# BiSR: Bidirectionally Optimized Super-Resolution for Mobile Video Streaming


Qian Yu
12133094@mail.sustech.edu.cn
SUSTech
Peng Cheng Laboratory
Shenzhen, China

Qing Li*
liq@pcl.ac.cn
Peng Cheng Laboratory
Shenzhen, China

Rui He
12032200@mail.sustech.edu.cn
SUSTech
Peng Cheng Laboratory
Shenzhen, China

Gareth Tyson
Hong Kong University of Science and Technology(GZ)
Guangzhou, China

Wanxin Shi
International Graduate School, Tsinghua University
Shenzhen, China

Jianhui Lv
Peng Cheng Laboratory
Shenzhen, China

Zhenhui Yuan
Northumbria University
Newcastle, United Kingdom

Peng Zhang
Yulong Lan
Zhicheng Li
Tencent
Shenzhen, China



## ABSTRACT

The user experience of mobile web video streaming is often impacted by insufficient and dynamic network bandwidth. In this paper, we design **Bi**directionally Optimized **S**uper-**R**esolution (BiSR) to improve the quality of experience (QoE) for mobile web users under limited bandwidth. BiSR exploits a deep neural network (DNN)-based model to super-resolve key frames efficiently without changing the inter-frame spatial-temporal information. We then propose a downscaling DNN and a mobile-specific optimized lightweight super-resolution DNN to enhance the performance. Finally, a novel reinforcement learning-based adaptive bitrate (ABR) algorithm is proposed to verify the performance of BiSR on real network traces. Our evaluation, using a full system implementation, shows that BiSR saves 26% of bitrate compared to the traditional H.264 codec and improves the SSIM of video by 3.7% compared to the prior state-of-the-art. Overall, BiSR enhances the user-perceived quality of experience by up to 30.6%.



## CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Networks** → *Mobile networks.*

## KEYWORDS

Mobile video streaming, video codec, super-resolution

**ACM Reference Format:**
Qian Yu, Qing Li, Rui He, Gareth Tyson, Wanxin Shi, Jianhui Lv, Zhenhui Yuan, Peng Zhang, Yulong Lan, and Zhicheng Li. 2023. BiSR: Bidirectionally Optimized Super-Resolution for Mobile Video Streaming. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583519



*Corresponding author: Qing Li




*WWW '23, April 30–May 04, 2023, Austin, TX, USA*



ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583519


## 1 INTRODUCTION

A recent survey revealed that there are over 4.6 billion active internet users in 2021, 92.6% of whom are mobile [21]. Meanwhile, according to the latest report from Ericsson [10], monthly video streaming takes up 66.8% (i.e., 5.99 GB) of the total data consumption of an individual mobile user. However, many mobile web users still suffer from unsatisfactory video streaming due to insufficient network bandwidth [31]. To improve the quality of experience (QoE), content delivery networks (CDNs) and mobile carriers have striven to increase their footprint and available network bandwidth [6, 11]. Furthermore, optimized adaptive bitrate algorithms [25, 26, 34, 48] have been developed to improve the QoE across a diverse range of users. However, the majority of existing works focus on optimizing network use without considering mobile devices' computing resources.

Recently, many works [4, 16, 22, 45–47, 49] have attempted to improve video streaming quality using client-side computation. Our work takes inspiration from NAS [46]. This technique uses deep neural networks (DNNs) to enhance video streaming quality. In NAS, clients upscale the quality of every frame during video streaming based on a pre-trained DNN. This reduces data transmission requirements, but requires a large amount of computing resources on the client, thereby making it unsuitable for mobile devices. To reduce the required resources, a recent extension, NEMO [45], tries to upscale only some of the frames using a pre-trained super-resolution (SR) DNN. This reduces the computational load on the client, yet at the cost of video quality.

This degradation occurs because NEMO's pre-trained SR DNN (deployed on mobile devices) must be sufficiently lightweight to allow real-time rendering of frames. Thus, the image quality of the SR video frames is reduced [45]. Moreover, the subsequent quality of the remaining upscaled frames is lower due to the decoding inaccuracies introduced during the upscaling process. Hence, in practice, NEMO brings limited quality improvement. In fact, our experiments show that the video quality of using NEMO to super-resolve low-resolution video is worse than directly using a video codec to encode high-resolution video (see Section 3.3 and 5.4).

To overcome the limitations of NEMO, we propose a new mobile web video streaming delivery system: BiSR. The key contribution of BiSR is a novel video codec integrated with a SR DNN that enhances the quality of mobile web video streaming. It does this by intelligently super-resolving only key frames (i.e., I frames or the first frame of a video chunk) and reallocating bitrate to other frames (i.e., non-key frames), such that they are encoded at a higher resolution. This avoids upscaling non-key frames on the client side. Specifically, on the server-side encoder, the chosen high-resolution (HR) key frame within a video chunk is first downscaled to a low-resolution (LR) by using a pre-trained DNN. Then the LR key frame is not only encoded for streaming but also super-resolved for the reference of the remaining HR non-key frames (i.e., P and B frames). On the mobile-side decoder, the received LR key frame is restored to the original high resolution using the pre-trained SR DNN. The other received HR non-key frames are then decoded referring to the former SR key frame.

This approach raises two interesting **constraints**. *First*, due to the limited resources on mobile devices, it is vital that the SR DNN model is lightweight. This, however, is difficult as it is simultaneously important to achieve good SR performance. For example, if the quality of the SR key frame is low, the inter-frame temporal redundancy between the key frame and the remaining ones will downgrade the coding efficiency for the final stream. This will substantially degrade the viewers' quality of experience.

*Second*, due to the diversity of mobile hardware and video content, it is vital that BiSR can adapt to both different video types and mobile device types. This is motivated by our observations that the SR DNN can experience varying performance when executed on different videos and device types (Section 5.4). For example, entry-level mobile devices with low computing resources cannot use complex DNNs to super-resolve video frames in real time. Yet, lightweight DNNs cannot achieve good SR performance on some videos. Thus, BiSR must be able to adaptively select the optimal configuration across a range of different mobile devices and video types for high-quality real-time video streaming.

To address the above two needs, BiSR introduces two important **design innovations**. *First*, BiSR designs a mobile-specific optimized *SR DNN* and content-aware *downscaling DNN*. The *SR DNN* is optimized for the hardware characteristics of mobile devices to speed up the SR process. Meanwhile, a *downscaling DNN* jointly trained with the *SR DNN* based on an auto-encoder structure is used to downscale key frames, to enhance the quality of SR key frames. Following this, BiSR trains overfitting DNN models for each *individual* video (similar to NAS and NEMO). However, to reflect diverse conditions, we also propose a novel video analyzer module that determines whether to use an *existing* trained model or to train a *new* video-specific (i.e., overfitting) model for new video based on the current computing resources. By reusing existing models for similar videos, we substantially decrease the computational load on the server. The *second* design innovation is a novel adaptive video streaming algorithm. Traditional adaptive bit rate (ABR) algorithms only select between video bitrates. However, BiSR introduces further per-stream decisions that must be taken into account. Specifically, our algorithm also adapts the choice of pre-trained SR DNN models to reflect the mobile device's computing resources, network bandwidth, and SR performance.

We implement the prototype of BiSR on multiple commodity mobile devices and then conduct comprehensive experiments on real network traces. The results show that BiSR saves 26% of video coding bitrate compared to the traditional H.264 codec [17], achieves a 3.7% SSIM of video improvement compared to NEMO, and brings a 17–30.6% quality of experience improvement to mobile web users.

The main contributions of our proposed BiSR are:

- We propose a novel video codec framework. The encoder downscales the key frame using a SR-oriented DNN, and super-resolves this downscaled key frame by a SR DNN for the reference of the other non-key frames' encoding. On the mobile-side decoder, the downscaled key frame is restored to a high resolution using the SR DNN, while the non-key ones are decoded based on the SR key frame.
- We propose a DNN downscaling strategy to retain better spatial information for the SR processing. The performance of the SR DNN is improved without increasing the computational complexity. Furthermore, our mobile-optimized SR DNN reduces the SR processing time by up to 55.7%.
- We propose a reinforcement learning-based adaptive video streaming scheme to intelligently select size-suitable SR DNN models and video chunks for mobile clients according to the mobile computational capacity, dynamic network bandwidth, and SR performance levels.

## 2 BACKGROUND & RELATED WORK

***Adaptive streaming:*** Adaptive streaming divides the video stream into different chunks across time and encodes each chunk with multiple bitrates. An adaptive bitrate algorithm (ABR) determines the streaming bitrates of the video chunks based on the client's historic playback information and bandwidth predictions (e.g., DASH [36]). Due to variable network bandwidth, achieving the optimal video chunk bitrate is difficult. MPC [48] uses an optimization strategy to predict the QoE of several video chunks in the future and then calculates the best bitrate for the current chunk. Of particular relevance to our work is Pensieve [26]. This proposes a deep reinforcement learning-based ABR algorithm, which makes the next video chunk bitrate decision by observing the performance of the past decisions. Despite these efforts, current ABR algorithms only rely on available network bandwidth, ignoring the video quality improvement based on clients' computing resources. Our work extends that of Pensieve to consider this important factor.

***Super-resolution:*** Super-resolution is a computer vision technique that recovers high-resolution images from low-resolution images. SRCNN [8] is a seminal work, which uses a DNN to super-resolve low-resolution images. Since, many further DNN-based SR methods have brought improved performance, e.g., VDSR [20]. Recently, works such as RFDN [24] and FALSR [5] have proposed efficient SR DNNs using information distillation and neural architecture search, which balance the computational cost and the SR performance. However, due to the limited computing resources of mobile devices, DNN-based SR still is a challenge for real-time video. Because of this, BiSR only uses SR DNN for key frames.

***Super-resolution enhanced video delivery:*** NAS [46] is the first work to integrate a SR DNN into an on-demand video delivery

system. In NAS, client locally upscales every single frame using a SR DNN. Thus, the client must have powerful computational resources. In live streaming, broadcasters in LiveSRVC [4] encode low-resolution key frames, and the server uses a powerful SR DNN (which requires a lot of computing resources) to upscale the low-resolution key frames. To reflect mobile on-demand video streaming requirements (considering the limited computing resources of mobile devices), NEMO [45] only super-resolves a subset of video frames. Then, NEMO jointly uses the super-resolved video frames and encoded streaming information of the low-resolution video to upscale the remaining frames. This reduces the required computing resources. However, the video quality of NEMO is poor due to decoding and upscaling inaccuracies introduced. According to our experiments, the video quality of using NEMO to super-resolve low-resolution video is worse than directly using a video codec to encode high-resolution video (see Section 3.3 and 5.4).

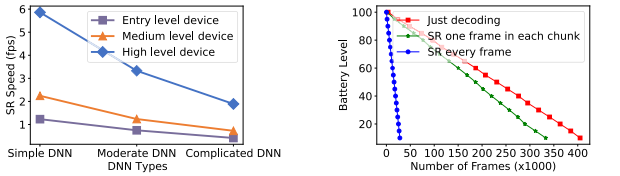
***Neural network downscaling:*** Downscaling images using a DNN has been applied to improve the performance of SR DNNs. TAD&TAU [19] is the first work that uses a DNN to downscale images to improve the performance of the SR process. CAR [37] uses a content-aware resampling kernel to downscale the image. IRN [44] applies an invertible DNN to downscale and super-resolve images, which significantly improves the performance of the SR DNN. However, the above works only strive to downscale and super-resolve images, and do not consider video delivery constraints such as limited mobile bandwidth and computational resources. To the best of our knowledge, we are the first to apply DNN downscaling to improve a practical video delivery system and verify its feasibility in a mobile video transmission application.

# 3 MOTIVATION

## 3.1 Impact of computing resources

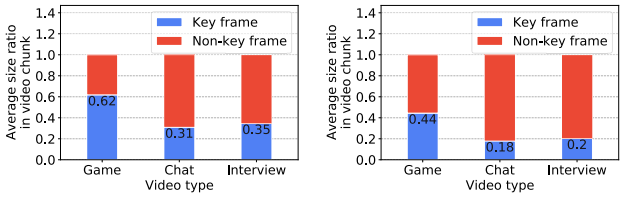
To ensure real-time playback of SR video stream, the client needs to super-resolve 24-30 frames within one second [45]. In NAS, clients with a powerful GPU could complete the above tasks. Figure 1(a) presents the speed of the SR DNN with three different complexities on three mobile devices (with different computing resources). The specific mobile phone configurations and SR DNN structures are introduced in Table 2 and Table 3, respectively.

In Figure 1(a), we see that the high-performance device can only super-resolve 6 frames per second, even with the lowest complexity DNN. Figure 1(b) also shows the battery power consumed by the processing of each frame on the high-performance device with the high-complexity SR DNN. We see that even for the high-performance device (with a large battery capacity), the battery is drained quickly when every frame is super-resolved (blue line). Compared to decoding the video frames using H.264 codec, super-resolving every single frame in the video consumes 14x the power, which dramatically reduces battery life. Therefore, super-resolving every frame in the video is not suitable for mobile devices. That said, we find that any commodity mobile device can super-resolve one frame within two seconds (the length of a video chunk with 48 frames). Importantly, Figure 1(b) confirms that super-resolving just one (key) frame in a video chunk would improve the battery life by 12x compared to super-resolving all frames.




(a) SR speed (fps) (b) Power consumption across frames

**Figure 1: The SR results on mobile devices**




(a) 1s video chunk (b) 2s video chunk

**Figure 2: Key frames size vs non-key frames size**

This shows that *super-resolving video stream is unsuitable for mobile devices, but super-resolving one frame in a chunk is feasible.*

## 3.2 Impact of the codec

Traditional video codecs compress video using the spatial and temporal dependency of frames (based on intra- and inter-frame prediction). Intra-frame compression tends to have lower compression rates than inter-frame compression. However, the first frame of a video chunk (i.e., the key frame or I frame) has no reference frame. Thus, all pixel blocks in the key frame must use intra-frame compression, leading to a lower compression ratio of key frames.

To demonstrate this, Figure 2 shows the size ratio of key frames vs. non-key frames on different videos. The results are shown for videos encoded using 1s and 2s video chunks at 24 frames per second (fps) and 1080p resolution using the H.264 codec [40]. We use six videos with three different types: Game [32, 39], Interview [1, 2], and Chat [9, 30]. We see that the compression efficiency of inter-frame compression is 10.2–37.6 times that of intra-frame compression. Although the key frames only take up 4.2% and 2.1% of the playback time in the videos of one or two seconds (i.e., one key frame per chunk), they constitute 31%–62% and 18%–44% of the video size, respectively.

Therefore, *downscaling key frames can substantially improve the compression ratio of the video. Meanwhile, it is cost-efficient that mobile devices with scarce computing resources can support super-resolving these downscaled key frames.*

## 3.3 Limitations of NEMO

NEMO [45] is a SR video streaming system for mobile devices. In NEMO, some frames in the video are selected to apply super-resolution DNN for maximizing the quality gains, and other frames are upscaled with reference to these SR frames. Here, upscaling is based on traditional interpolation methods (e.g., bilinear) and predicted motion vectors in the encoded stream. When the content of the video changes slightly, the upscaling process will naturally introduce more errors. This is because the motion vector between

frames and the residual of the inter-encoded block will become larger. Therefore, the predicted motion vector and the upscaled residual block would become more inaccurate. This causes the upscaled frames to have lower quality.

With super-resolving one frame on low-resolution video, even using a high-complexity SR DNN, after other frames of the video that is slightly different from the SR one, the PSNR of NEMO is lower than using a traditional video codec [42] to encode the high-resolution video with the same bitrate, detailed in Appendix A.

We argue that this creates an opportunity. Specifically, we observe that, except for the key frame, the video stream has a very high compression rate. This means the non-key frames can be encoded at a higher resolution without introducing excessive encoding overhead and avoiding the inaccurate upscaling process. Therefore, we conclude that *non-key frames which are encoded with reference to the SR key frames should adopt high-resolution encoding to ensure high video quality.*

## 4 BISR: SYSTEM DESIGN

Based on the above three observations, BiSR only downscales and super-resolves the key frames to reduce the transmission load with limited computing resources. BiSR then reallocates the bitrate to non-key frames, encoding them at a higher resolution to ensure video quality. BiSR aims to improve the QoE of mobile users under limited and dynamic network bandwidth.

### 4.1 Architectural Overview

Figure 3 introduces the system architecture, which includes three main parts: the server side, CDN side, and mobile side.

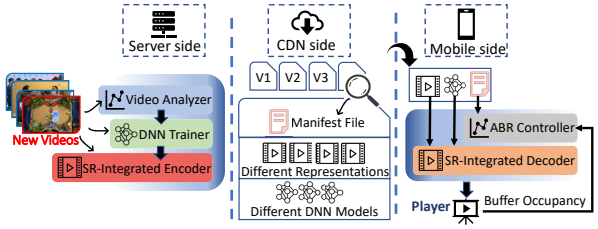



**Figure 3: Overview of the BiSR Framework**

***Server side*** : The server encodes videos into multiple representations with (four) different bitrates (400, 800, 1200, and 2400kbps) using the **SR-Integrated Encoder** module (see Section 4.2). These videos also are encoded at multiple (five) bitrates (400, 800, 1200, 2400, and 4800kbps) using the traditional H.264 video codec. When using the **SR-Integrated Encoder**, BiSR downscales the key frames (which have a larger size than the non-key frames). The remaining non-key frames are encoded at a high resolution.

To improve the performance of the **SR-Integrated Encoder** and adapt to mobile devices with differing computational capacities, BiSR uses a **DNN Trainer** module (see Section 4.3.1) to train specific overfitting DNN models with multiple complexities (sizes) for each video. The **DNN Trainer** adopts a DNN approach to downscale the key frames (which is superior to traditional techniques, e.g., bicubic and bilinear). Note, **DNN Trainer** also proposes mobile-specific SR DNN to speed up the SR process on mobile devices.

Training an overfitting model for each individual video creates notable computational overheads. To adapt to the variable resources on the server side, the **Video Analyzer** module (see Section 4.3.2) determines whether to reuse an existing trained model for new videos (or to train a new one). The former can be done if the new video is similar to an already trained one. In this way, we avoid the cost of training new specific models for each video.

Finally, the server generates multiple video representations (e.g., bitrates) and multiple-complexity (size) video-specific DNN models. It also creates a manifest file with the size information of all video chunks. These files, including videos, DNN models, and manifest files, will be delivered to Content Delivery Network (CDN) nodes.

***CDN side:*** The CDN receives these files and caches them. When the mobile client issues a video request, the CDN first delivers the video configuration file (manifest file). It then serves any subsequent requests by clients for video chunks and DNN models.

***Mobile side*** : To watch a video, mobile users download an appropriate SR DNN model based on the computational resources available. Meanwhile, mobile users request video chunks using the size information within the manifest file. The mobile users then decode and play the video chunks.

Recall, that the server trains multiple DNN models of varying complexity for each video. Because the computing power of each mobile user is different, the mobile user must therefore select the most suitable DNN model. Specifically, after installing our video player, the client tests the runtime of each model using random model parameters offline according to the structural configurations of different models. These configurations (see Table 3) have been integrated into our video player beforehand. This helps mobile users choose a model of the appropriate complexity.

When mobile users request videos, the user first downloads video chunks encoded using the H.264 video encoder. Once the video player has buffered sufficient chunks, the **ABR Controller** module (see Section 4.4) requests the appropriate DNN model (i.e., model parameters). The DNN model needs to have the highest feasible complexity, yet still be able to super-resolve one frame within one video chunk duration (two seconds). Note, this decision is done according to the prior offline testing runtime of models on our video player. Once the DNN model is fully downloaded, the mobile user can download and decode video chunks encoded by the SR-integrated encoder. Meanwhile, the buffer occupancy in the video player will be used as feedback to help the **ABR Controller** adaptively choose the video chunk encoding rate.

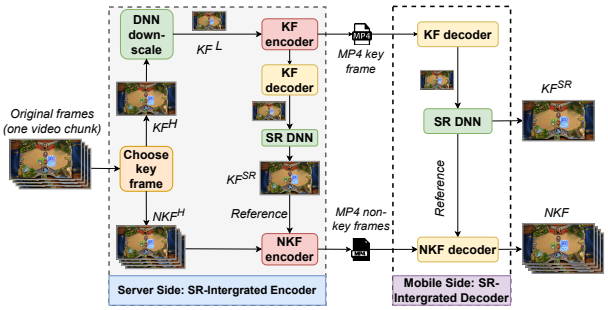



**Figure 4: The Pipeline of SR-integrated Codec**

## 4.2 SR-integrated Encoder and Decoder

Figure 4 presents the pipeline of our SR-integrated encoder and decoder. In Figure 4, The encoder and decoder of key frames and non-key frames are the traditional H.264 encoder and decoder respectively, and the SR DNN is used to up-scale key frames. Take a single video chunk as an example. On the video encoder side, the encoder first extracts the key frame ($KF^H$) from the video chunk. The key frame is then downscaled by the DNN to get the low-resolution version ($KF^L$). It is encoded to an MP4 file to reduce the video encoding rate. Non-key frames ($NKF^H$) will be encoded in a separate MP4 file with reference to the key frame ($KF^{SR}$) that is decoded first and then upscaled using SR DNN. The reason for encoding key and non-key frames in two separate MP4 files is to reduce the video processing time (detailed in Section 4.4).

Upon decoding, the MP4 file that contains the key frame is first decoded and upscaled to a high resolution using SR DNN. Another MP4 file containing the non-key frames ($NKF$) is then decoded with reference to the SR key frame ($KF^{SR}$). Because the computing resources of mobile devices are limited, BiSR only downscales and super-resolves the key frame (as it occupies a large fraction of the video chunk size). Meanwhile, it ensures high video quality by encoding and decoding high-resolution non-key frames.

## 4.3 DNN Trainer and Video Analyzer

In this section, we first introduce the design of DNNs (i.e., DNN Trainer). We then present the video analyzer that decides which videos share the same models based on the videos' similarities.

*4.3.1 DNN Trainer.* In this part, the design details of the SR DNN and downscaling DNN are introduced. The SR DNN is optimized for mobile devices, which can speed up the SR inference speed. Furthermore, the downscaling DNN improves the SR frame quality by more efficiently downscaling video frames without increasing the computational burden on mobile devices.

For the SR DNN, BiSR uses a Residual Feature Distillation Network (RFDN) [24] as its backbone. It achieves a better SR performance with fewer parameters using information distillation. Despite this, it is too heavyweight to operate on a mobile device (because of the channel attention module in RFDN). It also does not support half-precision floating-point (i.e., 16-bit floating point) inference during the DNN inference process, yet the half-precision floating-point inference can vastly reduce DNN inference time. To overcome this, we design a novel channel attention module.

Figure 5 outlines the channel attention module in BiSR. The module reduces the size of the feature map and improves the inference speed on mobile devices by using average pooling with a kernel size of 5x5 and a stride of 3, which also supports all precision types of DNN inference. As shown in Figure 5, these reduced feature maps pass through two convolutional layers with a kernel size of 1x1. It is finally restored to its original size by an interpolation layer.

Furthermore, BiSR adopts two methods to reduce the size of the DNN models sent by the CDN to mobile users. First, inspired by EDSR[23], the SR DNN models for differently encoded bitrate videos will share the same DNN model parameters except for the upsample layer. This reduces the size of the DNN models compared to having a separate DNN model for each encoded bitrate video. Second, BiSR uses the half-precision floating point to store and

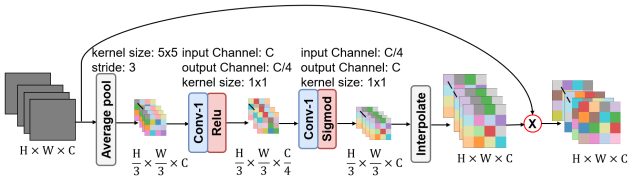



**Figure 5: Our channel attention module on mobile devices**

compress the parameters of DNN models. This is compared to using a full-precision floating point (i.e., 32-bit floating point).

Instead of traditional downscaling algorithms (such as bicubic), BiSR adopts the DNN to downscale key frames to improve the quality of SR key frames. For the downscaling DNN, we use the SR DNN without upsampling layer in BiSR as the backbone. We add a convolution layer whose kernel size is 4x4 and stride is 2 to downscale the feature maps. In addition, we also add the image of bicubic downscaling for residual learning and use the soft quantization function[13] to quantize the downscaled low-resolution (LR) output images. To reduce the streaming bitrate of LR images (i.e., key frames), unlike the most commonly used YUV420 format for a better LR visual experience, BiSR compresses these images using the YUV444 format to improve the quality of SR frames further. Appendix B details the benefits of downscaling DNN and encoding format by using an example.

*4.3.2 Video Analyzer.* The video analyzer reduces the server-side computational load for training video-specific models, by sharing the same model across multiple similar videos.

Traditionally, to improve the performance, a video-specific (e.g., overfitting) SR DNN model is trained for each video. This, however, is prohibitively expensive if a server hosts many popular videos. The video analyzer module is proposed to selectively train $K$ models for the total $N$ videos based on hierarchical clustering and video similarity when the server computing resources are insufficient. The value of $K$ is determined by the current computing resources. The idea is that similar videos may be able to use the same model (e.g., different episodes of the same news show).

BiSR leverages hierarchical clustering to divide $N$ videos into $K$ clusters. The $K$ videos that are closest to all videos of each cluster are used as training videos to train $K$ SR models. The other $N - K$ videos then use the above $K$ SR models. To calculate the distance between two videos, BiSR first extracts the key frames of each video, and clusters them according to the Phash [35] value between adjacent frames. Then, the distance (i.e., similarity) between two videos is calculated by the EMD [27–29] algorithm of their key frame clusters. Algorithm 1 in Appendix C introduces the above process of video clustering in detail.

## 4.4 ABR Controller on the Mobile Side

The ABR Controller adaptively requests video chunks and DNN models to obtain high QoE for mobile users.

In BiSR, the videos are encoded into video chunks with different bitrates. In addition, each video has its own DNN models of different sizes (i.e., complexities). This is to cope with the variable network bandwidth and mobile devices. To help mobile users get higher QoE under real network bandwidth, BiSR uses a reinforcement learning (RL) network (on the client) to select the appropriate size

**Table 1: States used in our ABR**

| Type | State |
|---|---|
| Pensieve's status | Network and video status specified in Pensieve [26] |
| BiSR's exclusive status | Past chunks avergae SR process time ($SR_t$) |
| | Next chunk key frame sizes ($I_t$) |
| | Next chunk effective sizes ($R_{\text{effective}}$) |
| | Average gain of the video quality due to SR ($g$) |
| | Leftover DNN model size ($m$) |

of video chunks and DNN models. It does this based on the available bandwidth and computing resources. BiSR builds on Pensieve's Asynchronous Advantage Actor-critic (A3C) architecture [26] by changing the input states, output actions, and RL network structures to choose suitable video chunks and DNN models.

Although NAS [46] has previously proposed adaptive bitrate algorithms for SR video stream on powerful clients, by default, it super-resolves all video chunks without considering the rebuffering caused by the SR process. This does not fit our scenario because of the limited computing power of mobile devices. This means SR video may cause additional rebuffering and trigger worse video quality compared to traditional video streaming.

Our ABR controller module works as follows. The mobile client observes input states, $s_t$, at each iteration, $t$, and makes an action $a_t$. The client then calculates the reward, $r_t$, and updates the input states to $s_{t+1}$. There are two available actions, $a_t$: to download a DNN model (at a particular complexity) or to download a video chunk (at a particulate bitrate).

BiSR considers the features of SR video delivery as the input state, $s_t$, listed in Table 1. (*i*) It takes the SR time as the input state to help calculate the rebuffering time. (*ii*) BiSR also takes the key frame size as the input state to reduce the rebuffering time due to the SR process (shown in Equation 3). (*iii*) BiSR takes the quality improvement of each SR video chunk into the input state. It does this to avoid requesting SR video chunks that have worse quality than chunks produced by traditional encoders. (*iv*) BiSR takes the average gain in video quality due to super-resolving as the input state to guide the delivery of DNN models. (*v*) It takes the remaining bytes to download for the model. (*vi*) Finally, it also includes the network and video status features from Pensieve [26].

The above process details the input state. The next thing to define is the reward function. The reward of A3C is the target QoE [26, 46]:

$$QoE = \sum_{n=1}^{N} q\left(R_n\right) - \mu \sum_{n=1}^{N} T_n - \lambda \sum_{n=1}^{N-1} \mid q\left(R_{n+1}\right) - q\left(R_n\right) \mid \quad (1)$$

where $N$ is the video chunk number; $R_n$ refers to the bitrate of video chunk $n$; $q(R_n)$ is the received quality of $R_n$ ($q(R_n) = R_n$ in our experiment); $T_n$ is the rebuffering time in the video player; $q(R_{n+1}) - q(R_n)$ represents the degree of video smoothness; $\mu$ and $\lambda$ are non-negative weighting parameters corresponding to the rebuffering time and the degree of video smoothness, respectively.

To represent the video quality enhancement based on the DNN in BiSR, we create a mapping function from quality to bitrate. Thus, we obtain the bitrate ($R_{\text{effective}}$) that corresponds to the enhanced quality in BiSR similar to NAS [46], and we use $R_{\text{effective}}$ instead of the $R_n$ in Equation 1. Further, the rebuffering time in the reward function (Equation 1), calculated as:

$$T_n = max(PT - f_t, 0) \quad (2)$$

where $f_t$, refers to the buffer length; $PT$ refers to the total video processing time (which is the time difference from when the CDN sends the video to when the video can be rendered). To calculate the above processing time ($PT$), we use the following equation:

$$\begin{aligned} KT = \frac{I_t}{b_t}, NKT = \frac{s_t - I_t}{b_t} \\ PT1 = KT + NKT \\ PT2 = KT + NKT + SR_t \\ PT3 = max(KT + SR_t, NKT) \end{aligned} \quad (3)$$

where $s_t$, $I_t$, $b_t$, and $SR_t$, respectively, are the size of the current video chunk, the size of the key frame in the video chunk, the network bandwidth, and the time of SR processing. $KT$, $NKT$ are the network transmission delays of the key frame and non-key frames, respectively. Note, when a mobile user requests video chunks that are produced using a traditional codec, the video processing time ($PT1$) equals the video transmission delay. In contrast, when a mobile user requests SR video chunks, they can only be rendered after their key frame is super-resolved. Therefore, the video processing time ($PT2$) is the sum of the video transmission delay and the time of the SR process.

However, as aforementioned, in BiSR, key frame and non-key frames are divided into two files for transmission. After the key frame transmission is completed, super-resolving the key frame and transmission of non-key frames are carried out simultaneously. Therefore, the video processing time shifts from $PT2$ to $PT3$ in Equations 3. Combining the above, BiSR iteratively strives to optimize the reward function (Equation 1) to select the optimal video chunks and DNN models.

## 5 EVALUATION

Our evaluation seeks to answer the following questions:

(1) Does BiSR improve (i.e., decrease) the video coding bitrate?
(2) Does BiSR improve QoE in dynamic network bandwidth?
(3) What is the performance contribution of each module in BiSR?

### 5.1 Datasets and Metrics

***Dataset:*** We use three types of 4K videos from YouTube [14] to evaluate BiSR: Interview, Game, and Chat. For each category, we select two popular videos with frame rates of more than 24 fps and lasting at least 5 minutes long. Since most commodity mobile devices come with a display resolution of 1080p, a video with a smaller resolution will be upscaled to 1080p by a traditional upscaling algorithm (e.g., bicubic) during the rendering process. This will cause a loss of video quality. Therefore, we transcode raw videos to different bitrates at 1080p resolution. Referring to [43]'s setup for video streaming, we transcode videos to five bitrates {400, 800, 1200, 2400, 4800}kbps at 1080p resolution using the traditional H.264 video codec. Furthermore, we also transcode each video into (*i*) videos with 270p key frames and 1080p non-key frames at two bitrates of 400, 800kbps, and (*ii*) videos with 540p key frames and 1080p non-key frames at two bitrates of 1200, 2400kbps using the SR-integrated encoder. The video chunks' length and frame rate are 2 s and 24 fps, respectively. We select video clips of 5 minutes in

length for each video. Raw 1080p videos are used as the reference for computing PSNR and structural similarity (SSIM) [41].

***QoE metrics:*** BiSR uses the QoE metric defined in Equation 1. Note, this is the same as the reward function of our ABR Controller. In this equation, $\mu$ and $\lambda$ are defined as 2.3 and 0.5 to encourage downloading high bitrate video chunks.

***Network traces:*** To train and evaluate BiSR for real-time video streaming, BiSR uses network throughput measurements taken from real broadband [7] and HSDPA mobile network datasets [33]. Note, like NAS [46], in our evaluation, we only use samples whose bandwidth is under 3.5 Mbps. This is to trigger the need for adaptive video streaming. (Because high bandwidth network will be able to accommodate the highest rate of encoding.)

## 5.2 Implementation

***Super-resolution hybrid codec:*** We implement the SR hybrid codec module using FFmpeg's API [12] based on H.264 video codec [40] that is the most popular codec [3]. BiSR uses the H.264 video codec to encode two MP4 files to represent the low-resolution key frame encoding file and the non-key frames encoding file, respectively.

Upon decoding, BiSR first decodes the key frame and upscales it using the SR DNN, and then decodes the non-key frames encoding file by referring to the SR key frame. Our above codec process can also be integrated into other video codecs (e.g., H.265 [18], VP9 [42]), but we will not discuss this in the paper. For the inference framework on mobile devices, we use the open-source NCNN [38] framework to accelerate the inference process of DNNs on mobile devices. We develop our own mobile video player using FFmpeg.

***Mobile devices:*** To verify whether BiSR can be deployed on a variety of devices, we use an entry-level smartphone (OPPO R9), a medium-level smartphone (Redmi K30 5G), and a high-level smartphone (Galaxy A90 5G) for the experiment. Their specifications are shown in Table 2.

***SR DNN configuration:*** For each input resolution, BiSR has three DNN models of different complexities for the above three mobile devices. Table 3 in appendix D shows the different SR DNN configurations in detail.

***Training content-aware DNNs:*** The training process of our content-aware DNNs is divided into three stages: (*i*) The downscaling DNN and the SR DNN are cascaded into an auto-encoder structure for end-to-end training. Specifically, the downscaling DNN downscales the high-resolution key frames into low-resolution key frames. Then the SR DNN super-resolves the output of the downscaling DNN into a high resolution. (*ii*) The low-resolution key frames downscaled by DNN are compressed in YUV 444 format using the H.264 encoder, to reduce the streaming bitrate. (*iii*) The compressed low-resolution key frames in the second stage and raw high-resolution key frames are used as the dataset for again training the SR DNN. This enables the SR DNN to perceive the impact of compression, improving the performance of SR DNN on compressed low-resolution key frames.

***Training ABR based on RL:*** We implement BiSR's integrated ABR by modifying Pensieve's implementation [15]. To simulate the computing power of different devices in different states, we use a uniform distribution function from 800 to 1200 microseconds to generate the SR process time. The training process of BiSR's ABR is the same as Pensieve, except that the weight of the entropy function in Pensieve is set to 0.4 to speed up the convergence process.

## 5.3 Baselines

We compare our solution against three baselines:
**1) Traditional Codec:** We use x264 [40], which is a classical H.264 codec implementation, as a baseline.
**2) NEMO:** In NEMO [45], some frames are upscaled by a SR DNN, and other frames are upscaled referring to these SR frames and the upscaled encoded streaming information. NEMO is implemented on VP9 [42], which has a different encoding performance from H.264. For a fairer comparison, the quality of NEMO is normalized with reference to the ratio of H.264 to VP9 quality. NEMO uses the same SR DNN and only super-resolves key frames as BiSR. In NEMO, 240p, 360p, and 480p videos with bitrates of 400 kbps, 800 kbps, and 1200 kbps are upscaled to 1080p. Due to the limited space, we only show the video quality of NEMO on the high-level device.
**3) Pensieve:** Pensieve [26] is an adaptive bitrate algorithm based on reinforcement learning. In our experiments, Pensieve can choose all video chunks encoded by traditional video codec [40].

## 5.4 Results

***Quality across bitrates:*** Figure 6 compares the attained video quality (SSIM) of different encoding bitrates. We contrast the performance of BiSR with that of the H.264 codec and NEMO. Across all bitrates, the attained video quality of NEMO is always worse than using a video codec to encode high-resolution video. Compared to NEMO at the same bitrate, BiSR improves the SSIM video quality by 3.7%. Compared to the traditional H.264 codec, BiSR averagely improves the SSIM video quality by 0.0044, 0.0063 and 0.0075 across all bitrates and video types on entry-, medium-, and high-level mobile devices, respectively. BiSR improves the video quality by more at low video bitrates. It brings 0.012–0.018 of the SSIM improvement at 400kbps. Although there are only minor quality gains here, BiSR saves an average of 12%, 22%, and 26% video coding bitrate across all the same quality levels on entry-, medium-, and high-level mobile devices, respectively. However, compared to the traditional H.264 codec, BiSR brings worse video quality at 2400kbps of game videos on an entry-level mobile device. This is because the low-complexity DNN has worse SR quality, and the performance of DNN is limited at higher encoding bitrates. Further, the SR DNN may deliver worse video quality even if using a high-complexity DNN. To this end, our ABR algorithm can adaptively choose appropriate SR video chunks and traditional video chunks according to the quality performance of SR DNN in video chunks.

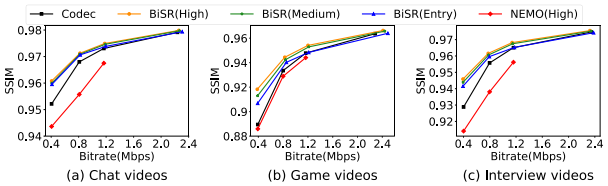
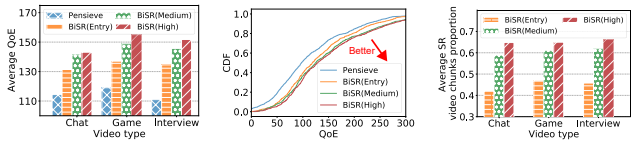



**Figure 6: SSIM with different bitrates on three type videos**

***QoE improvement:*** Figure 7(a) shows the average QoE achieved across all network traces. Compared to Pensieve, BiSR generates a higher QoE on all devices. Specifically, BiSR delivers 17% better QoE for the entry-level device, 26.6% for the medium-level device, and 30.6% for the high-level device across all videos and network traces. Figure 7(b) presents the CDF of QoE scores of all videos achieved across network traces; closer to the bottom right means better QoE. BiSR on the entry-level mobile device can bring better video quality improvement at a lower video bitrate, but cannot bring video quality improvement at a higher video bitrate. Therefore, in Figure 7(b), the QoE of the entry-level mobile device is better than Pensieve when the network bandwidth is low, and it is similar to Pensieve when the network bandwidth is higher. Meanwhile, the QoE of the medium- and high-level mobile devices are always better than the Pensieve because they can bring quality improvement at low and high video bitrates.




(a) Average QoE (b) Distribution of QoE (c) Average SR proportion

**Figure 7: Comparisons of the ABR approaches**

To demonstrate how the ABR algorithm adaptively selects SR chunks according to the performance of SR DNNs, we measure the fraction of SR video chunks delivered (out of all video chunks). Figure 7(c) indicates that the medium- and the high-level device always deliver more SR video chunks than the entry-level device. This is because SR DNNs have a better performance on both medium- and high-level mobile devices than on the entry-level device. Thus, the entry-level device relies more heavily on the video chunks encoding via traditional codec. The figure confirms that BiSR can adapt between these options on a per-device basis.

**Table 2: DNN inference time on mobiles**

| Mobile device | Processor | BiSR(F) | BiSR(H) | RFDN |
|---|---|---|---|---|
| OPPO R9 | SDM 660 | 4461ms | / | 4653ms |
| Redmi k30 5G | SDM 765 | 1584ms | 754ms | 1702ms |
| Galaxy A90 5G | SDM 855 | 2031ms | 1050ms | 2172ms |

∗ SDM: Qualcomm Snapdragon Mobile; H: Half precision; F: Full precision

***Mobile-specific optimized SR DNN and content-aware downscaling DNN:*** RFDN is not designed with mobile computing characteristics in mind, and it does not support half-precision floating-point inference. We next evaluate the efficacy of our optimizations for the SR DNN inference time. Table 2 shows the single thread (for fair comparison) inference time of the SR DNN on three mobile devices. Because the architecture of SDM 660 in OPPO R9 uses ARMv8.0a without the support of half-precision floating-point inference, the time of half-precision floating-point DNN inference time in OPPO R9 is not presented. We see that the SR DNN in BiSR reduces the inference time by 4.1%–6.5% when using the full-precision inference, and 51.7%–55.7% when the model uses the half-precision inference. We also measure the SR performance of RFDN and BiSR. In BiSR, OPPO R9 (entry-level device) uses full-precision DNN inference, Redmi k30 5G (medium-level device), and Galaxy A90 5G (high-level device) use half-precision DNN inference. Figure 8 shows the average PSNR of the above three types of video on three mobile devices for different DNNs. In Figure 8, although the BiSR without DNN downscaling is slightly weaker than RFDN, the SR performance of BiSR is 0.36–0.54 dB better than RFDN. In general, the SR performance of BiSR is always better than RFDN and saves up to 55.7% in inference time.

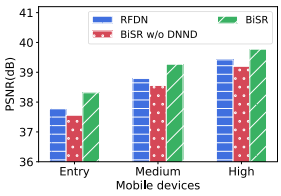



**Figure 8: SR performance (DNND means DNN downscale)**

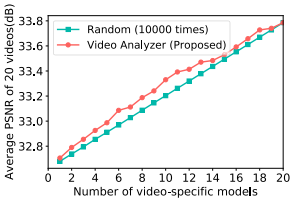



**Figure 9: Average PSNR of 20 videos using the different # of models**

***Video Analyzer:*** Finally, we evaluate the efficacy of the Video Analyzer module. For this, we select 20 test videos from YouTube and train a video-specific model for each one. To study the impact of the Video Analyzer, we vary the number of available models to super-resolve these 20 videos. Any video that has its own available video-specific model uses the model. The remaining videos select one of the available models using two strategies: (*i*) For our Video Analyzer, the remaining videos select the model based on the videos' similarity; and (*ii*) As a baseline, the remaining videos randomly select the SR model. Figure 9 presents the average PSNR of 20 videos on the different number of available models (using these two strategies). The X-axis represents the number of models available, whereas the Y-axis represents the average PSNR for all 20 videos. For example, 5 on the X-axis means that there are 5 per-video models trained. If the X-axis is 20, this indicates all videos use their own pre-trained model. Figure 9 shows that our video analyzer always outperforms the random baseline (average results of 10000 times).

## 6 CONCLUSION

This paper has presented BiSR, a new mobile web video streaming delivery system, which strives to improve perceived video quality. BiSR includes a novel video codec integrated with the downscaling and super-resolution of key frames. Further, BiSR adopts downscaling DNN and proposes a mobile-specific optimized SR DNN to improve the performance of SR DNN. Finally, BiSR designs an RL-based adaptive bitrate streaming mechanism and evaluates its performance on real network traces. With our full-system implementation, we have shown that BiSR reduces the video coding bitrate by 26% compared to the traditional H.264 codec, and improves the SSIM of video by 3.7% compared to NEMO. In general, BiSR improves user quality experience by 17–30.6% on the real network traces compared with Pensieve [26].

## ACKNOWLEDGMENTS

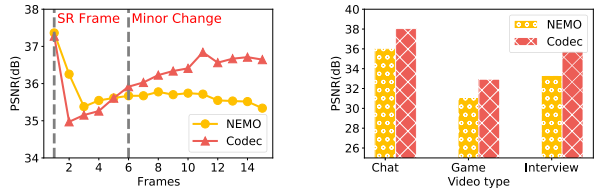
This work is supported by National Natural Science Foundation of China under grant No. 61972189, the Major Key Project of PCL under grant No. PCL2021A03-1, Shenzhen Science and Technology Innovation Commission: Research Center for Computer Network (Shenzhen) Ministry of Education, and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

## REFERENCES

[1] Hot 99.5. 2021. Monsta X Backstage Interview with Elizabethany at HOT 99.5 Jingle Ball 2021. Retrieved January 16, 2022 from https://www.youtube.com/watch?v=K4Vle0JFvQo
[2] Bionews. 2021. #WODC2019 - Exclusive Interview with Amanda Bok (EHC). Retrieved January 16, 2022 from https://www.youtube.com/watch?v=jGoLIPAo1Oc
[3] Bitmovin. 2022. Bitmovin's 5th Annual Video Developer Report 2021. https://go.bitmovin.com/video-developer-report-2021
[4] Ying Chen, Qing Li, Aoyang Zhang, Longhao Zou, Yong Jiang, Zhimin Xu, Junlin Li, and Zhenhui Yuan. 2021. Higher quality live streaming under lower uplink bandwidth: an approach of super-resolution based video coding. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. Istanbul, Turkey.
[5] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. 2021. Fast, accurate and lightweight super-resolution with neural architecture search. In *2020 25th International Conference on Pattern Recognition (ICPR)*. Taichung, Taiwan.
[6] Google Cloud. 2022. Cloud CDN: Content Delivery Network. https://cloud.google.com/cdn
[7] Federal Communications Commission. 2016. Raw Data - Measuring Broadband America. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016
[8] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2014. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*. Zurich, Switzerland.
[9] Big E. 2021. How to do Just Chatting Streams on YouTube Gaming. Retrieved January 16, 2022 from https://www.youtube.com/watch?v=iLDZtE0wnGE
[10] Ericsson. 2022. Ericsson Mobility Report. https://www.ericsson.com/en/mobility-report
[11] ETSI. 2022. Mobile Communications. https://www.etsi.org/technologies/mobile/mobile
[12] FFmpeg. 2022. FFmpeg. https://ffmpeg.org
[13] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Seoul, Korea.
[14] Google. 2022. Youtube.com. https://www.youtube.com/
[15] hongzimao. 2017. Pensieve. https://github.com/hongzimao/pensieve
[16] Siqi Huang and Jiang Xie. 2021. DAVE: Dynamic adaptive video encoding for real-time video streaming applications. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. Virtual Conference, 1–9.
[17] itu. 2021. H.264: Advanced video coding for generic audiovisual services. https://www.itu.int/rec/T-REC-H.264
[18] itu. 2021. H.265 : High efficiency video coding. https://www.itu.int/rec/T-REC-H.265
[19] Heewon Kim, Myungsub Choi, Bee Lim, and Kyoung Mu Lee. 2018. Task-aware image downscaling. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Munich, Germany.
[20] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. Amsterdam, The Netherlands.
[21] Sebastian Lambert. 2022. Number of Internet Users in 2022/2023: Statistics, Current Trends, and Predictions. https://financesonline.com/number-of-internet-users/
[22] Qing Li, Ying Chen, Aoyang Zhang, Yong Jiang, Longhao Zou, Zhimin Xu, and Gabriel-Miro Muntean. 2022. A Super-Resolution Flexible Video Coding Solution for Improving Live Streaming Quality. *IEEE Transactions on Multimedia* (2022).
[23] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. 2017. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. Honolulu, Hawaii, USA.
[24] Jie Liu, Jie Tang, and Gangshan Wu. 2020. Residual feature distillation network for lightweight image super-resolution. In *European Conference on Computer Vision*. Virtual Conference.
[25] Xiaoteng Ma, Qing Li, Longhao Zou, Junkun Peng, Jianer Zhou, Jimeng Chai, Yong Jiang, and Gabriel-Miro Muntean. 2022. QAVA: QoE-aware adaptive video bitrate aggregation for HTTP live streaming based on smart edge computing. *IEEE Transactions on Broadcasting* 68, 3 (2022), 661–676.
[26] 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*. Los Angeles, CA, USA.
[27] Ofir Pele and Michael Werman. 2008. A linear time histogram metric for improved sift matching. In *Computer Vision–ECCV 2008*. Marseille, France.
[28] Ofir Pele and Michael Werman. 2009. Fast and robust earth mover's distances. In *2009 IEEE 12th International Conference on Computer Vision*. Kyoto, Japan.
[29] Yuxin Peng and Chong-Wah Ngo. 2005. EMD-based video clip retrieval by many-to-many matching. In *International Conference on Image and Video Retrieval*. Singapore, Singapore.
[30] Matt Irwin Photography. 2021. THE LATEST-NEWS | Sigma - Fuji, Nikon or Canon ML? | CANON EXOTIC GLASS | Z9 More Cover | Matt Irwin. Retrieved January 16, 2022 from https://www.youtube.com/watch?v=NtS2SCpz9-w
[31] Andrea Pimpinella, Andrea Marabita, and Alessandro E. C. Redondi. 2021. Crowdsourcing or Network KPIs? A Twofold Perspective for QoE Prediction in Cellular Networks. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. Nanjing, China.
[32] Sean Potts. 2021. League of Legends - TFT - MacBook Air M1 16gb RAM & 7-core GPU - Medium-max settings. Retrieved January 16, 2022 from https://www.youtube.com/watch?v=_ljrdtYwJ10
[33] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*. Oslo, Norway.
[34] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711.
[35] Evan Klinger & David Starkweather. 2010. Phash. https://www.phash.org/
[36] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP– standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*. San Jose, CA, USA.
[37] Wanjie Sun and Zhenzhong Chen. 2020. Learned image downscaling for upscaling using content adaptive resampler. *IEEE Transactions on Image Processing* 29 (2020), 4027–4040.
[38] Tencent. 2022. GitHub - Tencent/ncnn: ncnn is a high-performance neural network inference framework optimized for the mobile platform. https://github.com/Tencent/ncnn
[39] Throneful. 2021. Hearthstone (2021) - Gameplay (PC UHD) [4K60FPS]. Retrieved January 16, 2022 from https://www.youtube.com/watch?v=hUi0eFuTi-g
[40] VideoLAN. 2022. x264. https://www.videolan.org/developers/x264.html
[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
[42] WebM. 2022. VP9 Video Codec. https://www.webmproject.org/vp9/
[43] Daniel Weinberger. 2022. Choosing the Right Video Bitrate for Streaming HLS and DASH. https://bitmovin.com/video-bitrate-streaming-hls-dash/
[44] Mingqing Xiao, Shuxin Zheng, Chang Liu, Yaolong Wang, Di He, Guolin Ke, Jiang Bian, Zhouchen Lin, and Tie-Yan Liu. 2020. Invertible image rescaling. In *European Conference on Computer Vision*. Virtual Conference.
[45] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2020. Nemo: enabling neural-enhanced video streaming on commodity mobile devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. London, United Kingdom.
[46] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural adaptive content-aware internet video delivery. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*. Carlsbad, CA.
[47] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: neural video enhancement at scale. In *Proceedings of the ACM SIGCOMM 2022 Conference*. Amsterdam, Netherlands, 795–811.
[48] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. London, United Kingdom.
[49] Yinjie Zhang, Yuanxing Zhang, Yi Wu, Yu Tao, Kaigui Bian, Pan Zhou, Lingyang Song, and Hu Tuo. 2020. Improving quality of experience by adaptive video streaming with super-resolution. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. Virtual Conference, 1957–1966.

# A RESULTS OF NEMO


**Figure 10: An example with minor motion (arms moving)**



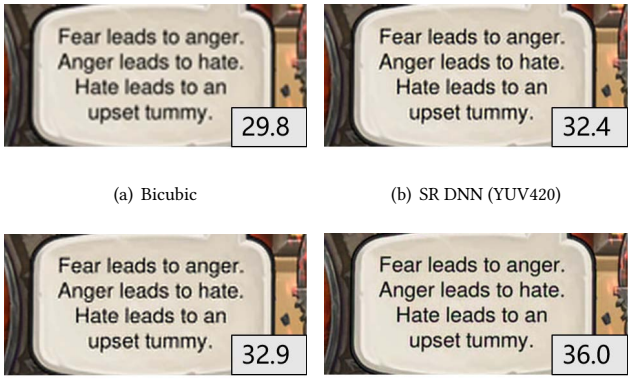

(a) PSNR between codec and NEMO in the video of Figure 10

(b) PSNR between codec and NEMO in all videos

**Figure 11: The PSNR results using the high-complexity SR DNN**

Figure 10 shows the example of several frames in this scene, highlighting the arm motion. Figure 11(a) shows the PSNR quality of each frame in the video chunk (Figure 10). Meanwhile, Figure 11(b) further shows NEMO's PSNR across the three video types using a high-complexity SR DNN (note, each type of video includes two videos). When super-resolving the first frame of each video chunk, the performance of using NEMO to upscale low-resolution videos is always worse than using a traditional video codec to encode high-resolution video (with the same bitrate).

# B AN EXAMPLE OF SUPER-RESOLUTION FRAME




(a) Bicubic

(b) SR DNN (YUV420)

(c) SR DNN (YUV444)

(d) BiSR (SR DNN + YUV444 + downscaling DNN)

**Figure 12: Super-resolution performance of BiSR in PSNR**

To highlight the benefits of our approach, Figure 12 offers an example of a super-resolution frame using four different techniques: (*i*) Bicubic means bicubic upscale with YUV 420 format; (*ii*) SR DNN(YUV 420) means DNN upscale with YUV 420 format; (*iii*) SR DNN(YUV 444) means DNN upscale with YUV 444 format; and (*iv*) BiSR (SR DNN + YUV444 + downscaling DNN) means DNN downscale and upscale with YUV 444 format.

Using bicubic (Figure 12(a)), the PSNR of the frame is only 29.8 dB. When the frame format is from YUV420 to YUV 444, the PSNR of the frame that is upscaled by SR DNN from 32.4 dB (i.e., Figure 12(b)) increases to 32.9 dB (i.e., Figure 12(c)). As can be seen in Figure 12, this still results in poor image quality. In contrast, BiSR uses a DNN to downscale the frame and encode the frame at the YUV444 format. The PSNR of the upscaled frame using SR DNN increases to 36.0 dB (i.e., Figure 12(d)), demonstrating the effectiveness of downscaling DNN and SR DNN based on YUV444 format.

# C THE ALGORITHM OF VIDEO ANALYZER

Algorithm 1 shows how Video Analyzer divides videos into different clusters.

**Algorithm 1:** Video Analyzer

**input** : Input $N$ Videos: $V_{set}$ $(v_1, v_2, ..., v_N)$;
Number of clusters: $K$;
Threshold that clusters key frames: $threshold1$

**output**: $K$ videos cluster: $C_{set}$ $(C_1, C_2, ..., C_K)$;

$C_{set}$ = Hierarchical-Clustering($V_{set}$, $K$, Distance-Calculator)

**return** $C_{set}$

**Function** *Distance-Calculator($V_A$, $V_B$, threshold1)*:
  $RF_A \leftarrow$ Get-Reference-Frame($V_A$, $threshold1$);
  $RF_B \leftarrow$ Get-Reference-Frame($V_B$, $threshold1$);
  $distance \leftarrow$ Earth-Mover's-Distance($RF_B$, $RF_A$);
  **return** $distance$

**Function** *Get-Reference-Frame(V, threshold)*:
  $KF \leftarrow$ Get-Key-Frame{$V$};
  $RF[1] \leftarrow KF[1]$;
  $prev\_frame \leftarrow KF[1]$;
  **for** $i \leftarrow 2$ **to** $length(KF)$ **do**
    **if** $Phash(prev\_frame, KF[i]) > threshold$ **then**
      $RF.append(KF[i])$;
    $prev\_frame \leftarrow KF[i]$;
  **return** $RF$

- Hierarchical-Clustering: return $K$ video clusters based on Distance-Calculator
- Distance-Calculator: return the similarity between two videos
- Get-Reference-Frame: return the clusters of key frames of a video
- Earth-Mover's-Distance: return the distance of the key frame clusters between two videos
- Get-Key-Frame: return all key frames of an input video
- Phash: return phash value of two input frames

# D SR DNN CONFIGURATION

**Table 3: DNN-model configurations (#Block, #Channel, #Size)**

| Input resolution | DNN quality level | | |
|---|---|---|---|
| | Low | Medium | High |
| 270p | 1, 48 | 2, 48 | 4, 48 |
| | 291 KB | 456 KB | 788 KB |
| 540p | 1, 16 | 1, 32 | 1, 48 |
| | 31 KB | 108 KB | 229 KB |

Table 3 shows the configurations of different SR DNNs for mobile devices. This includes the number of RFDB blocks [24] (with our channel attention module), channel dimensions, and the size of DNN models.