A Super-Resolution Flexible Video Coding Solution for Improving Live Streaming Quality

Qing Li, *Member, IEEE,* Ying Chen, Aoyang Zhang, Yong Jiang, *Member, IEEE,* Longhao Zou, *Member, IEEE,* Zhimin Xu, and Gabriel-Miro Muntean, *Senior Member, IEEE*

***Abstract*—In the context of the latest growing popularity of live video streaming, ensuring high video quality has become one of the most significant challenges faced by all live streaming platforms. Insufficient uplink bandwidth is an important factor that influences these live video transmissions, affecting their bitrate and latency and consequently the associated video streaming quality. This paper proposes a novel flexible super-resolution-based video coding and uploading framework (FlexSRVC) that improves the quality of live video streaming in limited uplink network bandwidth conditions. FlexSRVC includes a flexible video coding scheme, which compresses high-resolution key and non-key video frames to a lower bitrate in order to reduce the upload delay. A new flexible bitrate adaptation algorithm is also proposed to select dynamically the number of frames to be compressed and the compression ratio by jointly considering uplink network conditions and available cloud computing resources. Trace-driven emulations demonstrate that FlexSRVC provides the same quality while reducing up to 25% of the required bandwidth compared to the original encoding method (H.264). FlexSRVC improves users' QoE by at least 50% compared to a super resolution-based method which employs reconstruction of all video frames in uplink bandwidth constrained conditions.**

***Index Terms*—video delivery, live streaming, super-resolution, video coding**

I. Introduction

Recently, crowdsourced live video streaming has experienced an unprecedented global growth. A report from Cisco predicts that live video traffic will take up 17% of Internet video traffic by the end of 2022 [1]. This rapid increase in crowdsourced live video streaming services is highly positive for all stakeholders, attracting growing numbers of broadcasters and active viewers alike, but is significantly challenging the current network infrastructure, especially at network edge. In the context of crowdsourced live video streaming, broadcasters come from all over the world, and they distribute their video content to large number of viewers located worldwide. Low latency and high throughput are the most critical requirements for high-quality live video streaming and expected success of such services. However, the uplink bandwidth of a live broadcaster fundamentally constrains live video quality. The bandwidth required to transmit a 1080P HD online video is at least 6 Mbit/s [2], whereas the average uplink transmission rate of commercial live streaming servers is currently less than 1 Mbit/s [3]. In case there is insufficient uplink network bandwidth, live broadcasters cannot provide high-quality video, leading to poor video quality of the entire live streaming service. If high-quality video is uploaded under limited uplink bandwidth conditions, video rebuffering and loss may affect the transmission, influencing negatively the overall user quality of experience (QoE). Therefore, ensuring high throughput and low latency in limited uplink bandwidth network environments has become a significant challenge for live video streaming.

Significant efforts have been put to improve the quality of live video streams including by proposing innovative content delivery network (CDN) scheduling solutions [4] and adaptive bitrate algorithms [5], [6]. However, if the uplink bandwidth is limited during content delivery from broadcasters to the cloud, methods that focus on downlink quality improvement cannot help. Recently, in order to cater for situations with insufficient network bandwidth, Super Resolution (SR) techniques have been introduced to increase the quality of low resolution video transmissions. For instance the Deep Neural Networks (DNN)-based SR solution proposed in [7] achieved superior image restoration performance compared with a traditional up-sampling method, i.e. bicubic. The block-patching-based image post-processing solution (BIPP) [8] employed DNN-based SR in order to obtain high video compression ratio and reconstruct non-key frames in the video. The live neural adaptive streaming scheme (LiveNAS) [7] uploaded low-resolution videos in the cloud and used DNN-based SR to reconstruct all video frames, improving the video quality of live streaming. However, the SR long processing time results in high latency, making it hard to satisfy the stringent delay requirements of live video transmissions [9]. Therefore, there is an important interest in finding solutions for reducing the latency caused by limited network bandwidth during broadcaster video upload and video processing in the cloud, while continuing to enhance the overall experience of live streaming.

This paper proposes **a flexible SR-based video coding and uploading framework (FlexSRVC)** which reduces the upload latency from broadcasters to cloud servers and enhances the overall live streaming quality in limited uplink network

Q. Li is with Department of Mathematics and Theories, Peng Cheng Laboratory (PCL) , Shenzhen, China (Email: liq@pcl.ac.cn)

A. Zhang and Y. Chen are with the Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, Shenzhen, China (Email: zhang-ay18@mails.tsinghua.edu.cn; chen-yin18@mails.tsinghua.edu.cn).

Z. Xu is with the Multimedia Lab in Beijing Bytedance Technology Co., Ltd., Beijing, China (Email: xuzhimin@bytedance.com).

L. Zou is with the Department of Broadband Communication, Peng Cheng Laboratory (PCL) and also with Southern University of Science and Technology, Shenzhen, China (Email: zoulh@pcl.ac.cn).

Y. Jiang is with the Division of Information Science and Technology, Tsinghua Shenzhen International Graduate School and also with Department of Mathematics and Theories, Peng Cheng Laboratory (PCL), Shenzhen, China (Email: jiangy@sz.tsinghua.edu.cn).

G.-M. Muntean is with the Performance Engineering Lab, Dublin City University, Ireland (Email: gabriel.muntean@dcu.ie).

Corresponding authors: Longhao Zou (zoulh@pcl.ac.cn)

bandwidth conditions. FlexSRVC includes a flexible frame video coding scheme proposed to compress the key and non-key frames of high-resolution live streams to lower bitrates. This efficiently reduces the amount of data to be uploaded and consequently the associated upload delay. An innovative aspect is provided by an adaptive selection of key and non-key frames for compression, enabling flexible adaptation to dynamic network changes. FlexSRCV also includes a new flexible bitrate adaption algorithm designed to dynamically select the compression ratio and the number of frames to be compressed by jointly considering uplink network conditions and availability of cloud computing resources. By employing FlexSRVC broadcasters can offer high quality video experience by performing low bitrate live streaming in limited uplink network bandwidth conditions, effectively breaking the classic strong dependency between the uplink bandwidth and live video quality.

Extensive trace-driven simulations were conducted to evaluate FlexSRVC. The results show how FlexSRVC can save up to 25% of bandwidth in comparison with a classic live streaming method, while achieving the same video quality. The proposed flexible frame coding scheme was also compared to other state-of-the-art SR-based coding methods. The results show that the compression rate of our coding method is on average 10% higher than those of alternative solutions. Compared to the method of reconstructing all frames with SR, FlexSRVC improves user QoE by at least 50%. These results show how FlexSRVC and its components outperform other solutions and consistently achieve high live video streaming quality.

This paper contributions are as follows:

- **A novel video SR-based coding and uploading framework:** FlexSRVC is proposed to reduce the upload latency from broadcasters to cloud servers and enhance the overall viewing experience of live video streaming.
- **A novel flexible SR-based video coding scheme:** A SR based flexible frame compression scheme is introduced to accommodate live streaming under dynamic network conditions. This scheme compresses key and non-key frames of a high resolution live stream to a lower bitrate, which efficiently reduces the size of the video stream and hence decreases subsequently the upload latency. The key and non-key frames are adaptively selected for compression, providing flexibility when adapting to network changes.
- **A flexible bitrate adaptation algorithm:** A new flexible bitrate adaption algorithm (FLBA) is designed to select the video bitrate and frame compression ratio by considering the dynamics of the uplink bandwidth and cloud server computing capabilities. The algorithm goal is to further improve the quality of the live video streaming.

The next section of the paper presents background knowledge and discusses major related works. Section III describes the FlexSRVC framework design and section IV presents the proposed solution evaluation, including simulation testing setup, scenarios and result analysis. The final section concludes the paper and presents future work directions.

II. Background and Related Works

A. Crowdsourced Live Streaming

In a crowdsourced live streaming service, live video streams are being published and watched at any time, from any location, and under any network environments. Due to the complex environment and various network conditions, transmitting high quality video streams with low latency is a fundamental challenge for live video streaming services. Several efforts have been put to design solutions to improve the quality of live video stream delivery. Traditional real-time rate control methods include sender-based adaptation solutions which take into consideration various objectives such as fairness, estimated quality [10], energy efficiency [11], load-balancing [12] and user QoE levels [13] when adjusting the video delivery in order to match network dynamic situation. More recently, client-driven adaptive solutions including MPEG-DASH-based approaches such as those proposed by Yaqoob et al. [14] and Zou et al [15] were proposed. Live video streaming specific solutions include loss-based bitrate approaches [16], delay-based bitrate solutions [17] and model-based bitrate schemes [18]. Zhang et al. [4] proposed a deep reinforcement learning-based approach to deal with the resource scheduling problem in crowdsourced live streaming. The method proposed in Bakar et al. [19] introduced an adaptive video layer selection scheme based on the VP9 scalable video coding over WebRTC for low latency streaming. However, such methods that focus on downlink improvement or balancing real-time sending rate may not help if the uplink network bandwidth is limited during content delivery from broadcasters to the cloud, so alternative solutions need to be designed.

TABLE I: Comparisons between EDSR and CARN (vimeo90k-bi dataset)

Model(upscaling)	PSNR(dB)	SSIM	Parameters	Inference time(s)
CARN($\times$2)	35.14	0.94	257,027	0.0194
EDSR($\times$2)	35.66	0.93	1,369,859	0.0194
CARN($\times$3)	31.55	0.86	303,427	0.0176
EDSR($\times$3)	32.06	0.87	1,554,499	0.0166
CARN($\times$4)	29.66	0.80	294,147	0.0164
EDSR($\times$4)	30.16	0.81	1,517,571	0.0157

B. Super-Resolution (SR)-based Solutions

SR is a computer vision approach which reconstructs high resolution (HR) images/videos from low resolution (LR) images or videos [20], [21], [22]. Traditional SR methods, such as bicubic interpolation [23], double cubic [24] and Lanczos resampling [25], are very fast and simple, but are limited by their low accuracy. In order to improve the perceptual quality of the consecutive video frames, SR methods based on Generative Adversarial Networks (GAN) were introduced such as Super Resolution Generative Adversarial Networks (SRGAN) [26] and Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) [27]. A more comprehensive task-driven Video Restoration with Enhanced Deformable Convolutional Networks (EDVR) that consists of multiple stages including super-resolution, deblurring, denoising and de-blocking, was proposed in [28]. However, the video SR schemes mentioned above need a very large number of neighbor frames to be inputted together into the super-resolved

model, which would increase the amount of uplink video streaming traffic at the broadcaster side. In addition, the model parameters of ESRGAN and EDVR exceed 16 million and 20 million, respectively, putting pressure in terms of computational complexity. Recent methods (i.e., Efficient Sub-pixel Convolutional Neural Network (ESPCN) [29], Enhanced Deep Super-Resolution Network (EDSR) [30], and Convolutional Anchored Regression Network (CARN) [31]) use deep neural networks (DNN) to analyze the statistical relationship between LR and their corresponding HR counterparts from a large number of training examples, achieving significant improvement in terms of reconstruction quality. DNNs cannot be applied easily to all real-world applications due to their heavy computation requirements.

We have evaluated diverse situations when EDSR and CARN were employed and Table 1 presents the comparative results obtained. The models are trained and tested with the vimeo90k-bi dataset on Nvidia GeForce 2080Ti, and results for several scaling options (i.e. $\times 2$, $\times 3$ and $\times 4$) are presented for analysis. We found that both EDSR and CARN achieve very similar results regarding image quality scores (i.e. in terms of both PSNR and SSIM) and inference time. However, the number of model parameters of CARN is much lower than those of EDSR. This reduces much the computational requirements in terms of cloud-based or edge-based GPU hardware [32]. To reduce computation overhead while ensuring good performance, CARN employs an accurate and lightweight DNN for image SR with a cascading residual network. Therefore, this paper adopts a CARN-based approach in the cloud server for SR in the quest to meet the real-time and quality requirements of live video streaming.

C. SR-based Hybrid Coding Schemes

Video compression tries to achieve the best trade-off between video quality and bitrate. Because of the limited network bandwidth for data transmission and low storage capacity, very good video compression is critical for the support of low bitrate delivery applications. Many studies have focused on developing an efficient coding method based on SR to improve the low bitrate compression efficiency of high quality videos [8], [33], [34]. These methods adopt a strategy that downsamples frames before encoding and upsamples them after decoding. Early in in Shen et al. [35], the proposed super-resolution model was in fact the classic non-deep-learning method, which is based on the classic Local Linear Embedding (LLE) algorithm. A video sequence is divided into key frames and non-key frames adaptively in BIPP [8]. BIPP proposed an adaptive downsampling-based coding model to improve video compression ratio while ensuring good video quality. Key frames are encoded at high resolution at the encoder, whereas non-key frames are downsampled at a lower resolution. A proposed scheme in Liu et al.[36] aims to super-solve the CTU-level residue based on the intra-frame prediction signal generated by the HEVC encoder, which in general incurs more blocking artifacts. A DNN-based SR is employed to reconstruct non-key frames at the decoder. An SR-based block up-sampling method for intra-frame coding employed in a new convolutional neural network structure for upsampling is described in [33]. However, the methods mentioned above all focus on improving compression efficiency while taking a long time to encode and decode and do not consider the delay problem of actual deployment in a live video transmission system. The video coding methods proposed in this paper use a lightweight SR approach and adaptively select frames according to network conditions, thereby achieving short video processing time and helping reduce the transmission delay.

D. SR-based Video Delivery System

Recent research uses DNN-based SR to enhance quality in video delivery systems. To reduce download latency for video-on-demand systems, NAS [37] and SRAVS [38] employ super-resolution on the client-side on top of an adaptive video streaming solution, which requires strong computing power and high energy consumption of terminal equipment. LiveNAS [7] adopts SR in cloud servers for live video streaming, which provides high-quality live streams, but consumes high computational resources. Noteworthy is that the existing SR approaches apply SR to all frames of low bitrate videos. Reconstructing all video frames in a live transmission is time-consuming, which results in high latency of the video streaming process. This paper proposes a video coding approach that adaptively selects the number of frames to be compressed and upscaled in the cloud server to reduce significantly processing time and reduce the overall live streaming delay.

III. FlexSRVC Framework Design

A. System Overview

This paper proposes FlexSRVC, a new video coding and uploading framework that reduces the upload latency from broadcasters to cloud servers and improves the quality of live video streaming in scarce uplink network bandwidth conditions. FlexSRVC includes a flexible frame video coding scheme which is proposed to encode video chunks flexibly according to network conditions and availability of cloud computing resources. FlexSRVC also employs a novel flexible bitrate adaptation algorithm which is designed to coordinate with the flexible frame video coding scheme as part of the live video streaming system. Figure 1 illustrates the proposed FlexSRVC, which consists of two components: at the live broadcaster and at the smart cloud.

At the Broadcaster: The live broadcasters stream their video content under various network conditions and utilize our flexible frame video coding scheme to encode the captured raw frames with the bitrate and frame compression ratio determined by our *flexible bitrate adaptation algorithm (FLBA)*.

At the smart cloud: The cloud servers, the proposed *Online SR video coding model* is deployed to reconstruct the downsampled frames in the uploaded live stream.

B. Flexible Frame Video Coding Scheme

The video frames in a group of pictures (GOP) are encoded into key frames (KF) and non-key frames (NKF) sequentially. For a conventional video coding standard, i.e., H.264, KFs

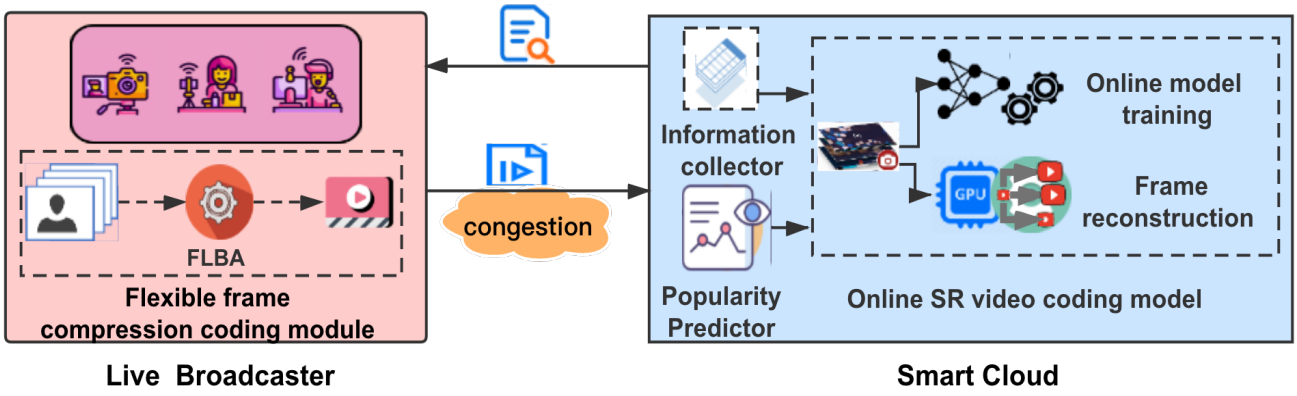


Fig. 1: System architecture

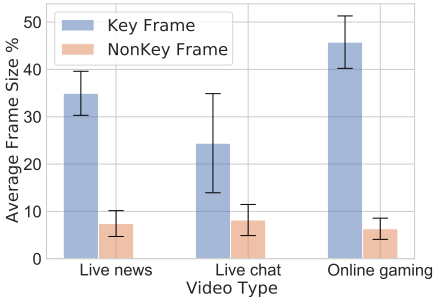


Fig. 2: Frame size ratio

are encoded by an intra-frame coding method, and NKFs are encoded by an inter-frame coding method. Figure 2 illustrates the average size of KFs and NKFs in a GOP for three different video types, respectively. It can be seen that the KF size accounts for most of the GOP size. As NKFs only contain the difference between them and the reference KF, the NKF size is smaller than that of a KF.

In order to further reduce the transmission video bitrate, the frames are not always transmitted at full resolution. Therefore, we design a novel flexible frame video coding scheme based on SR, which downsamples the frames to a desired lower resolution to reduce GOP size. To be flexible in adapting to network changes, the proposed video coding scheme compresses KFs and NKFs of a live video stream flexibly. The proposed scheme improves the compression efficiency and maintains high-quality video at a lower bitrate. Fig 3 and Fig 4 illustrate the stages of the proposed video coding scheme, which consists of two components: the encoding component - deployed at the broadcaster and the decoding component - deployed at the smart cloud.

At the broadcaster raw frames are encoded using the bitrate determined by our flexible bitrate adaptation algorithm most appropriate to the current network conditions and the state of the cloud computing resources. Figure 3 illustrates the frame processing overview at the broadcaster side. First, when the broadcaster acquires the original video frames, they are encoded into GoPs using a classic video encoder (e.g. H264). A high resolution key frame (KF^H) and high resolution non-key frames (NKF^H) are then extracted from a GOP. Next, in order to reduce the bandwidth consumption of the uplink, we further compress the video frames. Specifically, the proposed flexible bitrate adaptation algorithm dynamically selects the number of video frames and video compression ratio based on the current uplink network bandwidth and available computing resources. Further, KF^H and NKF^Hs that are selected for further compression are downsampled to a low resolution key frame (KF^L) and low resolution non-key frames (NKF^L). Next, the downsampled video frames are merged into the original video stream. We choose KF^L as the reference frame to encode NKF^Ls using inter-frame coding. Finally, the original KF^H and NKF^Hs are replaced with newly encoded KF^L and NKF^Ls. Note the proposed scheme selects some $NKFs$ to be downsampled to lower resolution only, whereas the other $NKFs$ preserve their original temporal and spatial characteristics. This improves the video compression ratio with good temporal and spatial information, retaining consistency, while also maintaining good cloud computing efficiency.

At the Smart Cloud, the video stream uploaded by the broadcaster is received. The video is composed of KF^Ls, KF^Hs and NKF^Hs. The compression information (i.e. which video frames in a GOP are compressed and the compression ratio) is also uploaded to the cloud server as metadata. Figure 4 shows how the compressed video stream is reconstructed in the cloud. First the coding information associated with the live video stream is analysed. Based on this information, KF^Ls and NKF^Ls only are extracted and decoded using intra-frame and inter-frame coding, respectively. The NKF^Hs of the video stream are not processed yet. KF^Ls and NKF^Ls are upsampled to their original resolution by employing the lightweight SR model CARN [31] with low latency. Finally, the SR-upsampled video frames are re-inserted into the video stream, making sure the smoothness of video playback is retained, so that the viewer does not perceive any coding processing. Specifically, the upsampled KF^Hs are used as reference frames to encode NKF^Hs. As other non-key frames are already at high resolution, $NKFs$ are remultiplexed with the preserved temporal and spatial details into the encoded video stream. The entire video coding process is such performed so that the player can decode the live video content without any issues.

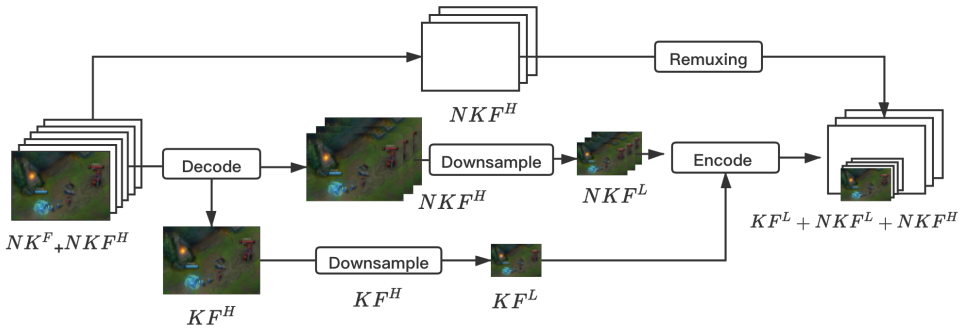


Fig. 3: Flexible frame processing at the broadcaster side

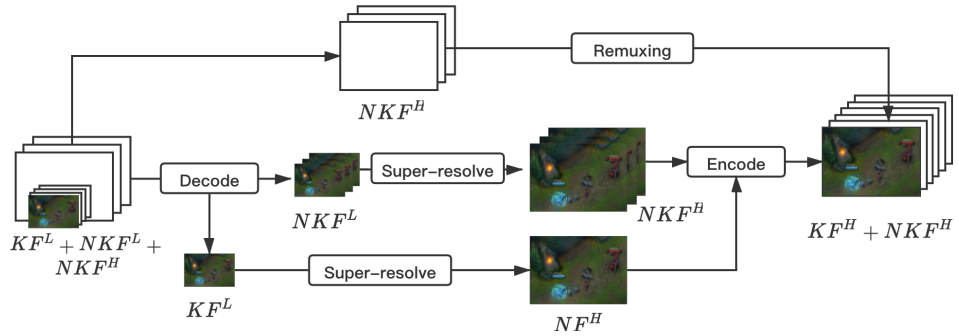


Fig. 4: Flexible frame processing at the cloud side

C. Online SR Video Coding Model

The smart cloud-located Online SR video coding model deploys the proposed SR video coding scheme. As any SR-based solutions, it requires training for accurate functionality. Unfortunately a single offline training session for the Online SR video coding model cannot guarantee a good reconstruction performance for all live broadcast scenes. A better method is to train the SR model for the live video content to ensure high perceptual quality of the reconstructed video frames. Due to the unique characteristics of live streaming, we cannot obtain all the content in advance, and aspects such as scene switching during the live streaming may occur at any time, i.e., switching from a game scene to a game commentator. The obvious picture difference determined following scene switching will lead to quality degradation in the reconstruction of the new scene using the SR model trained in the previous scene. Instead, we conduct an online training of the SR model with actual live content to improve the SR reconstruction quality.

First the impact of various training methods on model performance is studied. Consider $\mathbf{A}$ a set of one second high resolution video chunks captured at the beginning of a live video stream, and $\mathbf{B}$ a set of video chunks obtained after scene switching. We compare three training methods as follows: **A**: only set $\mathbf{A}$ is used for training; **A+B**: first set $\mathbf{A}$ is used and after a scene change is detected, set $\mathbf{B}$ is added and **A+aB**: similar to **A+B**, but set $\mathbf{B}$ has a greater weight a, as dataset B reflects the current status of the live video better [7].

Figure 5 illustrates comparatively the online learning performance with an online gaming video for the three methods. The performance is expressed in terms of the newly proposed quantitative indicator Video Multimethod Assessment Fusion (VMAF)[39], which estimates user perceptual video quality levels. There was an obvious switch in the live video scene around epoch 12500. As shown in Figure 5, as the training time increases before the scene switching occurs, the effect of video enhancement increases, and training gain tends to become saturated. After the scene switching occurred, the reconstruction quality of the video dropped rapidly. Comparing the three curves, we see that **A+aB** and **A+B** methods obtain significantly higher VMAF values than when employing training method **A**. Noteworthy is that by increasing the weight of the new video frames added during training, **A+aB** achieved the highest VMAF. Therefore, in this work, an **A+aB** approach is adopted to ensure the best quality enhancement of the proposed SR model. The cloud and broadcaster jointly assist the online model training: the cloud server regularly checks the video quality enhanced by the SR model, whereas the broadcaster detects scene changes during encoding. When the training gain tends to zero, the online training is paused and when a scene switching is detected, the online training is restarted.

Online incremental training provides better video enhancement effects than directly using offline pre-trained general models. However, online training and inference for each channel is a high computing resource-consuming task. Based on our observations, the popularity of live broadcasters is highly heterogeneous, with the highest proportion of traffic being generated by the few most popular live broadcasters. Figure 6a shows the popularity of live broadcasters crawled from Huya (one of the biggest live streaming platform in China) on May 25, 2020. The popularity of live broadcasters is highly skewed. Therefore, if the maximum processing power is used on the top channels, the most benefit for viewers is

obtained from the limited computing power available. The difficulty in exploiting the skew lies in quickly and accurately predicting the popularity of individual channels. We examined the access pattern of 3 representative channels, ranking 1, 4, and 7, respectively. As shown in Figure 6b, the streams of the same live broadcasters have similar peak popularity for different days. Based on the finding, the popularity of the video streams broadcasted by the same broadcaster is predictable. When a known live broadcaster has uploaded at least one video stream and uploads a new video stream, the popularity of the new video stream can be accurately predicted based on the earlier view counts.

Due to limited computing resources, we divide live channels into three categories according to their popularity (the number of views): *hot*, *normal*, and *cold*. We adjust the criteria for assigning videos into different categories dynamically according to cloud computing power availability. To meet the real-time transmission requirement, we adopt a well-known light-weight SR model CARN [31]. We provide discriminate computing resources to support each category: 1) For a *hot* channel, the cloud server trains the SR-based models based on the live stream content online and uses the trained model to enhance the video quality; 2) For a *normal* channel, only online quality enhancement is provided to improve quality, without online model training; 3) For a *cold* channel, no computing power is provided for training and enhancement. Two offline-trained SR models adopted for the *normal* channel are: a specific model (SP-similar) trained by other *hot* channels with similar content in the same category and an offline pre-trained general model.

D. Flexible Live Bitrate Adaptation Algorithm (FLBA)

The flexible video frame compression proposed to be used during live streaming requires determination of a targeted resolution for the video stream and selection of the appropriate compression ratio to yield high quality and low transmission latency. In this context, the flexible frame video coding scheme provides multiple bitrate options, because it can flexibly choose the number of compressed KFs and NKFs. In order to best maintain high live streaming quality given limited uplink network bandwidth and cloud computing resources, the flexible live adaptation bitrate algorithm (FLBA) is proposed. We integrate two parameters into the proposed FLBA algorithm for QoE optimization: the encoding bitrate R, and the optimal down-sampling scale $scale_n$. When the broadcaster generates a video chunk, FLBA algorithm will be called to make a decision based on the available uplink bandwidth and computing power in the smart cloud.

1) Problem Formulation: The performance of the proposed FLBA algorithm is determined by both uploading and processing of the live stream. Compared to the common streaming system, our proposed flexible coding scheme introduces additional time cost to compress and reconstruct video frames. Next we present a model of the network constraint, latency constraint and users' QoE, and formulate the optimization problem to be solved in the cloud.

Network Constraint: Let C_t denote the uplink throughput measured at the time slot t and C_n denote the average uplink

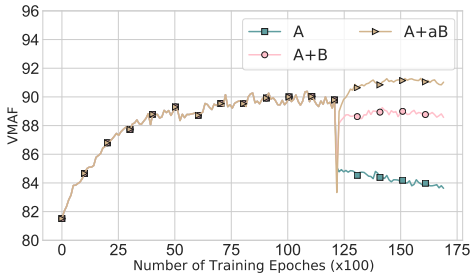


Fig. 5: Online learning quality improvement

throughput during the period of sending the n_{th} video chunk. C_n can be expressed as:

$$C_n = \frac{1}{t_{n+1} - t_n} \sum_{t=t_n}^{t_{n+1}} C_t, \tag{1}$$

where t_n is the starting time sending the n_{th} video chunk and t_{n+1} is the starting time of sending the next video chunk $n+1$. Considering the uplink bandwidth of the live broadcaster, the throughput of the current uploading video stream must be less than the uplink bandwidth capacity. Therefore, the network constraint can be written as:

$$R_n \leq C_n, \tag{2}$$

where R_n is the selected bitrate of the n_{th} video chunk.

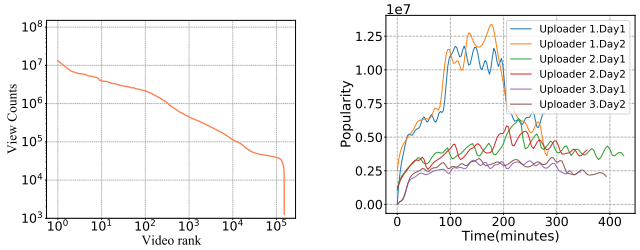
Latency Constraint: In this paper the delay refers to the time spent by the broadcaster uploading a live video chunk to the cloud server until it is distributed to the end users. This is relevant as the focus is on improving live video delivery quality on the video uploading side. The latency is then composed of: 1) the transmission time of uploading a live video chunk from the broadcaster to the cloud server and 2) the video processing time of the n_{th} video chunk in the cloud. The latency L_n can be written as follows:

$$\begin{aligned} L_n &= \frac{S_n}{C_n} + enhance_n * pt_n, \\ enhance_n &\in [0, 1]. \end{aligned} \tag{3}$$

where S_n denotes the size of the n_{th} video chunk. The time taken to fully upload the video chunk is $\frac{S_n}{C_n}$. The video chunk can be either a compressed chunk or an original high resolution chunk. $enhance_n$ is a binary variable to indicate whether the video chunk needs to be processed by the SR technique in the cloud. The video processing time of the n_{th} video chunk is $enhance_n * pt_n$ where pt_n is the estimated total SR processing time according to frame downsample scale $scale_n$.

QoE Model: The ultimate goal of the live bitrate adaptation algorithm is to improve QoE in order to achieve long-term user satisfaction. Recent studies show that user perceived video quality features play a vital role in evaluating the performance of adaptive bitrate streaming services [40]. Motivated by these previous perceptual quality-based studies, we mainly refer to the QoE model defined in [40]. In particular, the QoE of each live stream considered in this paper consists of three aspects as follows:

- Cumulative video quality Q_N: $q(.)$ denotes a non-decreasing function that maps the bitrate of a video



(a) Popularity distribution rank (b) Channel popularity

Fig. 6: Live streaming popularity

chunk R_n to the quality $q(R_n)$ perceived by the user. We use VMAF[39] to measure the perceptual video quality $q(R_n)$. The cumulative video quality of a live video can be written as:

$$Q_N = \sum_{n=1}^{N} q(R_n). \tag{4}$$

- Cumulative video latency L_n: Since we focus on the upload latency from live broadcasters to cloud servers, we only consider the upload latency L_n including the upload time of the n_{th} chunk from broadcasters to cloud servers and the video processing time by using the super-resolution technique. The cumulative video latency of a live video can be expressed as:

$$L_N = \sum_{n=1}^{N} L_n. \tag{5}$$

- Cumulative video quality switching W_N: From a viewer's perspective, switching the video quality from a low bitrate to a high bitrate affects positively the viewing experience, while switching the quality from a high bitrate to a low bitrate decreases experienced viewing quality. Therefore, the QoE model should consider the benefits of switching from a low bitrate to a high bitrate and penalise switching from a high bitrate to a low bitrate. The positive video smoothness WP_P, switching from a low bitrate to a high bitrate, is defined as follows:

$$WP_P = \sum_{n=1}^{N-1} [q(R_{n+1} - q(Rn))]_+. \tag{6}$$

The negative video smoothness WP_N, associated with switching from a high to a low bitrate, is written as:

$$WP_N = \sum_{n=1}^{N-1} [q(R_{n+1} - q(Rn))]_-. \tag{7}$$

However, frequent video quality switches are not desirable for end viewers. To avoid frequent switching, even from a low bitrate to a high bitrate, we set the weight of positive video smoothness to a small value.

Finally, the overall QoE for a live video is the weighted sum of the above three aspects, namely:

$$QoE = \alpha Q_N + \beta L_N + \lambda WP_P - \sigma WP_N, \tag{8}$$

Algorithm 1 Flexible bitrate adaptation algorithm.

Input: $C_{[t_1,t_n]}$, $train_n$, $enhance_n$
Initialize
Target = $QoE(v_n) + \gamma \cdot train_n \cdot M_{gain}(v_n)$
for n = 1 to N **do**
  $\hat{C}_{[t_n,t_{n+K}]} = ThroughputPred(C_{[t_1,t_n]})$
  if $enhance_n = 0$
    $Avail_{set}$ = $\mathcal{R}$
    $1, R_n$ = $f_{mpc}(\hat{C}_{[t_n,t_{n+K}]}, R_{n-1}, Avail_{set})$
    encode n_{th} chunk with bitrate R_n
  elif $enhance_n = 1$
    $Avail_{set}$ = $\mathcal{R} \bigcup \mathcal{R}_{compressed}$
    $scale_n, R_n$ = $f_{mpc}(\hat{C}_{[t_n,t_{n+K}]}, R_{n-1}, Avail_{set})$
    encode n_{th} chunk with bitrate R_n, and downsample the key and/or non-key frames with $scale_n$
  endif
end for

where α, β, λ and σ are non-negative weighting parameters corresponding to video quality, delay, positive quality smoothness and negative quality smoothness, respectively.

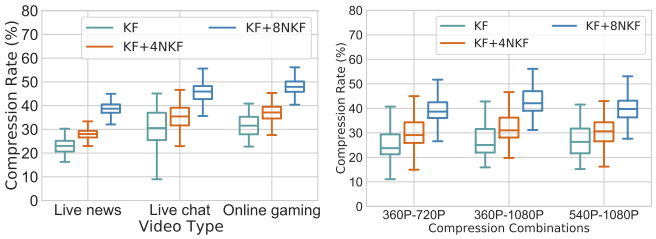
Additionally, we also need to consider the demand for high resolution frames to train the SR video coding model. Thus, our optimization goal is to maximize the overall QoE and the future quality improvement from online SR video coding model training:

$$\begin{aligned} &\arg\max_{scale_n, R_n} QoE(v_n) + \gamma \cdot train_n \cdot M_{gain}(v_n) \\ &\textbf{subject to Eq. (1) - (8)}, \\ &train_n \in [0, 1]. \end{aligned} \tag{9}$$

The first term $QoE(v_n)$ is the user QoE of the n_{th} chunk, and the second term $M_{gain}(v_n)$ is the gradient of the online training quality gain curve, which is provided by the cloud server through periodic feedback. γ is a discount factor of the expected quality gain if the online training model is used for a particular channel. $train_n$ is a binary variable that indicates whether the cloud server provides incremental SR model training, given the capacity of the uplink bandwidth C_n. This optimization generates the following as outputs: 1) encoding bitrate decision R_n, and 2) frame downsample scale $scale_n$.

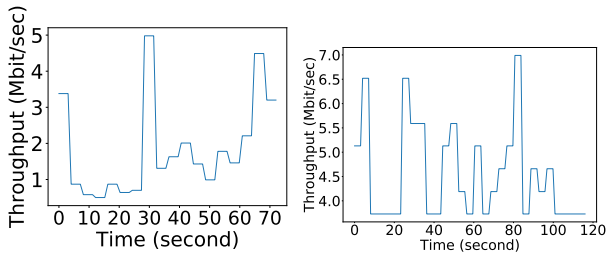
2) FLBA algorithm: As mentioned, the flexible video frame compression offers a greater number of bitrate options than the traditional coding methods since we can flexibly compress key frames and non-key frames. Bitrate adaptation can be considered a stochastic optimal control problem. The Model Predictive Control (MPC) algorithm [41] is one of the most well-known control algorithms for optimizing control problems since MPC optimizes complex control targets online by using predictions in dynamic systems under constrained conditions. The proposed FLBA extends the optimization goal and constraints of MPC to respond to the requirements and constraints of the flexible frame video encoding.

Algorithm 1 presents the proposed FLBA algorithm. The algorithm chooses the bitrate R_n for the n_{th} chunk and the



(a) Compress rate for different video types (b) Compress rate for different resolutions

Fig. 7: Compress rate for proposed encoding scheme



(a) Airport trace (b) Canteen trace

Fig. 8: Self-measured bandwidth

downsample scale $scale_n$ by looking ahead at K future chunks and solves our QoE maximization problem with throughput predictions $\hat{C}_{[t_n,t_{n+K}]}$. For the n_{th} chunk, the broadcaster side maintains a moving horizon from chunk n to $n+K-1$ (i.e. K = 3 during testing).

The algorithm performs the following main updates: line 2 is the objective function based on whether the cloud server will train the specific SR model of the current channel online $train_n$. Line 4 estimates the available throughput $\hat{C}_{[t_n,t_{n+K}]}$ for the next K chunks at time t based on history throughput using a common throughput predictor based on 1D-CNN. According to whether the cloud server has allocated computing power for video stream enhancement $enhance_n$, we decide the available bitrate set $Avail_{set}$ that the broadcaster can encode. If the cloud server provides a certain amount of computing resources to perform the task of recovering compressed KFs and NKFs ($enhance_n$ = 1), the broadcaster can adopt the SR-based video coding method to compress the video stream to $R_n \in \mathcal{R} \bigcup \mathcal{R}_{compressed}$, where $\mathcal{R}$ is the set of bitrates for the original coding and $\mathcal{R}_{compressed}$ represents the bitrate set after flexible frame compression. If there are no computing resources available ($enhance_n$ = 0), the n_{th} video chunk can only be encoded to $R_n \in \mathcal{R}$. Then the encoding bitrate and flexible frame compression ratio is selected according to the estimated uplink bandwidth with the MPC algorithm.

IV. FlexSRVC Framework Evaluation

A. Evaluation Setup

Implementation. At the broadcaster side, we implement our flexible frame video coding scheme by modifying remuxing.c, decode_video.c, encode_video.c and transcoding_video.c in FFmpeg [42]. Our proposed flexible live bitrate adaptation algorithm is implemented by the Pytorch framework. At the

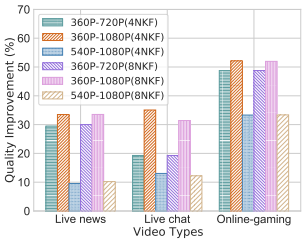


Fig. 9: Quality improvement

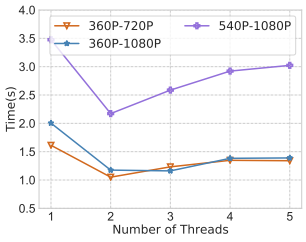


Fig. 10: Multi-thread processing per GOP

cloud server side, we implement the Smart Cloud by employing Nginx [43], uWSGI [44], and Django [45]. The online training process and the inference process are implemented by the Pytorch framework in a separate process. We use two GeForce RTX 2080Ti GPUs on the smart cloud, one for inference and the other one for the training process. We use the lightweight model from CARN [31] for video SR and use the benchmark DIV2K dataset [46] for training offline the pre-trained general model. The weights of the compression ratio, frame SR quality, and SR processing time are set to 0.5, 0.5 and 0.01, respectively.

Videos. We select three different stream categories (live news, live chat, and online gaming) from YouTube. Each video is split into one-second video chunks (GOP size=24). The high-resolution chunk is encoded at a bitrate of {2560, 4800} Kps by H.264 codec, which corresponds to the video resolution {720P, 1080P}. Key and non-key frames can be downsampled from 1080P to 360P/540P and from 720P to 360P, respectively.

Network Traces. We use two types of bandwidth traces: an open dataset (100 4G network uplink traces [47]) and a self-collected dataset. We collected real uplink network bandwidth information in different environments (such as airport, canteen, train and pedestrian) by using the Mobile Intelligent network measurement tool[1]. We recorded uplink throughput every 5000 milliseconds. This measurement effort is ongoing, and we are constantly adding additional traces to our uplink dataset for future research. To simulate the dynamic changes between broadcasters and the smart cloud in a realistic network, we use the Linux Traffic Control tool [48] to control the sending rate on the broadcaster side. In particular, we select network traces with an average uplink bandwidth of less than 2.5Mbps to simulate a bandwidth-constrained environment. We also use our measured network uplink traces for evaluation.

QoE Parameters. In our experiments, the weights of video quality, and quality variations defined in Eq. 8 in this QoE criterion are set to $\alpha = 0.8$, $\beta = 5$, $\lambda = 0.2$ and $\sigma = 1.06$, respectively. The impact of video quality reduction on user viewing is much greater than that of video quality improvement, so we set $\lambda < \sigma$ when setting hyperparameters. The evaluation index of VMAF [39] used to measure the video is [0, 100], the overall distribution of the rebuffering time will be smaller than the value of VMAF. To balance the relationship between rebuffering and video quality, we set the video rebuffering penalty to a large value. In Eq. 9, γ is a discount coefficient less than 1, and its purpose is to balance

[1]MobileIntelligent [Online] https://appsonwindows.com/apk/8486138

TABLE II: Broadcaster Processing Time per GOP (i.e. 1s)

Compression combination	KF + 4NKF (ms)	KF+8NKF (ms)
360P - 720P	180 ± 38.48	201 ± 28.95
360P - 1080P	298 ± 50.08	334 ± 57.58
540P - 1080P	324 ± 38.62	366 ± 48.77

TABLE III: Cloud Processing Time per GOP (i.e. 1s)

Compression combination	KF + 4NKF (ms)	KF+8NKF (ms)
360P - 720P	233 ± 72.81	269 ± 66.34
360P - 1080P	316 ± 29.96	359 ± 34.37
540P - 1080P	369 ± 105.95	447 ± 95.59

the super-resolution computing power allocated to the current video chunk and the computing power used for online training of the model. The smaller the γ value is, the less important is the model training and vice-versa. Hence, the weight of the training gain γ is is recommended to be set to 0.8.

Baselines. We evaluate the performance of the proposed FlexSRVC in terms of various aspects by comparing it with three methods:

- The original method: broadcasters upload live video encoded using H.264 based on the uplink available bandwidth, and the cloud server does not perform any video enhancement process.
- LiveNAS: broadcasters send low-resolution video streams, and the cloud server super-resolves all frames of the low-resolution streams used in [7].
- NEMO-uplink: NEMO-uplink: it is a modified version of NEMO [49] to support the uplink streaming scenario in which broadcasters only apply the DNN training to a subset of selected frames and afterwards send the processed frames and models to the cloud-side. The cloud server super-resolves only selected frames based on the received models and upscales separately the remaining frames from the cache.

B. Compression Ratio Choice

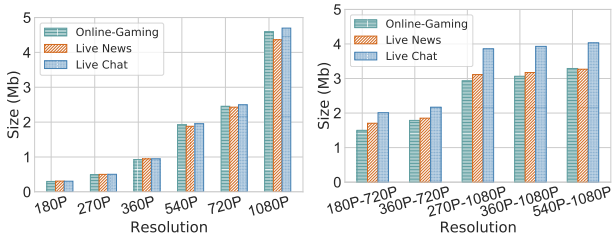
In order to identify the most appropriate compression ratio to be used in the proposed SR-based coding scheme, real uplink network bandwidth information is collected in different environments. Figure 11a indicates the original GOP size for each resolution. As shown in Figure 11a, the size of high-resolution GOPs is almost double the collected average uplink bandwidth. It is difficult for a live broadcaster to upload high-resolution GOPs under limited uplink bandwidth. Suppose a live broadcaster chooses the low-resolution GOP to upload. Even though the transmission time can be shorter, the quality of the live video content degrades and affects users' QoE. Compared with traditional image quality metrics, such as Peak Signal-to-Noise Ratio (PSNR) [50] and Structural Similarity (SSIM) [50], VMAF is the closest in terms of the subjective perception of the human eye. A VMAF score ranges from 0 to 100, with a score of 0-20 indicating an unacceptable quality, 20-40 mapping to bad quality, 40-60 to reasonable, 60-80 to decent, and 80-100 showing an outstanding quality level [39]. In addition, the quality of the corresponding different types of videos in different sizes are presented in Figure 12a in terms of their associated VMAF scores, indicating how the quality of videos is positively correlated with the video sizes.

The possible choices for compression include 180P-720P, 360P-720P, 270P-1080P, 360P-1080P and 540P-1080P. For example, the combination 180P-720P indicates that broadcaster-side downsamples several frames of an online video to 180P, and the cloud-side upsamples the received video to 720P. We set the GOP size to 24 frames in our experiment. Since there are a variety of compression frame combinations to choose from, we only show the GOP size obtained by possible compression choices in Figure 11b. As we can see from figure 11b, the bandwidth consumption for the compressed GOPs obtained by our proposed video coding module is below the average uplink bandwidth and is at least 20% smaller than that of the required bandwidth when delivering a 1080HD online stream. Figure 12b illustrates the quality of compressed GOPs using SR in terms of VMAF scores. For all the different types of videos in Figure 12b, it can be seen that the compression choices 360P-720P and 360P-1080P achieve better quality on average than 180P-720P and 270P-1080P with the similar compressed chunk size shown in Figure 11b, respectively. The compression option 540P-1080P also achieves good video quality for three types of videos. Therefore, we select the following compression options: 360P-720P, 360P-1080P and 540P-1080P when aiming at bandwidth saving while also ensuring high-quality video.

In our considered upload video system, since we can choose to compress different numbers of video frames and can downsample them by multiple compression scales, the FLBA algorithm has a significant number of bitrate options. Assuming that each resolution corresponds to a bitrate in the initial encoding, D is the number of different resolutions, each encoded GOP includes M frames and the number of selectable downsampling scales is A, there are $D \times M \times A$ optional bitrates. However, we can not include all possible bitrate choices when performing bitrate selection for each video chunk since this would be associated with a high computational cost. Thus, it is significant to refine the bitrate choices before performing the FLBA algorithm. Figure 14 shows the compression ratio of different frame compression for online gaming videos. It can be seen from Figure 14 that the more frames to be compressed are, the greater the compression rate is. When the number of frames is one, only the key frame is compressed for a GOP. We can achieve a compression rate of 30%-48% when KF and NKFs are compressed. When compressing NKFs, the increase in compression rate becomes smaller than that of compressing KFs. In order to reduce the computational cost, in the case of a relatively fixed scene, the number of optionally compressed frames is set to a fixed interval greater than one.

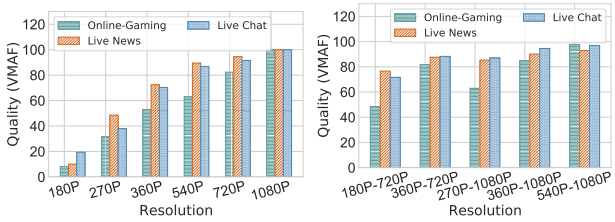
C. Performance of the Flexible Frame Video Coding Method

1) Video compression efficiency: In order to further study our flexible frame video coding scheme, we implement our video coding module on three different video types. Figure 8 shows two sample traces for our bandwidth collection process under Airport and Canteen environments. The range of the uplink bandwidth is from 0.8Mbps to 8.39Mbps. The average uplink bandwidth is 3.7Mbps. Figure 7a shows the



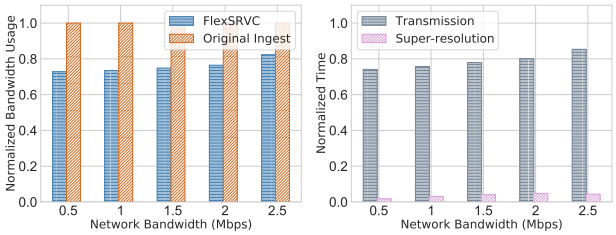
(a) Original Chunk Size (b) Compressed Chunk Size

Fig. 11: Chunk Size (i.e. GoP Size=24)



(a) Original GOP Quality (b) Compressed GOP Quality

Fig. 12: GOP Quality



(a) Bandwidth savings (b) Transmission time savings

Fig. 13: Comparison with the original method

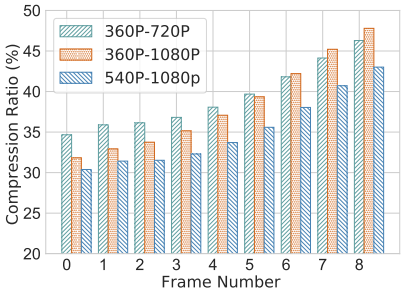


Fig. 14: Compression rate across consecutive frames

compression rate of different video types using our flexible frame video coding method to downsample 1080P to 360P. KF indicates that we only compress the key frame of a video stream. KF+4NKF and KF+8NKF indicate that we compress four non-key frames and key frame, and eight non-key frames and key frame, respectively.

As we can see in Figure 7a, the more compressed frames are, the higher the video compression is. The overall median video compression ratio is between 22% and 48%. The median video compression ratio for KF+4NKF is on average 5% higher than KF and the median video compression ratio for KF+8NKF is 12% higher than KF+4NKF. The increase in compression ratio from KF to KF+8NF is getting smaller since the size of a key frame is greater than the size of a non-key frame due to its intra-frame coding. Figure 7b shows the average compression rate of different resolution combinations over different video types. 360P-720P means we compress 720P key frame to 360P on the broadcaster side. As shown in Figure 7b, the compression combination of 360P-1080P achieves the highest compression ratio among all the compression choices since 360P-1080P uses the largest downsample scale.

2) Processing time: Compared with the traditional encoding method, the new encoding method introduces an additional processing time, including 1) time for selectively downsampling and re-encoding the video frames at the broadcaster side. 2) time to enhance the downsampled low-resolution video frames and then re-encode the reconstructed high-resolution framed, at the cloud server-side. The new encoding method is transparent to the viewer, and the viewer can decode and play high-definition video normally without modifying its decoding process.

Although our coding module adds a stage to the traditional video encoding process in live streaming, the resulting process latency is incredibly low. Table III and Table II show the average additional time consumption of the proposed flexible video coding module for a one-second video chunk which contains one GOP (GOP size = 24) at the broadcaster and smart cloud side, respectively. The column index KF+4NKF means that the key frame and first four non-key frames are compressed, and the column index KF+8NKF indicates that the key frame and first eight non-key frames are compressed. The row index 360P-720P indicates that the broadcaster compresses 720P key frames to 360P and the smart cloud super-resolved 360P key frames to the original 720P. The definition is repeated in the next two rows.

We employed a regular PC with one CPU (Intel Core i7) for video frames compression at the broadcaster side and a 2080Ti GPU for super-resolution at the smart cloud server. As we can see from Table II and Table III, the average additional processing times and one standard deviation of the mean processing time at the broadcaster and at the cloud side for all the compression combination, i.e., compressing five 720P frames to 360P (360P -720P) is 180 millisecond with 38.38 millisecond standard deviation. The higher the input resolution is, the greater the compression time is, too. 540P-1080P has the greatest processing time on both the broadcaster and the cloud sides. As the delay associated with our video coding module is extremely low, this module is appropriate for live video streaming.

D. Performance of the online SR video coding model

1) Online learning performance: Figure 15 shows the original high-resolution 1080P video on the left, and different snapshots of the marked area after enhancement from 360P to 1080P on the right. The bottom right corner of each snapshot is the VMAF score. Compared with the bicubic upsampling method, using a general model can significantly improve the video quality. Although the video enhancement quality of the SP-similar model trained on other channels is lower than that of the specific model, its performance is still better than that of the offline pre-trained model due to the similarity of the

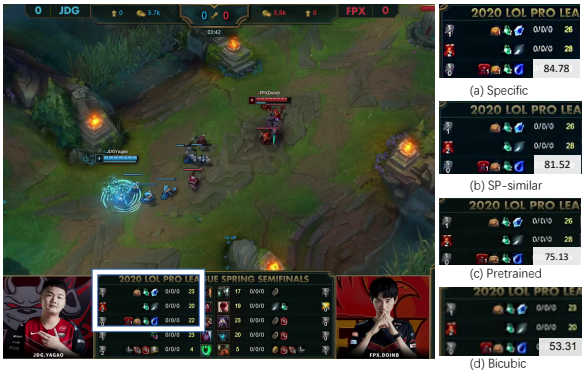


Fig. 15: 360P to 1080P super-resolution results

scene. If there is no SR-similar model, the offline pre-trained model is used.

2) SR inference quality: Figure 9 indicates the quality improvement of super-resolution using our proposed framework FlexSRVC. We use 360P and 540P as the super-resolution inputs from three video categories. The super-resolution model upgrades 360P to 720P and 1080P and reconstructs 540P to 1080P. We compress the key frame and four non-key frames (4NKF) and the key frame and eight non-key frames (8NKF) in H.264 using the fastest option in FFmpeg. It can be seen from Figure 9 how FlexSRVC improves the quality of the original low-resolution streaming by between 10% and 50% for both schemes. The overall quality improvement of 4NKF is greater than that of 8NKF.

3) SR inference efficiency: Non-key frame compression requires longer super-resolution processing time and more computing resource consumption compared to key frame compression. We observe that a single super-resolution task cannot fully utilize the GPU processing power. Therefore, we conduct several experiments using a GeForce RTX 2080 Ti GPU to discover the maximum processing capacity of a single GPU. We use a different number of threads to run a single super-resolution task through a single GPU.

Figure 10 shows the super-resolution inference time for a GOP of 24 frames by reconstructing low-resolution 360P frames to 720P and 1080P and 540P frames to 1080P. As the number of threads increases, GPU utilization increases, but excessive parallelization affects the overall system efficiency. As we can see from Figure 10, super-resolution with 2 threads achieves the shortest processing time. With more than 2 threads, although more parallelization is employed, the super-resolution processing is longer than that for 2 threads. Therefore, we adopt multi-thread super-resolution to optimize the overall GPU utilization efficiency (but keep the number of threads low) and reduce video processing latency.

E. Comparison of Video Coding Methods

This subsection evaluates our proposed flexible frame video coding scheme in comparison with BIPP [8] that compresses the non-key frames only in the video. In our experiments, we compress non-key frames by a downsampling factor of 2 corresponding to a 720P high-resolution chunk (720-360-nonkey), and by a downsampling factor of 2 and 3 corresponding to a 1080P high-resolution chunk (1080-360-nonkey and 1080-540-nonkey). Compared to non-key frame compression, key frame compression is associated with the highest cost-benefit ratio, as it can achieve a high compression ratio with minimal computational resource consumption. Therefore, we only compress the key frames of a live video to evaluate the superiority of our flexible video coding module.

TABLE IV: Comparison of compression methods

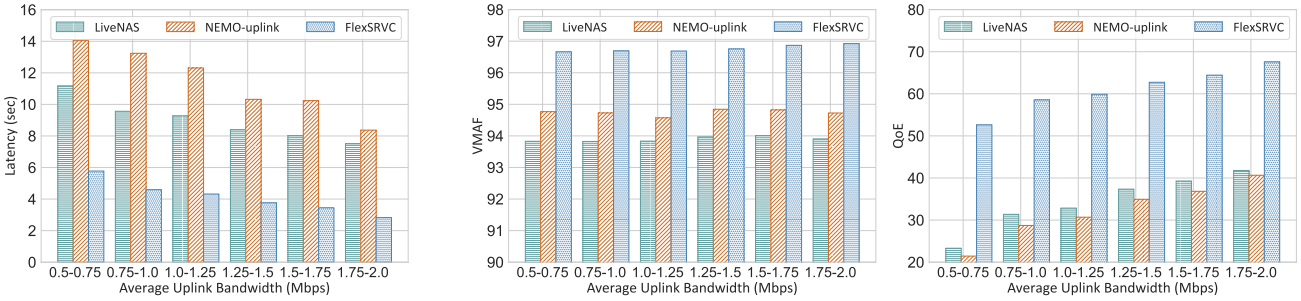
Method	Compression	Time (sec)	Quality
720-360-nonkey	29%	1.63	74.3
1080-360-nonkey	53%	2.20	80.0
1080-540-nonkey	28%	3.64	94.6
720-360-key	40%	0.03	79.1
1080-360-key	44%	0.03	80.4
1080-540-key	41%	0.19	92.8

For our video coding module, we use the same combination of the super-resolution based hybrid coding method (720-360-key, 1080-360-key, 1080-540-key) already described. We evaluate the compression rate of both coding methods and the inference time of downsampled frames using the super-resolution model and compare the quality of outputs. Table IV shows that both methods achieve similar video quality for each combination. Even though the number of non-key frames in a video chunk is greater than the number of key frames, the compression rate of our key frame coding module is on average 10% higher than that of BIPP since the size of a key frame is greater than the size of a non-key frame. The inference time of our coding module is at least 20 times less than that of BIPP, which indicates that our solution is more efficient than BIPP.

F. Comparison of Live Video Uploading Methods

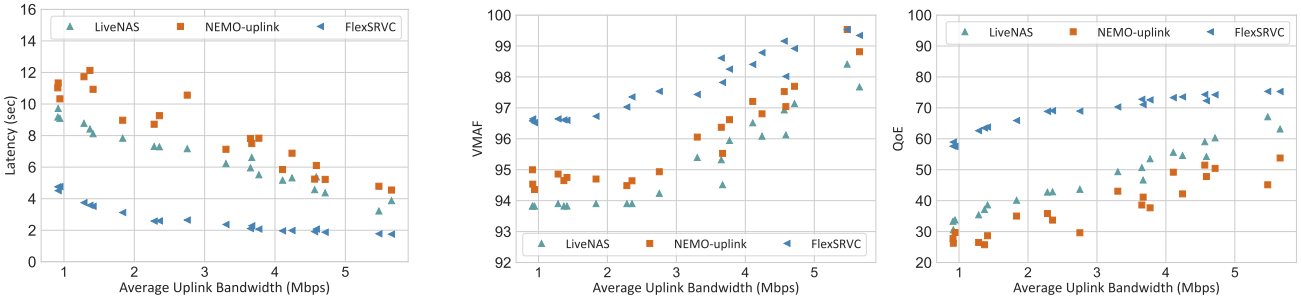
1) Comparison with the original method: Our proposed video uploading framework, FlexSRVC, can provide higher video quality than the original method in limited uplink bandwidth conditions since our model reduces the influence of the uplink bandwidth capacity on the live streaming quality. In order to verify the bandwidth consumption reduced by FlexSRVC, we evaluate the bandwidth usage of FlexSRVC normalized to the bandwidth usage for the original live streaming when delivering the same quality video with 1080P under fixed network traces. The uplink bandwidth in the network traces is {0.5, 1.0, 1.5, 2.0, 2.5}Mbps. As shown in Figure 13a, FlexSRVC achieves similar quality levels as the original method using only 74% - 85% of the required bandwidth for different network conditions. Figure 13b indicates the transmission time using FlexSRVC and the super-resolution processing time normalized to the transmission time for the original live streaming when delivering 1080P video. As it can be seen, FlexSRVC reduces the transmission time by on average 20% of the time to upload the original high-resolution video. Noteworthy is that only 2% -5% of the transmission time of the original live streaming is used for the SR inference.

2) Evaluation in the open 4G traces: We evaluate comparatively the performance of FlexSRVC, LiveNAS and NEMO-uplink with our proposed bitrate adaptation algorithm for video bitrate upgrades from 0.5Mbps to 2Mbps when a live video is encoded in H.264 using the fastest option in FFmpeg. Figure 16 shows the comparative performance of FlexSRVC,



(a) Latency (b) Quality (c) QoE

Fig. 16: Comparative performance assessment when using the open 4G traces.



(a) Latency (b) Quality (c) QoE

Fig. 17: Comparative performance assessment when using the self-measured traces.

LiveNAS and NEMO-uplink using our adaptation algorithm. Figures 16a, 16b and 16c illustrate the average latency, quality and QoE of different uplink bandwidth ranges. Figure 16a indicates the average upload latency from broadcasters to cloud servers, including the video upload time and video processing time in the cloud, for different uplink bandwidth intervals. FlexSRVC outperforms LiveNAS and NEMO-uplink since FlexSRVC compresses key frames and non-key frames flexibly according to current network conditions. Even in extremely low uplink bandwidth, FlexSRVC still achieves the lowest upload latency while ensuring the highest video quality and QoE. As the uplink bandwidth increases, the latency of FlexSRVC reduces significantly, while the latency of LiveNAS is maintained at a relatively high level. NEMO-uplink incurs a much higher delay as it has to train the DNN model at the broadcaster side and cannot adapt to the network conditions. Since both LiveNAS and NEMO-uplink need to reconstruct all low-resolution frames in the cloud, when the uplink bandwidth increases, the transmission time becomes shorter, but the video processing time by SR is very time-consuming. Figure 16b and 16c show that with the assistance of our flexible bitrate adaptation algorithm, FlexSRVC achieves higher video quality for all network traces, even under a poor uplink bandwidth environment. Finally, FlexSRVC also achieves the highest users' QoE for all network traces. Compared with LiveNAS and NEMO-uplink, the QoE obtained by FlexSRVC in different network conditions has increased by 61% on average, demonstrating that FlexSRVC can adapt very well to network situation and supports high service quality.

3) Evaluation with the self-acquired traces: To further confirm the superiority of our proposed framework, FlexSRVC, we conduct experiments with the self-acquired traces in different environments. Figure 17 presents the performance of FlexSRVC, LiveNAS and NEMO-uplink with our proposed bitrate adaptation algorithm for each network trace. As shown in Figure 17a, FlexSRVC has better performance than LiveNAS and NEMO-uplink, achieving high video quality under various uplink bandwidth conditions. Under extreme low uplink bandwidth, the upload latency for FlexSRVC is low since FlexSRVC can flexibly compress key frames and non-key frames and hence adapt better to the poor uplink bandwidth environment. As the uplink bandwidth increases, the upload latency of LiveNAS and NEMO-uplink are maintained at a relatively high level due to the SR processing of all low-resolution frames in the cloud. Compared with LiveNAS and NEMO-uplink, FlexSRVC reduces on average the upload latency by 54.8% and 64.2%, and improves users' QoE by 81.4% and 94.2%, respectively. Noteworthy is that the proposed framework, FlexSRVC outperforms LiveNAS and NEMO-uplink for all the network traces, providing better live streaming services under different network conditions.

V. Conclusions and Future Work

Video quality when employing traditional live video delivery strategies is strongly related to the available bandwidth. In this paper, we introduce FlexSRVC, a novel live video stream delivery framework based on SR, which improves the video quality for live streaming at low bitrates with short cloud processing time. Specifically, FlexSRVC integrates a flexible frame video coding scheme design for improving compression efficiency. Unlike the previous SR-based hybrid coding algorithms, which further compress the non-key frames, our coding scheme downsamples the key frames and non-key frames flexibly at the broadcaster side. This new coding method increases

compression efficiency and reduces the computation load as it processes the key frames only. FlexSRVC also introduces a novel bitrate adaptation algorithm (FLBA) to adjust content delivery dynamically to the uplink network bandwidth and cloud server computing resources. Testing shows how FlexSRVC achieves 10%-50% video quality improvement when compared to traditional live streaming approaches and obtains similar video quality while reducing with up to 25% the associated bandwidth requirements. Compared to the method of reconstructing all frames with SR during live streaming, it improves users' QoE by up to 85%, and ensures low latency during live video streaming.

Future work will consider using a more complex QoE model to address other relevant QoE influencing factors such as upscaling effects, time the highest quality is employed, and temporal pooling. Additionally, as the network bandwidth is limited, proposing an edge-assisted video delivery framework for live streaming will also be considered.

ACKNOWLEDGMENT

This work is supported by the Key-Area Research and Development Program of Guangdong Province under grant No. 2020B0101130006), National Natural Science Foundation of China under grant No.61972189 and No. 62201244, the Shenzhen Key Lab of Software Defined Networking under grant No.ZDSYS20140509172959989 and the Major Key Project of PCL under grant No. PCL2021A03-1. G.-M. Muntean acknowledges the support of the Science Foundation Ireland Research Centres Programme grant number 12/RC/2289_P2 (Insight).

REFERENCES

[1] "Cisco annual internet report (2018–2023) white paper," Mar. 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] Apple, "Hls authoring specification for apple devices," 2020. [Online]. Available: https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices

[3] Z. Wang, Y. Cui, X. Hu, X. Wang, W. T. Ooi, and Y. Li, "Multilive: Adaptive bitrate control for low-delay multi-party interactive live streaming," in *IEEE Conference on Computer Communications, INFOCOM, Toronto, ON, Canada, July 6-9*. IEEE, 2020, pp. 1093–1102.

[4] R.-X. Zhang, T. Huang, M. Ma, H. Pang, X. Yao, C. Wu, and L. Sun, "Enhancing the crowdsourced live streaming: a deep reinforcement learning approach," in *The 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV 2019, Amherst, MA, USA, June 21, 2019*. ACM, 2019, pp. 55–60.

[5] P. Dogga, S. Chakraborty, S. Mitra, and R. Netravali, "Edge-based transcoding for adaptive live video streaming," in *USENIX Workshop on Hot Topics in Edge Computing, HotEdge, Renton, USA, July 9*, 2019.

[6] Z. Xu, X. Zhang, and Z. Guo, "Qoe-driven adaptive k-push for http/2 live streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1781–1794, 2018.

[7] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han, "Neural-enhanced live streaming: Improving live video ingest via online learning," in *ACM SIGCOMM, Virtual, USA, Aug. 10-14*. ACM, 2020, pp. 107–125.

[8] H. Lin, X. He, L. Qing, Q. Teng, and S. Yang, "Improved low-bitrate hevc video coding using deep learning based super-resolution and adaptive block patching," *IEEE Transactions on Multimedia*, vol. 21, no. 12, pp. 3010–3023, 2019.

[9] "Video latency in live streaming," 2020. [Online]. Available: https://aws.amazon.com/media/tech/video-latency-in-live-streaming/?nc1=h_ls

[10] G.-M. Muntean, "Efficient delivery of multimedia streams over broadband networks using qoas," *IEEE Transactions on Broadcasting*, vol. 52, no. 2, pp. 230–235, 2006.

[11] J. Adams and G.-M. Muntean, "Adaptive-buffer power save mechanism for mobile multimedia streaming," in *IEEE International Conference on Communications*, 2007, pp. 4548–4553.

[12] G.-M. Muntean, P. Perry, and L. Murphy, "A comparison-based study of quality-oriented video on demand," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 92–102, 2007.

[13] G.-M. Muntean and N. Cranley, "Resource Efficient Quality-Oriented Wireless Broadcasting of Adaptive Multimedia Content," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 362–368, Mar. 2007.

[14] A. Yaqoob, T. Bi, and G.-M. Muntean, "A DASH-based Efficient Throughput and Buffer Occupancy-based Adaptation Algorithm for Smooth Multimedia Streaming," in *International Wireless Communications Mobile Computing Conference (IWCMC)*, Jun. 2019, pp. 643–649.

[15] L. Zou, T. Bi, and G.-M. Muntean, "A DASH-Based Adaptive Multiple Sensorial Content Delivery Solution for Improved User Quality of Experience," *IEEE Access*, vol. 7, pp. 89 172–89 187, 2019.

[16] Y. Geng, X. Zhang, T. Niu, C. Zhou, and Z. Guo, "Delay-constrained rate control for real-time video streaming over wireless networks," in *2015 Visual Communications and Image Processing, VCIP 2015, Singapore, December 13-16, 2015*. IEEE, 2015, pp. 1–4.

[17] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (webrtc)," in *International Conference on Multimedia Systems, MMSys, Klagenfurt, Austria, May 10-13*. ACM, 2016, pp. 13:1–13:12.

[18] E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu, and J. Lyu, "Real-time bandwidth prediction and rate adaptation for video calls over cellular networks," in *The 7th International Conference on Multimedia Systems, MMSys Klagenfurt, Austria, May 10-13*. ACM, 2016, pp. 12:1–12:11.

[19] G. Bakar, R. A. Kirmizioglu, and A. M. Tekalp, "Motion-based rate adaptation in webrtc videoconferencing using scalable video coding," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 429–441, 2019.

[20] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[21] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 1646–1654.

[22] J. Caballero, C. Ledig, A. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi, "Real-time video super-resolution with spatio-temporal networks and motion compensation," in *IEEE Conf. on Computer Vision and Pattern Recognition, Honolulu, USA, July 21-26*, 2017, pp. 2848–2857.

[23] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[24] H. Hou and H. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 26, no. 6, pp. 508–517, 1978.

[25] C. E. Duchon, "Lanczos filtering in one and two dimensions," *Journal of applied meteorology*, vol. 18, no. 8, pp. 1016–1022, 1979.

[26] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Conf. on Computer Vision & Pattern Recognition*, 2017, pp. 4681–4690.

[27] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *European Conf. on Computer Vision Workshops (ECCVW)*, Sep. 2018.

[28] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. C. Loy, "Edvr: Video restoration with enhanced deformable convolutional networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[29] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Las Vegas, NV, USA, June 27-30*. IEEE, 2016, pp. 1874–1883.

[30] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops, Honolulu, HI, USA, July 21-26*. IEEE, 2017, pp. 1132–1140.

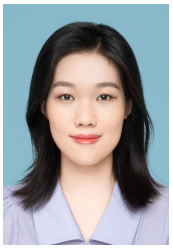
[31] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Computer Vision - ECCV 15th European Conference, Munich, Germany, September 8-14*. Springer, 2018, pp. 256–272.

[32] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 65–76, apr 2009.

[33] Y. Li, D. Liu, H. Li, L. Li, F. Wu, H. Zhang, and H. Yang, "Convolutional neural network-based block up-sampling for intra frame coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 9, pp. 2316–2330, 2017.
[34] J. Glaister, C. Chan, M. Frankovich, A. Tang, and A. Wong, "Hybrid video compression using selective keyframe identification and patch-based super-resolution," in *IEEE Int. Symposium on Multimedia, ISM, Dana Point, CA, USA, Dec. 5-7*. IEEE, 2011, pp. 105–110.
[35] M. Shen, P. Xue, and C. Wang, "Down-sampling based video coding using super-resolution technique," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 6, pp. 755–765, 2011.
[36] K. Liu, D. Liu, H. Li, and F. Wu, "Convolutional neural network-based residue super-resolution for video coding," in *2018 IEEE Visual Communications and Image Processing (VCIP)*, 2018, pp. 1–4.
[37] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *13th USENIX Symposium on Operating Systems Design and Implementation(OSDI), Carlsbad, CA, USA, October 8-10*. USENIX Association, 2018, pp. 645–661.
[38] Y. Z. Yinjie Zhang, Y. T. Yi Wu, P. Z. Kaigui Bian, and H. T. Lingyang Song, "Improving quality of experience by adaptive video streaming with super-resolution," in *39th IEEE Conference on Computer Communications, INFOCOM, Toronto, ON, Canada, July 6-9*. IEEE, 2020, pp. 1957–1966.
[39] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," *The Netflix Tech Blog*, vol. 6, p. 2, 2016.
[40] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Comyco: Quality-aware adaptive video streaming via imitation learning," in *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*. ACM, 2019, pp. 429–437.
[41] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *ACM SIGCOMM, London, UK, August 17-21*. ACM, 2015, pp. 325–338.
[42] "H.264 video encoding guide," 2020. [Online]. Available: https://trac.ffmpeg.org/wiki/Encode/H.264/
[43] "Nginx," (2020-03-30). [Online]. Available: http://nginx.org/
[44] "uwsgi documents," https://uwsgi-docs.readthedocs.io/en/latest/tutorials/, (2020-11-01). [Online]. Available: https://uwsgi-docs.readthedocs.io/en/latest/tutorials/
[45] "Django," (2020-03-30). [Online]. Available: https://www.djangoproject.com/
[46] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops, Honolulu, HI, USA, July 21-26*. IEEE, 2017, pp. 1122–1131.
[47] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4g lte dataset with channel and context metrics," in *The 9th International Conference on Multimedia Systems (MMSys), Amsterdam, The Netherlands, June 12-15, 2018*. ACM, pp. 460–465.
[48] B. Hubert *et al.*, "Linux advanced routing & traffic control howto," (2002-07-22). [Online]. Available: https://tldp.org/HOWTO/Adv-Routing-HOWTO/
[49] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, *NEMO: Enabling Neural-Enhanced Video Streaming on Commodity Mobile Devices*. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3372224.3419185
[50] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of psnr in image/video quality assessment," *Electronics letters*, vol. 44, no. 13, pp. 800–801, 2008.

Qing Li (S'10-M'14) received the B.S. degree (2008) from Dalian University of Technology, Dalian, China, the Ph.D. degree (2013) from Tsinghua University, Beijing, China. He is currently an Associate Researcher with with Peng Cheng Laboratory, Shenzhen, China. His research interests include reliable and scalable routing of the Internet, in-network caching/computing, intelligent self-running network, edge computing, etc.

Ying Chen received the B.Eng. (Hons) degree in Engineering Science from The University of Auckland in 2016. She is currently a Master student with Tsinghua-Berkeley Shenzhen Institute. She majors in Data Science and Information Technology. Her research interests include adaptive multimedia and multimedia streaming, resource allocation and user quality of experience.



Aoyang Zhang received the B.Eng. degree in Communication Engineering from Beijing Institute of Technology in 2018. She is currently a Master student with Tsinghua-Berkeley Shenzhen Institute. She majors in Data Science and Information Technology. Her research interests include adaptive multimedia and multimedia streaming, resource allocation and user quality of experience.

Yong Jiang is currently a Full Professor with Tsinghua Shenzhen International Graduate School. He received his B.S. degree and Ph.D. degree both from Tsinghua University, respectively in 1998 and 2002. He mainly focuses on the future Internet, edge computing, multimedia transmission, AI for networks, etc.

Longhao Zou (S'12-M'19) received the B.Eng and Ph.D degrees from Beijing University of Posts and Telecommunications (BUPT), Beijing, China and Dublin City University (DCU), Ireland in 2011 and 2016, respectively. He was a postdoctoral researcher with the EU Horizon 2020 NEWTON Project at DCU. Now he is an Associate Researcher with Peng Cheng Laboratory, Shenzhen, China. His research interests include mobile and wireless communications, adaptive mulsemedia and multimedia streaming, resource allocation and user quality of experience.

Zhimin Xu is currently an algorithm engineer in Multimedia Lab in Beijing Bytedance Technology Co., Ltd., Beijing, China. He received the B.S. degree in network engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016, the M.S. degree in Wangxuan Institute of Computer Technology, Peking University, Beijing, China, in 2019. His research interests include video streaming, multimedia communication, QoE, 360-degree video, playback experience optimization. He is the Runner-up of the IEEE International Conference on Multimedia and Expo (ICME) DASH-IF Grand Challenge Award on Dynamic Adaptive Streaming over HTTP (DASH) both in 2017 and 2018.

Gabriel-Miro Muntean (M'04, SM'17) is a Professor with the School of Electronic Engineering, Dublin City University (DCU), Ireland, and Co-Director of the DCU Performance Engineering Laboratory. Prof. Muntean was awarded the PhD degree by DCU for research on adaptive multimedia delivery in 2004. He has published over 400 books, chapters and papers in top-level international venues. His research interests include quality, performance, and energy saving issues related to rich media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. Prof. Muntean is an Associate Editor of the IEEE Transactions on Broadcasting, the Multimedia Communications Area Editor of the IEEE Communications Surveys and Tutorials, and chair and reviewer for top international journals and conferences.