict the range of future observations. We model the range as $[V_f - V_e, V_f + V_e]$, where V_f is described as a baseline value, and V_e is its error between the baseline value and future observations. However, the range will lose its function if V_e is too large. Motivated by this, We propose a novel neural network architecture, named as "bounded neural network", with two neural networks to maximize the accuracy using single value V_f and to minimize the error value V_e. We describe its network structure and its training methodology. As shown in Figure 3, neural network uses the method which involves training two neural networks, prediction network (PN) and error estimation network (EEN). The detailed functionalities of these networks are explained below.

Inputs After the duration of k time-slots, rate estimator takes inputs $S_t = (b_t, q_t, e_{t+1})$ to its neural networks. Where b_t is a sequence variable that represents the network throughput measurement for the past time k; q_t is the delay gradient sequence data of the past time k, which represents the difference of each latency.

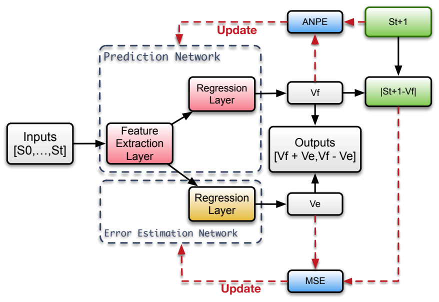

Figure 3: Neural Network Architecture

Additionally, e_{t+1} is the delay gradient on demand for the next time $t + 1$, which is computed by the delay filter.

Prediction Network Prediction network (PN) aims to forecast the baseline value of future observations by using historical time series observations. PN uses a neural network to make logistic regression. Furthermore, the neural network is divided into two layers, feature extraction layer, and linear regression layer. The feature extraction layer is used to extract features from inputs, and the linear regression layer is proposed to estimate the outputs with a fully connected layer. In particular, the feature extraction layer is also the inputs of EEN. For each S in inputs, it is permitted to be multidimensional. The outputs S_{t+1} of PN is a linear value that falls into $(-\infty, +\infty)$, representing the predicted value for the time $t + 1$, which is also the baseline value of the proposed neural network.

Error Estimation Network The Error Estimation Network (EEN) is used to estimate the absolute value that PN generates. The inputs of EEN is the outputs of feature extraction layer in PN. Same as PN's outputs, a linear value is proposed to describe the error in PN, which is the variable value of the proposed neural network.

Outputs Upon receiving S_t , rate estimator needs to predict the value corresponding to the throughput S_{t+1} for the given delay gradient e_{t+1} of next time $t + 1$. In our model, the value is represented by a range from minimum value to maximum value which is inflected by the output of PN and EEN, respectively.

4.2.2 Training Methodology. We now describe how to train and optimize the neural network architecture. The pseudo-code for training methodology is given in Algorithm 1. PN and EEN update

Algorithm 1 Neural Network Gradient Training Algorithm

```
Input: Historical observations $[S_0, S_1, \dots, S_t]$,
       Prediction observations $S_{t+1}$, Learning rates $\alpha, \beta$
1: procedure UPDATE GRADIENT($[S_0, S_1, \dots, S_t], S_{t+1}$)
2:     $\hat{S}_{t+1} \leftarrow PN.Predict([S_0, S_1, \dots, S_t])$;
3:     $Loss_{pred} \leftarrow ANPE(S_{t+1}, \hat{S}_{t+1})$;
4:     $E_{t+1} \leftarrow |\hat{S}_{t+1} - S_{t+1}|$;          ▷ ground truth of EEN
5:     $H_{pred} \leftarrow PN.FeatureExtractionLayer()$;
6:     $\hat{E}_{t+1} \leftarrow EEN.Predict(H_{pred})$;
7:     $Loss_{err} \leftarrow |E_{t+1} - \hat{E}_{t+1}|^2$;
8:     $PN \leftarrow PN - \alpha\nabla_{pred}Loss_{pred}$
9:     $EEN \leftarrow EEN - \beta\nabla_{err}Loss_{err}$
```

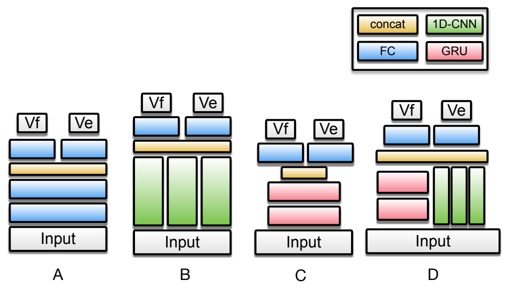

Figure 4: Architecture Overview For Each Candidate

their gradient respectively during the training process. After receiving $S_{0\ldots t}$, PN predicts the future observation $\hat{S}_{t+1}$. By comparing the ground truth value S_t, we can optimize PN. To evaluate the difference between value predicted and the ground truth, we use absolute normalized prediction error (ANPE), which is described in Eq.5. After calculating ANPE, PN then updates its gradients for minimizing ANPE by using back propagation method. While updating the gradient of PN, the absolute error between value predicted and true value (Eq.4), which is denoted as E_t, will be used as a ground truth of EEN. After that, EEN will use mean square error (MSE) to update its network gradient.

$$ANPE(y, \hat{y}) = \left|\frac{y - \hat{y}}{\hat{y}}\right| \tag{4}$$

$$E_t = |y_t - \hat{y_t}| \tag{5}$$

4.2.3 Best Architecture Selection. To select the best architecture, we derive four different deep neural network architectures for the rate estimator as depicted in Figure 4, named as Arch-A to Arch-D, respectively. Arch-A considers network variable features as a common feature, so we only use two fully-connected layers to design its architecture. Arch-B is similar except that some of network metrics are assumed as a signal input, and more internal and hidden features can be shown by convolution neural network (CNN). The next two architectures consider the temporal properties. The third design (Arch-C) uses GRU, a well-known recurrent neural network (RNN), to extract the hidden features of time series. The last design (Arch-D) is a hybrid neural network architecture which combines both Arch-A and Arch-C with a merged layer. We test and compare the prediction accuracy with four architectures for selecting the best one. We collect a small scale of network status as the dataset from real-world network environment for training and validation. In detail, we leverage ANPE (Eq.5), error estimation rate (EER) and coverage rate (CR) to analyze the estimation accuracy of each architecture. EER is defined as follows w.r.t. root mean square error (RMSE) (Eq.6).

$$EER = 1.0 - \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left|\hat{E_i} - E_i\right|^2} \tag{6}$$

$$CR = \frac{1}{n}\sum_{i=1}^{n} P_i, where\ P = \begin{cases} 1 & \hat{y_i} \in [V_f - V_e, V_f + V_e] \\ 0 & \hat{y_i} \notin [V_f - V_e, V_f + V_e] \end{cases} \tag{7}$$

Where $\hat{E_i}$ represents the estimated throughput, E_i is the future throughput observed, and n is the size of test dataset. CR is employed in a ratio to estimate the coverage of throughput range

Architecture	ANPE	EER	CR %
Arch-A	0.59	0.14	100.0
Arch-B	0.68	0.98	94.0
Arch-C	0.49	0.99	92.3
Arch-D	**0.71**	**0.99**	**95.2**

Table 1: Results for Bounded Neural Network Architectures

during the session (Eq. 7). After training and testing, the final results are shown in Table 1. In short, by comparing integrated among ANPE from PN, EER from EEN and CR from range predicted of each network, we summarize their performance as follows:

- Despite the outstanding performance of CR by using Arch-A, the EER metric in a small value indicates that its error estimation network has not been converged completely. Therefore, ARCH-A cannot adequately predict the error range.
- Comparing the overall performance with Arch-B, Arch-C, and Arch-D, we find that Arch-D has better overall performance.

Finally, we choose Arch-D, a hybrid neural network to implement the neural network in rate estimator.

4.2.4 Implementation. In conclusion, rate estimator passes $k = 8$ past throughput to the feature extraction layer, which consists of a convolution network (CNN) and a recurrent network.The convolution network passes $k = 8$ past throughput to the convolution layer with 64 filters, each of size 3 with stride 1. Past delay gradient measurement is passed to another 1D-CNN with the same shape. Results from these layers are then aggregated with other inputs in a hidden layer with 64 neurons, which can be regarded as the hyper feature of inputs. The recurrent network passes $k = 8$ throughput and delay gradient measurement to a gated recurrent unit(GRU) layer with 64 hidden units, then the states of that layer are passed to another GRU layer with the same hidden units. A hidden layer is defined as the hidden output of the last GRU layer. These two hidden layers are finally merged into one layer, which is the inputs of EEN. Then the neural network is divided into two parts. Similarly, both parts use the final output as a linear neuron, but their duties are quite different. For PN, it updates the gradient from the network inputs to the prediction outputs. For EEN, it only updates the gradient from the extracting layer to error estimation outputs.[2]During the training process, learning rates α, β for PN and EEN are configured to be 0.000625 and 0.001 respectively. In our experiment, we use single GPU GTX1080Ti to train our model, and our neural network converges within 100 epochs. Finally, we use TensorFlow[1] to implement this architecture, in particular, we leveraged the TFLearn deep learning library's TensorFlow API to declare PN and EEN.

4.3 Rate Control

We now consider how to combine the outputs of the rate estimator with the video encoder parameters. The video encoder requires a smooth bitrate range rather than a bitrate value, so we set video encoder type as constrained variable bit rate (CVBR) . In this type, encoder needs $bitrate_{max}, bitrate_{min}, bitrate$ as inputs, so after receiving the feedback message which contains the V_f, V_e, we set

[2]Details of the neural network architecture can be found at https://github.com/godka/bounded-neural-network.

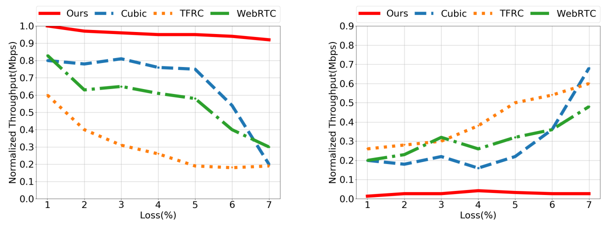

Figure 5: Comparing our method with several proposed methods, results are collected under different loss ratio in the simulation environments.

$bitrate_{min} = V_f - V_e, bitrate = V_f, bitrate_{max} = V_f + V_e$. Although the bitrate range can be controlled by changing α in the delay filter module. In practice, we also use a moving average threshold to protect the encoder from switching frequently in stationary network environments.

5 EXPERIMENTS AND RESULTS

In this section, we establish a real-time video streaming system to experimentally evaluate our rate control method. We compare our approach with current widely used methods on a wide range of network conditions. Our results answer the following questions:

(1) What is the best coefficient α in our scenario?
(2) By comparison of different schemes, how much impact does range have on the results?
(3) Compared with previously proposed schemes, how does our approach perform on the throughput, latency, and their stability performance?

5.1 Dataset

To train rate estimator module, we collect a large number of network status dataset. For this reason, we use a proprietary dataset of packet-level live-cast session status from all platforms APPs of Kwai collected in January 2018. [3] The dataset consists of over 14 million sessions from 47,000 users covering 50 thousand unique sessions over three days in January 2018. In each session, a receiver set up a KTP (Kwai Transport Protocol) connection with a sender and pull video stream. Within each session, we collect network status of the stream in packet-level including send time, receive time, receive packet size, and session id.

5.2 Simulation Environment Experiments

In this experiment, we aim to evaluate the average throughput and performance stability of several state-of-the-art rate control methods in some simulated network environments, and by using Network Emulator for Windows Toolkit, we can simulate different environments. After running five minutes for each approach, we collect the average throughput from the receiver. We compare their performance under different loss ratio. In this experiment, the performance is compared with TFRC, a hybrid loss-based approach[8], Cubic[7], a conventional loss-based approach and Google Congestion Control[4], a hybrid delay-based and loss-based approach that is used in WebRTC. Normalized throughput results have been illustrated in Figure 5, which has demonstrated that the performance of our proposed method outperforms the widely used approaches, with improvements in average packet loss and latency of 10% to 83%, especially in error-prone environments, when packet loss bursts, our approach remains high throughput and stability.

α	B_T(%)	$\sigma(T)$	$\bar{L}$(ms)	$\sigma(L)$
0	83.3	0.0888	10.7279	0.6904
0.6	84.3	0.0868	10.6047	0.7126
1.2	**84.7**	0.084	**10.4435**	**0.2868**
1.8	83.2	**0.0832**	10.7916	0.6046
2.4	82.8	0.0867	10.5954	0.5552

Table 2: Comparing performance of the delay filter that is initialed by different coefficient α with throughput and latency measurement on the network emulator. Results are collected under the fixed bandwidth = 4Mbps, loss rate = 10% and latency = 100ms respectively.

5.3 Real Testbed Experiments

Starting from scratch, as depicted in Figure 1 we set up a testbed where the sender uploads live stream captured by the camera to the receiver via UDP protocol. Our rate control approach is set up on the receiver. The sender has two modules: one is a video encoder which encodes video as H.265 baseline profile; the other is the packet transmission module that transfer the stream encoded to the receiver. The receiver is composed of two main parts, packet reception module, which receives video streaming from the sender and measures its network status, and rate controller, which uses hybrid neural network architecture to constrain the end-end delay. With theoretical analysis and recent experiments, we set time-slot t to 1.0s, and target delay is equal to the latency that is measured from the first time-slot.[4] The evaluation of rate control algorithm is split into three experiments.

5.3.1 Comparison Of Different Coefficient α. In this part, we design an experiment to confirm the best coefficient α to optimize the delay filter module. We compare throughput and latency with different α in the same network environment. Two representative experiments are designed to evaluate our proposed approach. The result is shown in Table2, which includes bandwidth utilization and standard deviation of throughput ($B_T, \sigma(T)$) and latency ($\bar{L}, \sigma(L)$) by the delay filter. The delay filter is started with $\alpha \in \{0, 0.6, 1.2, 1.8.2.4\}$. In our experiment, we set α as 1.2 to estimate next delay gradient on demand.

$$\begin{aligned} USI = & 2.15 \times \log(bitrate) - \\ & 1.55 \times \log(jitter) - 0.36 \times RTT \end{aligned} \tag{8}$$

5.3.2 Effectiveness of Range Factor. We evaluate the effectiveness of our range. We compare our method with the ones with fixed range and the one without range, using the Users Satisfaction Index (USI) [5] defined as the QoE metric (Eq.8). Where the bitrate is the average video bitrate, jitter is the delay gradient, and RTT is the round-trip time between the sender and the receiver. We use two fixed range methods as baselines, which are 100Kbps and 500Kbps. The result is plotted at Figure 6. From the result, we can see that

[3]Kwai is a leading platform in China which has over 700 million users worldwide, and millions of original videos are published on it every day.

[4]In this paper, we focus on how to get high video bitrate and bandwidth utilization with delay-constrained instead of estimating target delay.

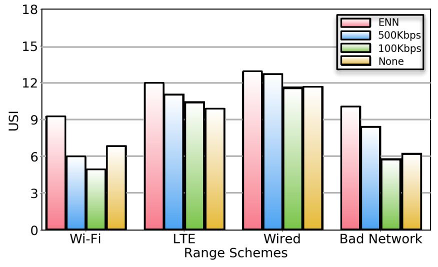

Figure 6: Comparing with the USI score of different ranges, results are collected under various network conditions, such as Wi-Fi, LTE, wired, and "very bad network".

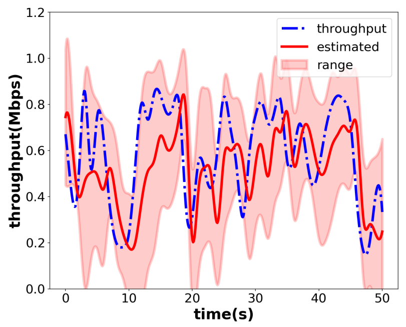

Figure 7: Comparing the accuracy of neural network using range with the one without using range, the result shows that estimating throughput in range can be more conducive to describe the future observations.

dynamic range method from EEN performs better than any other range schemes. In particular, our approach using range works well in "very bad network"[5], and "Wi-Fi" environments, which improves 35% and 19% in USI respectively. The output of our rate control approach is shown in Figure 7, and comparing with the one without using range, our method is more conductive.

6 CONCLUSION

In this paper, we focus on rate control method in the real-time live streaming scenario. Previously proposed approaches are all devoted to finding a single precise prediction value which can not well adapt to the stochastic dynamics of network conditions. To solve this open issue, we describe future observations as a range instead of a value. In this paper, we design a delay-constrained rate control approach based on end-end deep learning. The proposed method predicts a future throughput range with both previously network status and future delay required. To optimize its accuracy in different network characteristics, we train this on a large number of real-time video streaming data. In the experiment, our approach is deployed on the receiver, and by comparing with the state-of-the-art methods and the method without using range, our method all shows a better performance in bandwidth utilization. The study also concludes that predicting an optimal range performs better than predicting a value.

Additional research may focus not only on the real-time live streaming scenario but also on any time-series forecasting environments.

Acknowledgments. We thank the anonymous NOSSDAV reviewers for their valuable feedback. This work was done in close cooperation with Kwai Technology Co., Ltd, and was part-funded by the National Natural Science Foundation of China under Grant No. 61472204, 61521002, Beijing Key Laboratory of Networked Multimedia (Z161100005016051).

[5]Very bad network is a network profile in Network Link Conditioner (a tool which is provided by Xcode), which describes the network status under the bandwidth = 1Mbps, Loss rate = 10% and latency = 500ms.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.

[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] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-based congestion control. *Queue* 14, 5 (2016), 50.

[4] 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.

[5] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. 2006. Quantifying Skype user satisfaction. In *ACM SIGCOMM Computer Communication Review*, Vol. 36. ACM, 399–410.

[6] Yufeng Geng, Xinggong Zhang, Tong Niu, Chao Zhou, and Zongming Guo. 2015. Delay-constrained rate control for real-time video streaming over wireless networks. In *Visual Communications and Image Processing (VCIP), 2015*. IEEE, 1–4.

[7] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[8] Mark Handley, Sally Floyd, Jitendra Padhye, and Jörg Widmer. 2002. *TCP friendly rate control (TFRC): Protocol specification*. Technical Report.

[9] David Andrew Hayes and David Ros. [n. d.]. Delay-based congestion control for low latency.

[10] 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.

[11] 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.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[13] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 197–210.

[14] K. Nihei, H. Yoshida, N. Kai, D. Kanetomo, and K. Satoda. 2017. QoE maximizing bitrate control for live video streaming on a mobile uplink. In *2017 14th International Conference on Telecommunications (ConTEL)*. 91–98. https://doi.org/10.23919/ConTEL.2017.8000044

[15] 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. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 3. 1337–1345 vol.3. https://doi.org/10.1109/INFCOM.1999.752152

[16] Dario Rossi, Claudio Testa, Silvio Valenti, and Luca Muscariello. 2010. LEDBAT: The New BitTorrent Congestion Control Protocol.. In *ICCCN*. 1–6.

[17] Dapeng Wu, Yiwei Thoms Hou, and Ya-Qin Zhang. 2000. Transporting real-time video over the Internet: Challenges and approaches. *Proc. IEEE* 88, 12 (2000), 1855–1877.

[18] Hiroshi Yoshida, Kozo Satoda, and Tutomu Murase. 2013. Constructing stochastic model of TCP throughput on basis of stationarity analysis. In *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 1544–1550.

[19] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K Sinha. 2015. Can accurate predictions improve video streaming in cellular networks?. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 57–62.