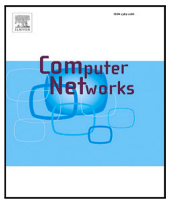
Computer Networks 257 (2025) 110994

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



ABUV: Adaptive bitrate and upsampling for video streaming on mobile devices

Yichen Lu [b], Ji Qi [a], Sheng Zhang [b,*], Gangyi Luo [a], Andong Zhu [b], Jie Wu [c], Zhuzhong Qian [b]

[a] China Mobile (Suzhou) Software Technology Co., Ltd., Suzhou, Jiangsu, China
[b] State Key Laboratory for Novel Software Technology, Nanjing University, China
[c] Center for Networked Computing, Temple University, Philadelphia, PA 19122, USA

ARTICLE INFO

Keywords:
Video streaming
Mobile computing
Super-resolution
Deep neural networks
Reinforcement learning

ABSTRACT

Fueled by the popularity of mobile devices, mobile channels have become the preferred video delivery medium. However, users often encounter a poor quality of experience (QoE) due to bandwidth limitations, despite the implementation of adaptive bitrate (ABR) techniques. Recent advancements in super-resolution (SR) models have offered a potential solution to this situation, while the process of SR on mobile devices can introduce energy overhead and latency. To tackle these issues, we present ABUV, a system designed to enhance mobile video streaming by integrating adaptive bitrate and super-resolution technologies. ABUV leverages deep reinforcement learning to dynamically adjust both the bitrate and upsample decisions jointly based on considerations such as energy overhead and available bandwidth. It employs an optimized SR model specifically tailored for mobile devices, selectively applying the upsample process to chosen frames. Additionally, ABUV incorporates user-specific streaming information and adapts to the unique network environment through online training. In our experiments, we evaluate ABUV using real network traces and a diverse collection of videos, and the results show that ABUV can save up to 59% of data consumption and improve QoE by 27% compared to other video streaming systems.

1. Introduction

With the continuous increase in the number of mobile users, recent reports indicate that the average user spends over 5 h per day using their mobile phones [1,2]. Concurrently, there has been an exponential surge in the demand for mobile video streaming, with mobile devices contributing to over 75% of all video views [3]. However, the ever-changing network environment poses a significant challenge to seamless video streaming for mobile users. The instability and fluctuations in mobile networks often lead to issues such as video stuttering, blurriness, and prolonged loading times, significantly impacting user satisfaction and engagement. Mobile users have increasingly high expectations for video quality, prompting video streaming service providers to strive for higher resolutions and an enhanced user experience.

Dynamically Adaptive Streaming over HTTP (DASH) is a widely employed video streaming technology that utilizes Adaptive Bitrate (ABR) algorithms to enhance users' Quality of Experience (QoE) under fluctuating network conditions. These ABR algorithms collectively aim to determine the optimal bitrate for each video chunk, taking into account the current network conditions. Nevertheless, they face challenges in upholding QoE, particularly in scenarios with limited bandwidth, as they must strike a delicate balance between video smoothness and quality.

Super-resolution (SR) is a powerful tool that can reconstruct high-resolution images from low-resolution ones. Researchers have investigated the application of SR models to videos [4–6], and recent studies have specifically aimed at leveraging these models to enhance video streaming [7–9]. While SRAVS [9] integrates SR into adaptive streaming to enhance user QoE, it relies on a PC with a GPU to run complex neural networks, making it unsuitable for mobile devices which often lack sufficient computing resources. While NEMO [8] has been successful in upsampling videos on mobile devices using the VP9 codec, it is limited by a fixed upsampling factor and the inability to consider bitrate selection. As a result, it struggles to enhance the quality of experience (QoE) effectively, which hampers NEMO's ability to adapt to dynamic changes in the network environment and deliver optimal video quality.

Based on SR models, we have the capability to upsample low-definition video chunks that are fetched in low-bandwidth environments. However, this process introduces additional energy overhead and potentially increases the inference latency, which can degrade the user's QoE. Therefore, before running client-side upsampling, it is

* Corresponding author.
E-mail addresses: sheng@nju.edu.cn (S. Zhang), jiewu@temple.edu (J. Wu).

https://doi.org/10.1016/j.comnet.2024.110994
Received 19 April 2024; Received in revised form 19 November 2024; Accepted 11 December 2024
Available online 25 December 2024

crucial to optimize the SR models specifically for mobile devices. The goal is to minimize energy overhead and inference time while maintaining the improvement in video quality. Furthermore, the definition of the fetched video chunk also affects the complexity of running SR. It becomes a challenging task to determine which bitrate of the chunk to fetch and which upsampling factor to apply. Balancing the energy consumption and video quality is essential, and finding a solution to achieve this balance is critical. Reinforcement learning emerges as an important tool to tackle this problem, as it allows us to learn the complex relationships among bitrate, upsampling factor, network environment, and battery information.

In this paper, we present a novel system for video streaming on mobile devices, which combines ABR and SR models to enhance the viewing experience. To the best of our knowledge, ABUV represents the first energy-efficient mobile video streaming system capable of dynamically adjusting both the bitrate and upsampling factors in a synchronized manner. Our contributions are outlined as follows:

- We introduce an agent based on Proximal Policy Optimization (PPO) [10] that dynamically adjusts bitrate and upsampling decisions jointly in response to real-time network and energy consumption, instead of conventionally changing bitrate with a fixed upsample factor.
- We optimize the super-resolution model, ensuring its lightweight nature to facilitate efficient video upsampling on resource-constrained mobile phones.
- We enable our system, ABUV, to perform adaptive retraining based on user-specific video transmission history, enabling improved adaptation to diverse mobile environments.
- To validate our approach, we conduct experiments employing real-world network traces and videos to train and evaluate ABUV. Our results demonstrate that ABUV achieves up to 59% reduction in data consumption while enhancing the QoE by 27%.

2. Related works and background

Adaptive bitrate streaming, commonly used in DASH [11] and HTTP live streaming (HLS) [12], facilitates the streaming of video content at varying bitrates according to the available network bandwidth. Videos are encoded into multiple bitrates and segmented into fixed-length chunks. During streaming, the client-side employs an ABR algorithm to dynamically select the appropriate chunk based on current network conditions.

The current state-of-the-art ABR algorithms can be categorized into two main types: rule-based ABR [13,14] and learning-based ABR [15,16]. Rule-based ABR algorithms utilize predefined rules to determine the optimal bitrate for the given network conditions. For instance, BOLA [17] is a classic rule-based ABR algorithm that selects the download bitrate of each video chunk based on a utility function considering the segment's quality level and buffer occupancy at the time of downloading. BOLAE [18] extends BOLA by incorporating available bandwidth and latency estimation into the bitrate decision process, leading to improved video quality and reduced rebuffering time. MPC [19] adjusts the video bitrate in real-time based on feedback from the video buffer and the available network bandwidth.

On the other hand, learning-based ABR algorithms employ machine learning to predict the best bitrate selection. Pensieve trains a neural network model based on observations such as occupancy, throughput, and latency, collected from client video players. Similarly, Fugu combines classical control strategy with a learning network predictor trained on data from real deployment environments.

Among these approaches, GreenABR represents a significant advancement by introducing energy consumption considerations into ABR decisions. It develops a power model and evaluates QoE using the standard perceptual quality metric VMAF rather than just the video bitrate. However, GreenABR has several limitations: (1) it can only adjust bitrate selection to save energy, without the capability to enhance video quality through super-resolution; (2) its fixed decision strategy cannot adapt to diverse mobile environments and user preferences; and (3) the power model it uses may not accurately reflect the complex energy consumption patterns in modern mobile devices, especially when processing high-resolution videos. In contrast, our ABUV system addresses these limitations by: (1) integrating super-resolution capabilities to enhance video quality while maintaining energy efficiency; (2) employing adaptive retraining to accommodate different network environments and user patterns; and (3) incorporating real-time energy consumption feedback for more accurate decision-making.

While OnRL takes a different approach by using online training instead of offline training and adapts its models over time based on real-world user-specific network traces, it still lacks the ability to enhance video quality through super-resolution and cannot make joint decisions on bitrate selection and upsampling factors like ABUV does.

Super-resolution is a technology that enhances the resolution of low-resolution images using various methods. Recent advances in SR research include deep learning-based approaches that use convolutional neural networks (CNN) [20–22], generative adversarial networks (GAN) [23,24], or probabilistic models [25] to learn the mappings between low and high-resolution domains. Although video is more complex than a single image, recent advances in super-resolution research have led to the development of many effective video super-resolution (VSR) systems [4]. Real-ESRGAN uses a multi-scale architecture and a perceptual loss function to enhance the quality of super-resolved images. It performs well in scenarios where high-resolution images are not available during training and can be useful in various applications, such as image and video editing, medical imaging, and surveillance [6]. BasicVSR++ [5] leverages a spatial–temporal alignment module to align the input frames accurately, which helps enhance the spatial details and temporal coherence of super-resolved videos.

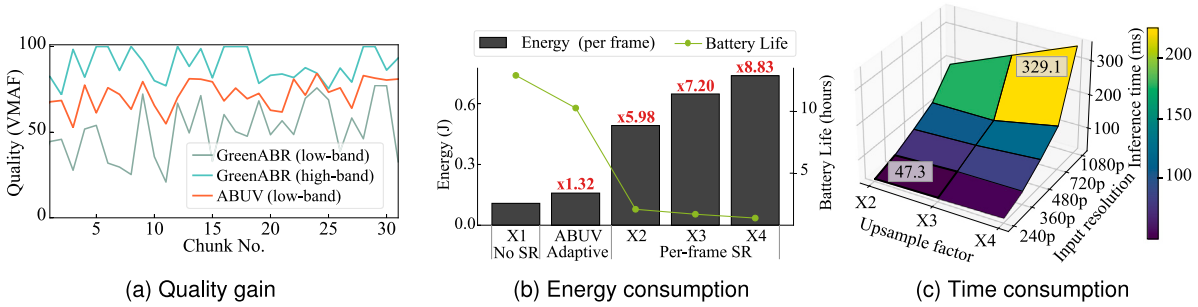
To overcome the challenges of applying SR models on mobile devices with limited resources, several innovative frameworks have emerged, each with distinct trade-offs:

- MobiSR [26] optimizes hardware collaboration among CPU, GPU, and DSP to enable real-time SR, but focuses solely on image enhancement without considering video streaming dynamics.
- NEMO [8] achieves mobile-friendly video upsampling through VP9 codec optimizations and frame-level cache reuse. However, its fixed upsampling approach limits adaptation to varying network conditions.
- BiSR [27] improves efficiency by selectively applying SR to key frames, reducing computational overhead. Yet, it treats bitrate selection and upsampling as separate decisions, missing opportunities for joint optimization.
- SRAVS [9] integrates SR into adaptive streaming but relies on PC-based GPUs, making it impractical for mobile devices.

These existing approaches highlight several key challenges in mobile video streaming enhancement: (1) the need for efficient SR models that can run on resource-constrained devices, (2) the importance of dynamic adaptation to network conditions, and (3) the challenge of balancing video quality with energy consumption. ABUV addresses these challenges through joint optimization of bitrate and upsampling decisions, mobile-optimized SR models with adaptive retraining capabilities, and comprehensive consideration of network conditions, video quality, and energy constraints.

3. Motivation

The fluctuating nature of mobile users' network bandwidth challenges pure ABR algorithms in delivering high-quality video experiences consistently. Employing client-side SR, given recent advancements in smartphone GPUs, is both imperative and feasible. By using SR



(a) Quality gain (b) Energy consumption (c) Time consumption

Fig. 1. Motivating measurements for adaptive bitrate and upsample factor.

models to enhance low-resolution video segments, significant improvements can be achieved. However, frame-by-frame upsampling with a fixed upscaling factor demands substantial time and energy resources, potentially affecting mobile phone battery life. We conducted experiments using a Mi 10 (SnapDragon 865) running the Android platform, utilizing videos from YouTube [28].

Limitations of ABR. ABR algorithms face the challenge of compromising video quality to maintain smooth playback under bandwidth constraints. Harnessing SR models can mitigate this trade-off by enhancing low-definition video segments. Our experiment compares GreenABR [16] with our proposed ABUV, which integrates SR. Fig. 1(a) highlights the potential of SR in delivering high-quality videos in low-bandwidth environments. ABUV outperforms GreenABR by making joint decisions regarding bitrate and upscaling factor, optimizing the downloading and upscaling process in low-bandwidth scenarios.

Limited throughput of SR. Despite ABPN [29] being a lightweight SR model, real-time processing is unattainable due to excessive time consumption. Inference time increases for higher resolutions and larger upsample factors. Fig. 1(c) depicts the relationship between upsample factor, input resolution, and inference time. To deliver a seamless viewing experience, it is crucial to address frame selection for upsampling and develop lightweight models for real-time upsampling on mobile devices.

Excessive energy consumption of SR. We evaluate the energy consumption of mobile devices executing ABPN with various upsample factors, measuring the current battery level using Android API [30]. Fig. 1(b) shows per-frame energy consumption and battery life. Per-frame SR incurs significant energy overhead, reducing battery life substantially. Adaptive upsampling in ABUV selectively applies SR when necessary, achieving a balance between video quality and energy consumption.

Challenges in Joint ABR and SR Implementation. The integration of ABR and SR techniques on mobile devices presents several interconnected challenges:

First, developing lightweight SR models is essential for real-time upsampling on resource-constrained mobile devices. These models must minimize computational complexity and reduce inference time while maintaining acceptable upsampling quality. The challenge intensifies when considering that lower bitrate selections, while reducing network load, increase the computational demands for SR processing.

Second, the system must simultaneously optimize two interdependent decisions — bitrate selection and upsampling factor. This joint optimization is particularly complex as network conditions and device battery levels constantly fluctuate. Higher upsampling factors can improve video quality but at the cost of increased energy consumption and processing time. Additionally, to prevent stuttering and minimize latency, careful chunk selection for upsampling becomes necessary.

Third, the system must efficiently manage resources by balancing computational demands between video decoding and SR processing while maintaining acceptable battery consumption levels. This requires a dynamic trade-off between video quality improvements through SR, energy consumption from processing, network bandwidth usage, and buffer occupancy levels.

ABUV addresses these challenges through an integrated approach using the Proximal Policy Optimization (PPO) algorithm [10]. Through iterative training, ABUV learns optimal policies for joint adaptive decision-making, capturing intricate relationships among network conditions, video quality, computational resources, and energy consumption. In contrast to NEMO and BiSR, which separate bitrate decisions from upsampling, ABUV simultaneously adjusts both bitrate and upsample factors, considering the complete system state. This holistic approach achieves a balance between video quality, playback smoothness, and energy efficiency.

4. ABUV designs

ABUV is a mobile video streaming system that combines ABR and SR decisions. These decisions are made on the server-side, while the upsampling process occurs on the client-side. The overall system architecture is illustrated in Fig. 2.

RL for Adaptive Decisions. ABUV utilizes a reinforcement learning (RL) agent to make informed decisions regarding bitrate and upsampling. The RL agent is based on the Proximal Policy Optimization (PPO) algorithm and is trained within a predefined environment that emulates the video streaming process. Once trained, the RL agent receives input from the client-side video player and server-side preknown data and dynamically adjusts bitrate and upsampling settings as needed. To accommodate changing network conditions, ABUV incorporates adaptive retraining. When performance degradation is detected, the RL agent undergoes retraining using user-specific streaming history, allowing it to enhance its decision-making capabilities.

SR for Video Upsampling. ABUV introduces the ability to select different upsampling factors by utilizing various ABPN-light models for super-resolution. Initially, a large dataset is employed for pretraining to obtain the base weights once in an offline manner. Subsequently, ABUV utilizes both high-resolution and low-resolution versions of the target video to train content-aware super-resolution models. During the video streaming process, when a mobile user requests a specific bitrate version of a video chunk from the server and decodes it using the video player, ABUV's client-side implementation comes into play. It performs upsampling on the decoded frames based on the adaptive decisions made by the RL agent. This upsampling process significantly enhances the quality of the fetched low-resolution (LR) video chunk, resulting in output super-resolution (SR) frames. Consequently, users can enjoy videos with greatly improved visual quality within the same mobile device and environment.

In the following sections, we will provide further details on the SR model (Section 4.1) and RL model (Section 4.2) employed in ABUV, our mobile video streaming system.

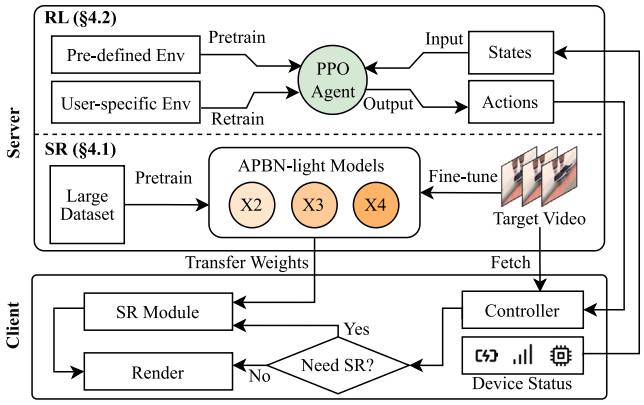


Fig. 2. ABUV system overview.

4.1. Mobile video super-resolution

To enhance the efficiency of super-resolution on mobile devices, we conduct performance tests on various state-of-the-art SR models. After careful evaluation, we selected the anchor-based plain net (ABPN) [29] as the most suitable choice, considering its favorable balance between effectiveness and computational cost. In order to further improve its efficiency, we have customized the ABPN model at the layer level, resulting in a highly optimized variant known as ASPN-light. This model exhibits remarkable performance, enabling real-time delivery of 2K videos on mobile devices.

4.1.1. Model selection

To perform super-resolution on the mobile client side, the model must be able to output frames in real-time. For instance, if a low-resolution chunk is to be upsampled by 4 times, each frame in that chunk shall be upsampled. As a result, the model shall be able to output frames in real-time after decoding the chunk and before rendering the frames. It is not feasible to do upsampling in advance (e.g., as soon as the chunk is downloaded [9]) as chunk-level reconstruction brings extra decoding and encoding overheads, which is not acceptable for mobile users.

Given that super-resolution models demand substantial computing resources and memory, it is vital to strike a delicate balance between computational complexity and performance to achieve real-time super-resolution on mobile client devices. Moreover, energy consumption and memory usage must also be taken into consideration, as battery life is one of the Quality of Experience (QoE)-related metrics for mobile users. Additionally, when adaptive upsampling during video streaming is required, models for various upsample factors (X2/X3/X4) need to be delivered and loaded on the client side, making models with smaller sizes and lower memory usage preferable.

To determine the most suitable super-resolution model for mobile devices, we conducted tests using the Vimeo-90K dataset [31] and state-of-the-art mobile super-resolution models. Our evaluation criteria encompassed quality improvement, inference time, energy consumption, and memory usage. Instead of relying solely on PSNR, we employed VMAF [32] as the video quality metric. VMAF accounts for critical factors such as color accuracy, contrast, and sharpness, which are essential for human perception of video quality. Our test results on the Android platform (Xiaomi 10, SnapDragon 865) revealed that RealSR [33] achieved the highest quality improvement, albeit with an average frame upsampling time of 702 ms. In contrast, XLSR [34] provided real-time upsampling (over 30 frames/s) but with a slightly lower quality gain. ABPN [29] demonstrated a favorable balance between VMAF improvement and acceptable latency, indicating its efficiency potential. Consequently, we selected ABPN for integration into our mobile video streaming system. Fig. 3 presents the test results for upscaling videos from 720p to 1440p, showcasing both video quality and the average inference latency per frame.

4.1.2. Super-resolution optimization

While ABPN has undergone optimization efforts, including INT8 quantization and anchor-based learning, it faces limitations in fully supporting ultra-high-resolution videos. For example, upscaling a 360p image to 1440p using ABPN results in an inference latency exceeding 70 ms. Additionally, minimizing memory consumption is crucial to enable the simultaneous loading of models for various upsampling factors, enhancing the system's adaptability. To address these challenges, we have implemented several optimizations and introduced ABPN-light, a lighter version with a simplified network structure. ABPN-light maintains real-time performance while supporting 2K video output. These optimizations effectively reduce both inference latency and memory consumption, facilitating the seamless integration of SR into ABUV.

Layer-level Modification. The adaptive upsample factor approach requires downloading and initializing models for different upsampling factors, making model size optimization crucial for efficient deployment. The original ABPN model, with a file size of approximately 58 kB, serves as our starting point. To support multiple upsampling factors (X2/X3/X4) efficiently throughout the video streaming process, we optimized the neural network architecture of ABPN. Specifically, we reduced the model complexity by removing three deep feature extraction layers while retaining one essential layer. To compensate for potential quality degradation from this reduction, we optimized the number of channels, which was previously fixed at 3 in ABPN. Fig. 4 illustrates the resulting network structure of ABPN-light.

These architectural modifications yielded substantial reductions in model size while preserving satisfactory video quality. Although the changes introduce a minor quality degradation of approximately 0.4 dB in PSNR, they successfully reduce the total number of parameters from 42.54K to 21.29K and compress the model file size from 58 kB to under 27 kB. This optimization facilitates efficient model deployment in practice — downloading all three models with different upsampling factors (totaling less than 90 kB) consumes less bandwidth than retrieving a typical 240p video chunk (300 kbps, 30 fps, 4 s).

Key Frames Only. To further optimize inference latency and energy consumption, we implemented a selective super-resolution strategy inspired by BiSR [27]. This innovative approach strategically applies super-resolution processing to key frames (I-frames) only, while intelligently propagating these enhancements to non-key frames (P and B frames) through their dependency relationships. By leveraging the hierarchical structure inherent in H.264 video encoding, where non-key frames are derived from key frames through motion prediction, this method achieves efficient quality enhancement across the entire video sequence while significantly reducing computational overhead.

The implementation required careful modification of the FFmpeg library and H.264 decoder, which we subsequently integrated into Google Exoplayer. This optimization significantly reduces both computational demands and inference time, enabling real-time super-resolution on mobile platforms. Our evaluation shows that the chunk-level inference latency is reduced by at least a factor of 4, though the actual improvement varies with the key frame distribution in different videos. When combined with our Layer-level Modification, the overall inference latency decreases from 70 ms to an average of 31 ms per frame, as shown in Fig. 5(a), representing a more than 2x improvement in processing speed.

Activation Function. The choice of activation function is pivotal in convolutional neural network-based SR models, such as the innovative ABPN-light. Moving beyond the traditional ReLU, ABPN-light harnesses the capabilities of Parametric ReLU (PReLU) [35], a sophisticated variant that integrates a learnable parameter within each channel. This subtle yet powerful adjustment allows for a more dynamic adaptation during the learning process. By integrating PReLU, ABPN-light not

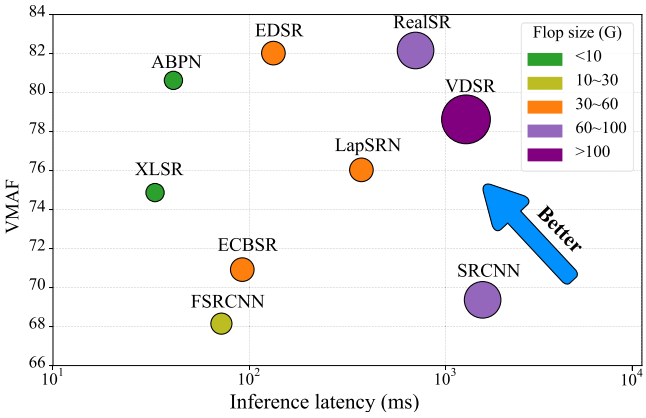


Fig. 3. Mobile super-resolution models comparison. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

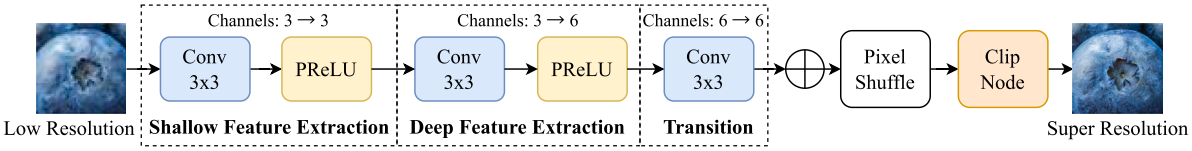


Fig. 4. The network architecture of ABPN-light.

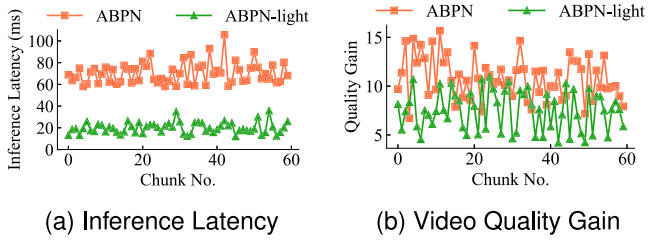


Fig. 5. ABPN-light vs. ABPN.

only enhances its inference performance but does so with a negligible increase in energy consumption. The adaptive nature of PReLU ensures that the model can fine-tune its activation pathways, resulting in a more nuanced and efficient mapping of input to reconstructed high-resolution outputs. This enables ABPN-light to set a new benchmark for effective and energy-conscious super-resolution in image processing.

Content-aware Training. Super-resolution models are typically trained using general dataset training or content-aware learning. Content-aware learning trains and evaluates the model using low-resolution and high-resolution versions of a specific video, leveraging content-specific information for improved performance.

Training the model with a single target video may be insufficient due to its limited length. We use transfer learning [36] to fine-tune the model on our server-side infrastructure. First, a large dataset (e.g., DIV2K [37]) is used for pretraining once and offline. Then, for each video, we perform content-aware fine-tuning on the server using its low-resolution and high-resolution versions. While this approach requires storing a separate model for each video on the server, the storage overhead is minimal due to ABPN-light's compact size (less than 27 kB per model). When a user requests a video, the corresponding fine-tuned models are efficiently transmitted to the mobile device along with the video chunks, adding negligible overhead to the streaming process.

We conduct experiments on a YouTube video [28], upscaling an image from 360p to 1080p using three methods: fine-tuned ABPN, fine-tuned ABPN-light, and generally-trained ABPN-light. Fig. 6 presents visual results. The upscaled image quality is acceptable, with some loss compared to the original high-resolution image. ABPN-light, fine-tuned through content-aware learning, outperforms training with a general dataset.

Our optimization results in ABPN-light for real-time video upscaling on mobile devices. By reducing parameters and improving inference latency, ABPN-light achieves efficient performance. The server-side content-aware learning and transfer learning techniques, combined with the model's small size, demonstrate the effectiveness of these optimizations in enhancing super-resolution models while maintaining practical deployment feasibility. This approach enables the ABUV system to deliver seamless real-time super-resolution on mobile devices with video-specific optimization.

4.2. Adaptive bitrate and upsample factor decision

ABR algorithms adapt video quality based on network conditions. Traditional rule-based adaptive bitrate algorithms primarily consider network throughput for bitrate selection. However, for mobile devices, adaptive upsampling decisions should also account for remaining battery capacity. To understand the complex relationship between these variables, we employ deep reinforcement learning techniques to learn mappings and make adaptive decisions in our research. Although Pensieve [15], an A3C-based ABR algorithm, has gained popularity in the research community, we have opted to use PPO [10] as our intelligent agent. PPO replaces A3C and incorporates a novel objective function that includes clipping surrogate gradients, preventing large gradient updates. This approach significantly improves the stability and efficiency of the learning process, ultimately enhancing our intelligent agent's performance.

4.2.1. PPO-based RL model

Fig. 7 depicts the architecture of ABUV's RL model that we built based on PPO, which reads the states from video streaming environment and makes adaptive decisions accordingly.

States. To make a decision, the agent needs to collect information from the environment. Some pre-known information is available from the server-side without additional communication, such as the sizes and video qualities of different bitrates for the next chunk. Other information must be transferred from the client-side, including the

Fig. 6. Visual results of different models and training sets.

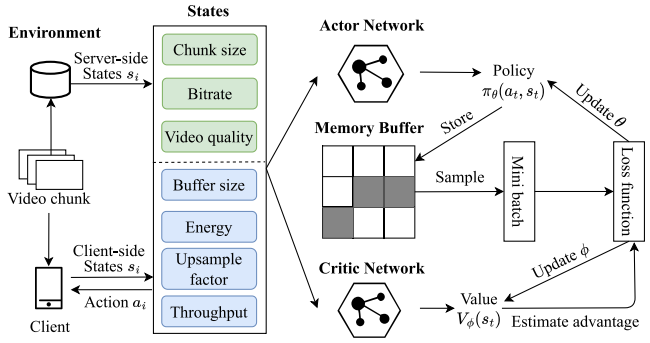


Fig. 7. The reinforcement learning architecture.

buffer size, as well as the time and energy consumption of the previous chunk.

Actions. ABR algorithms allow the client to select the bitrate and fetch the chunk from the server. In our case, the agent also selects the upsampling factor to perform content-aware upsampling. The agent can choose one of four upsampling factors: 1 (no super-resolution), 2, 3, and 4. The actions are a combination of bitrate and an upsampling factor and can be represented as $\{b, u\}$ where b is the bitrate and u is the upsampling factor.

Rewards. To measure the agent's performance, we define the QoE reward function as:

$$QoE_i = \alpha \times q_i - \beta \times (q_i - q_{i-1}) - \eta \times p_i - \omega \times E_i, \tag{1}$$

where q_i represents chunk i's video quality, $(q_i - q_{i-1})$ is the smoothness penalty, p_i is the rebuffer penalty, and E_i is the energy consumption penalty. These metrics are weighted by different impacting factors $\alpha, \beta, \eta, \omega$ and summed. Video quality q_i is calculated using the Video Multimethod Assessment Fusion metric (VMAF) and ranges from 0 to 100, making smoothness change fall within $[-100, 100]$. Rebuffer penalty p_i is rebuffering time in seconds, typically less than 30 for one video chunk.

The energy consumption penalty E_i is determined through comprehensive benchmark measurements on the target device. We conducted extensive experiments to measure the energy consumption for all possible combinations of bitrates and upsampling factors, using the Android Battery Manager API [30]. For each combination, we performed multiple runs and calculated the average energy consumption to create a device-specific energy consumption lookup table. During runtime, E_i is obtained by querying this pre-computed table based on the selected bitrate and upsampling factor for chunk i. Note that this energy profiling is device-dependent and needs to be performed once for each new hardware platform to create its corresponding energy consumption profile.

The reward function aims to maximize video quality while minimizing energy consumption and rebuffering time. To ensure QoE_i meaningfully guides the agent, metrics are normalized by adjusting parameters $\alpha, \beta, \eta, \omega$. In our ABUV experiment, to maximize video quality without excessive energy overhead, they are empirically set to 0.04, 0.01, 0.2, and 0.0005, respectively.

Network Architecture. ABUV uses a PPO-based model to learn the mapping from the states to the actions. The network structure is shown in Fig. 7. The input of the network is the states and the output is the actions. The network is composed of two parts: the actor and the critic. The actor predicts the actions, while the critic predicts the QoE reward. The actor is a fully connected neural network with 3 hidden layers, and the critic is also a fully connected neural network with 2 hidden layers. Both the actor and the critic have 256 units in their hidden layers. The output of the actor is the probability of each action, and the output of the critic is the QoE reward. The loss function of the actor is the cross entropy loss, while the loss function of the critic is the mean squared error loss. Compared with other prediction algorithms, such as CNN or rule-based methods, PPO shows better performance in making adaptive decisions, and it is also more efficient for adaptive training.

4.2.2. RL agent training

The goal of RL model training is to maximize the total cumulative rewards for each action:

$$R_i = \sum_{k=i+1}^{N} \gamma^k QoE_k, \tag{2}$$

where the cumulative reward R_i for video chunk i is the sum of immediate rewards QoE_k obtained by the agent for each chunk k that can be gained in the future, discounted by a factor γ between 0 and 1 that measures the discounting of future rewards. The maximum chunk number of the task is denoted by N.

To guide policy updates and improve the agent's decision-making ability, advantage estimation is a critical part of RL model training. In this work, we adopt the Generalized Advantage Estimation (GAE) method proposed by Schulman et al. [38]. The GAE method introduces a hyperparameter λ, which controls the trade-off between bias and variance in the advantage estimation. The advantage function $\hat{A}(s, a)$ is defined as follows:

$$\hat{A}(s, a) = QoE_i - V_\phi(s) + \gamma\lambda\hat{A}(s', a'), \tag{3}$$

where QoE_i represents the immediate reward obtained by the agent for chunk i as mentioned in Eq. (1), λ is the GAE parameter, and s' and a' represent the next state and action in the trajectory, respectively. By incorporating the GAE parameter λ into the advantage estimation, we are able to control the bias–variance trade-off and fine-tune the advantage estimation according to the specific requirements of our RL model training.

The loss function used by ABUV combines the policy loss and the value loss in a PPO-like structure. The policy loss is defined as:

$$L(s, a, \theta', \theta) = \min\left(\frac{\pi_\theta(a, s)}{\pi_{\theta'}(a, s)}\hat{A}(s, a), g(\epsilon, \hat{A}(s, a))\right), \tag{4}$$

where $L(s, a, \theta', \theta)$ is the loss function of the current state s, selected action a, and the parameters of the current policy network θ and the old policy network θ'. $\pi_\theta(a, s)$ refers to the probability of selecting action a in state s according to the current policy network θ, while $\hat{A}(s, a)$ is the advantage estimate of action a in state s according to the old policy network θ'. $g(\epsilon, A)$ is a clipped version of the advantage function, where:

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A, & if\ A \le 0, \\ (1-\epsilon)A, & if\ A < 0. \end{cases} \tag{5}$$

To ensure that policy updates are balanced and not excessively conservative or drastic, it is common practice to introduce a small

positive value, denoted as ϵ, during the clipping process. This value is typically set to 0.1 and plays a crucial role in controlling the magnitude of policy updates. By incorporating this smooth scaling mechanism, the algorithm achieves improved stability and convergence while maintaining the flexibility needed for effective policy updates. The RL model training process employed in ABUV is included in Appendix A.

Adaptive retraining. For mobile users, the real-world network traces can vary greatly from the predefined training environment, which may cause the agent to fail drastically. To tackle this issue, ABUV implements adaptive retraining, enabling the agent to become user-specific. ABUV continually collects the video streaming data of each user, including immediate rewards, rebuffer time, and estimated bandwidth (throughput). When the system's performance significantly declines, the adaptive online retraining process is initiated. Specifically, a counter records the number of consecutive chunks for which the immediate reward is negative. When the counter reaches a threshold of δ, which we set to 3 in our experiments, the online retraining process is activated. During online retraining, the agent uses the real-world network trace collected over the last 100 video chunks to update its policy network. The new policy network is then utilized to select the next action. After the retraining process, the counter is reset to 0.

5. Implementation

We primarily implement ABUV using Python 3.10 on the server side, with TensorFlow 2.11.0 [39] for the ABPN-light model and PPO-based RL model, running on Ubuntu 18.04. On the client side, we implement the SR module using C++ and Java within the Android platform, using the TensorFlow Lite C++ API [40].

We make modifications to the ABPN repository [41] to run SR on Android platforms. Specifically, we implement our ABPN-light model with an input image patch size of 64 × 64 and a batch size of 16. As discussed in Section 4.1.2, we pretrain three raw SR models with different upsample factors using the DIV2K dataset once in an offline manner. To fine-tune the models for content-aware upsampling, we train them using different resolutions of the target video, with the training data sizes outlined in Appendix B.1. Finally, we use TensorFlow Lite Converter [39] to convert the trained models to TensorFlow Lite format.

In the offline-training phase, the RL agent is trained using a simulator environment where the network trace is generated based on predefined rules. The energy and time consumption of upsampling are collected through on-device testing with the fine-tuned super-resolution models. To balance exploration and exploitation, we set the discount factor γ to 0.96, and both the actor and critic learning rates are set to 0.0003. We run 1000 training episodes, with each episode comprising 200 chunks. The batch size is set to 64, and the model is trained for 10 epochs.

6. Evaluation

We evaluate ABUV by answering the following questions:

- How well does ABUV perform in terms of QoE compared to other baselines?
- What is the training cost of ABUV, including the pretraining and fine-tuning for both super-resolution and reinforcement learning models?
- Can adaptive retraining help improve the performance in new network environments?

6.1. Methodology

Videos. We source popular YouTube videos from six categories: Games (C1), Music (C2), Education (C3), News (C4), Sports (C5), Entertainment (C6). Within each category, we carefully selected four standard videos (horizontal orientation) and four shorts (vertical orientation) based on their view counts, ranging from 46K to 140.8M views. Standard videos have a maximum duration of 10 min, while shorts are limited to 60 s. We download source videos encoded in H.264 with a 1440p (2K) resolution and re-encode them following YouTube's recommendations [42]. For RL training, we use 75% of the original videos as the training set, with the remaining videos for testing. For SR models, the training and validation set consist of low-resolution and high-resolution videos, as shown in Appendix B.1.

Network Traces. We employ updated public network traces from U.S. broadband (2020) [43] to simulate mobile environments. We filter data for bandwidth, resulting in a downloading throughput range of 0.23 Mbps to 41.11 Mbps. We generate 400 network traces following Pensieve's approach [15], allocating the initial 300 traces for training and reserving the remaining 100 traces for testing.

Hardware. We use the Xiaomi 10 device, powered by the Snapdragon 865, as our mobile platform. We rely on the AI benchmark [44] and Google's ExoPlayer [45] to gather speed and energy cost measurements for mobile video super-resolution. For SR and RL model training, we utilize two GTX 3090 GPUs.

Action space. We consider upsample factors 1, 2, 3, 4 and bitrates 600, 1200, 2500, 4000, 8000, 12,000 kbps for each video chunk. We set a maximum resolution limit of 2K for the target video to ensure efficient resource utilization and avoid unnecessary delays or energy costs. We have a total of 15 actions, as illustrated in Fig. 11. To simulate the transfer of SR models, we initially set the upsample factor to 1 for the first 3 chunks due to the small size of the SR models, as discussed in Section 4.1.

Baselines. We compare ABUV with state-of-the-art baselines, including the buffer-based approach BOLA [17], the learning-based ABR approach GreenABR [16], and the SR-integrated-codec solution BiSR [27]. NEMO [8] is not included in the comparison as it exclusively supports videos encoded in VP9 format.

QoE metrics. When comparing the RL model, which is trained to maximize the QoE score described in Eq. (1), it would be unfair to directly utilize it for comparison purposes. Therefore, we adopt the perceptual QoE model created by the GreenABR team [16]. This model is built upon the Waterloo Streaming QoE Database III (SQoE-III) [46] and defined as follows:

$$\begin{aligned} QoE = {} & \alpha * \sum_{i=1}^{n} q_i - \beta * \sum_{i=1}^{n} p_i - \gamma * \sum_{i=1}^{n} E_i \\ & - \sigma * \sum_{i=2}^{n} (|q_i - q_{i-1}|) - \mu * \sum_{i=2}^{n} \lfloor \frac{|q_i - q_{i-1}|}{20} \rfloor, \end{aligned} \tag{6}$$

where q_i, p_i, E_i represent video quality (VMAF), rebuffer penalty, energy cost just like in Eq. (1). The parameters α, β, γ, σ, μ are set to 0.0771, 1.2497, 2.8776, 0.0494 and 1.4365, respectively to align with the QoE model in baselines [16,27]. This QoE model is used to evaluate the performance of all the baselines and ABUV.

6.2. ABUV vs. Baselines

QoE improvement. ABUV aims to enhance the QoE for mobile users in low-bandwidth environments through efficient SR. To evaluate its performance, we compare ABUV with state-of-the-art baselines, including BOLA, BiSR, and GreenABR, using real videos.

Fig. 8 presents the average QoE results, where BOLA is used as the baseline, and the results are normalized. ABUV outperforms all other baselines in all tested categories. However, the performance of ABUV is sensitive to the video content, with frequent scene changes leading to a reduction in super-resolution performance. For instance, in

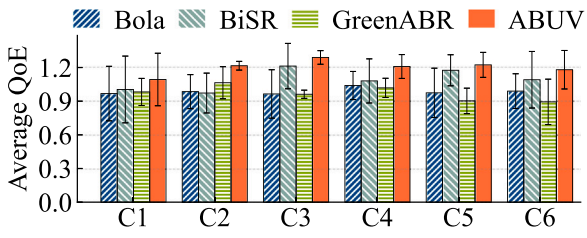


Fig. 8. QoE comparison with baselines.

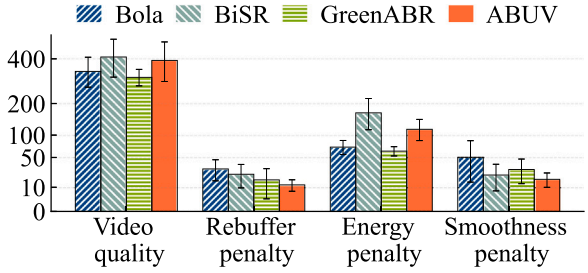


Fig. 9. QoE breakdown analysis.

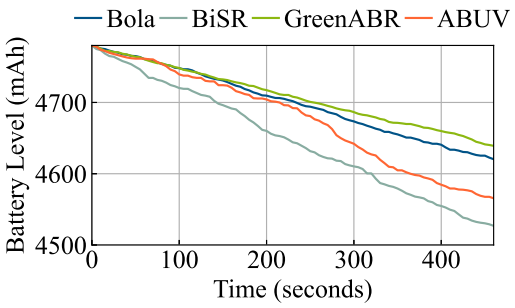


Fig. 10. Battery consumption trace.

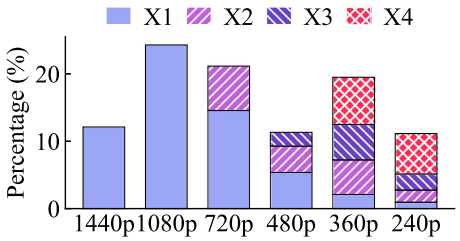


Fig. 11. Bitrate and upsample decisions.

category C3 (Education), which often includes consecutive still frames like slides or whiteboards, ABUV achieves the highest QoE gain of 32.1%. On the other hand, in category C1 (Games), which has complex and fast-changing scenes, ABUV still achieves a significant 12.7% QoE improvement.

As per Eq. (6), QoE comprises four factors: video quality, rebuffer penalty, energy cost, and smoothness penalty. We analyze the contribution of each factor to the overall QoE improvement achieved by ABUV, and present the results in Fig. 9. Our findings suggest that ABUV's QoE gain is mainly contributed by the increase in video quality and decrease in rebuffer and smoothness penalty. However, ABUV increases energy penalty by 28.9% on average, as it requires super-resolution on the client-side. Despite this trade-off, we conclude that the QoE gain of ABUV is significantly higher than the energy cost, hence the trade-off is acceptable. Notably, the calculation of QoE is unbiased as we do not use the QoE model for training ABUV directly, but the QoE as defined in Eq. (6), which has been adopted and verified by the GreenABR team [16].

Bandwidth savings. Limited bandwidth is a major challenge that prevents mobile users from enjoying high quality videos. Despite the requirement of downloading three super-resolution models for each video, ABUV consumes much less bit than other video delivery systems, as mentioned in Section 4.1.2. ABUV saves approximately 59.2% data consumption compared to BOLA and 32.6% compared to GreenABR. Although ABUV selects low bitrate like other ABR algorithms when the bandwidth is limited, it can avoid video quality degradation through super-resolution. By integrating SR into ABR, ABUV can save bandwidth and improve QoE simultaneously.

Energy-efficient upsampling. As depicted in Fig. 10, ABUV, when compared to GreenABR and BOLA, does introduce an increase in energy consumption due to super-resolution processing. However, in contrast to BiSR which runs SR more aggressively, ABUV's adaptive upsample decisions enable more efficient energy usage. By dynamically adjusting bitrate and applying super-resolution judiciously, ABUV effectively balances improved video quality and reduced bandwidth consumption while keeping energy consumption in check. This characteristic makes ABUV a more energy-efficient choice compared to SR-integrated-codec methods like BiSR. Further insights into the video streaming process are detailed in Appendix B.3.

6.3. Improvement with adaptive retraining

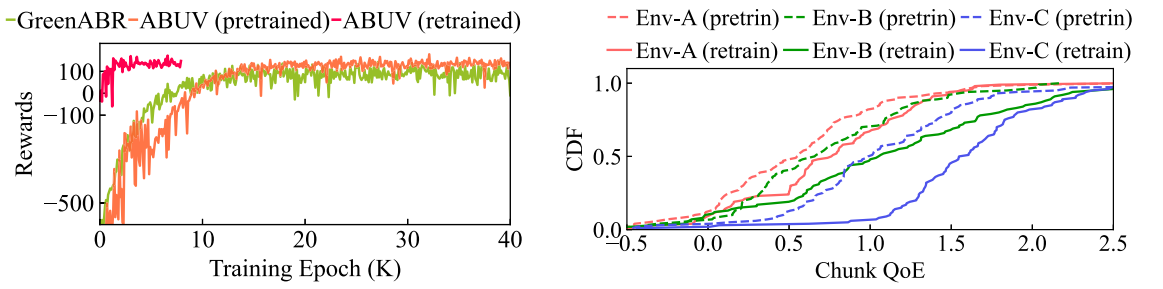
The training expenses for ABUV encompass both pretraining and retraining, with the pretraining cost specifics provided in Appendix B.2. Mobile network environments are inherently unstable, and real-world network traces frequently deviate substantially from predefined ones. Consequently, the RL agent must undergo retraining whenever the network environment shifts to maintain optimal performance. Fig. 12(a) demonstrates the influence of a new network environment on the agent's decision-making process, where catastrophic decisions (reward < 0) may transpire.

To simulate changes in the network environment, we select three network distributions with identical IP addresses based on a publicly available dataset [43]. These distributions are as follows: 0.3 Mbps to 2 Mbps (Env-A), 1.2 Mbps to 20.4 Mbps (Env-B), and 13.8 Mbps to 41.1 Mbps (Env-C). To address performance degradation, ABUV utilizes historical network traces to retrain the RL agent offline. Fig. 12(b) depicts the cumulative distribution function (CDF) of the QoE for both the pretrained and retrained RL agent in different network environments.

For the pretrained RL agent, the QoE can be negative in Env-A, as the agent does not learn how to make correct decisions with such limited bandwidth. However, the retrained RL agent is more likely to achieve not only positive but also higher QoE in new environments. This indicates that adaptive retraining is necessary for the RL agent to adapt to a new network environment.

7. Conclusion

This paper introduces ABUV, an innovative video streaming system designed specifically for mobile devices. ABUV leverages a lightweight super-resolution model called ABPN-light, which is optimized to operate efficiently on mobile platforms. Additionally, ABUV incorporates a PPO-based RL agent that enables adaptive bitrate and upsample factor decisions. The RL agent can be retrained to adapt to changing network environments. To assess the effectiveness of ABUV, we perform rigorous experiments using real-world network traces. The results unequivocally indicate that ABUV delivers a remarkable enhancement in QoE, with improvements ranging from 12.7% to 27.3%. What is more, ABUV demonstrates substantial savings in network usage, reaching up to



(a) Training iteration comparison

(b) CDF of QoE with pretrain and retrain RL in different environments

Fig. 12. Effects of adaptive retraining.

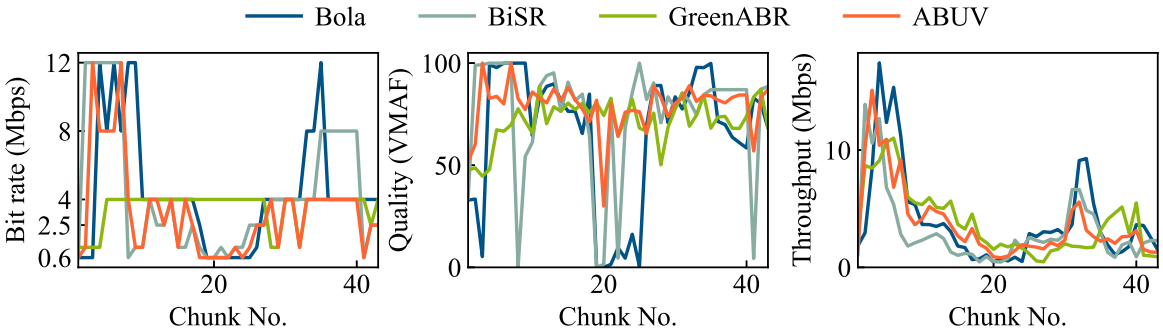


Fig. 13. Detail trace of video streaming.

an impressive 59.2% reduction compared to existing state-of-the-art video streaming systems. These outcomes unequivocally demonstrate the immense potential of ABUV in greatly augmenting the user experience of video streaming on mobile devices, especially in scenarios characterized by demanding network conditions.

CRediT authorship contribution statement

Yichen Lu: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Ji Qi:** Data curation, Conceptualization. **Sheng Zhang:** Writing – review & editing, Methodology, Conceptualization. **Gangyi Luo:** Data curation, Conceptualization. **Andong Zhu:** Writing – review & editing, Investigation. **Jie Wu:** Writing – review & editing. **Zhuzhong Qian:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by Nanjing University-China Mobile Communications Group Co.,Ltd. Joint Institute, Nanjing Key S&T Special Projects (202309006), and Collaborative Innovation Center of Novel Software Technology and Industrialization. Sheng Zhang is the corresponding author.

Appendix A. PPO-based RL model training

Algorithm 1 outlines the RL model training process employed in ABUV, which utilizes a memory buffer to store the trajectories and samples mini-batches from it to update the policy and value function networks. Batch-level updates are adopted instead of step-level updates to reduce the variance of gradient estimates and improve the training stability.

Algorithm 1 ABUV's RL Training

Initialize policy network $\pi_\theta(a, s)$ and value function network $V_\phi(s)$ with random weights;
Initialize memory buffer M to store trajectories;
for each training episode **do**
  Initialize or reset the video streaming environment;
  for each video chunk **do**
    Sample an action a_i from current policy network;
    Store the state p_i, action a_i, reward r_i (Equation (1)) and new state s_{i+1} in memory M;
  end for
  for each mini-batch update **do**
    Sample mini-batch D from memory buffer M;
    Calculate the rewards-to-go R (Equation (2)) and advantage estimates $\hat{A}$ (Equation (3));
    Update policy network:
    $\theta \leftarrow \underset{\theta}{\operatorname{argmax}} \frac{1}{|D|} \sum_{(s,a,r,s')\in D} L(s, a, \theta', \theta)$
    Update value function network:
    $\phi \leftarrow \underset{\phi}{\operatorname{argmin}} \frac{1}{|D|} \sum_{(s,a,r,s')\in D} \left(V_\phi(s) - R\right)^2$
  end for
end for

Appendix B. Experimental details

B.1. Training sizes for SR models

To fine-tune the models for content-aware upsampling, we train them using different resolutions of the target video, with the training data sizes outlined in Table B.1. We exclude 480p and 1440p from training the X3 model since scaling up a 480p (854 × 480) video by

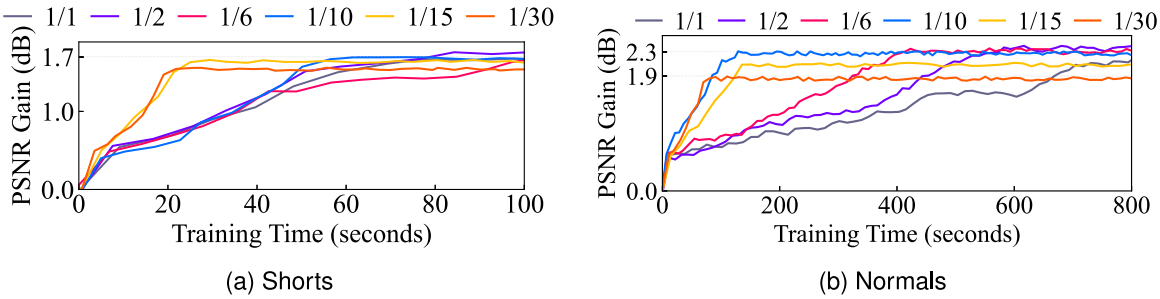


Fig. B.14. Training time with different intervals.

Table B.1
Resolutions of videos for model fine-tuning.

Upsample factor	Input size	Output size
X2	720p: 1280 × 720	1440p: 2560 × 1440
X3	360p: 640 × 360	1080p: 1920 × 1080
X4	360p: 640 × 360	1440p: 2560 × 1440

three times does not correspond to the resolution of a 2K (2560 × 1440) video, and neither clipping nor resizing can fix this pixel discrepancy.

B.2. Training cost on server-side

We analyze the training cost of ABUV's server-side components, including both SR model training and RL agent training, as shown in Fig. B.14.

B.3. Video streaming traces

Fig. 13 illustrates the video streaming process by chunks. As a rule-based ABR algorithm, BOLA can sometimes be overly conservative, leading to a low bitrate selection. GreenABR exhibits a similar trend, as it is trained to conserve energy. In contrast, BiSR and ABUV intelligently utilize SR to enhance video quality. However, BiSR does not account for the energy cost of super-resolution, resulting in the upsampling of low-definition chunks even when bandwidth is sufficient. ABUV, on the other hand, achieves higher QoE by dynamically balancing download and upsampling times.

B.3.1. SR models training

To train the super-resolution models, ABUV adopts the DIV2K dataset [37] as the base checkpoint for pretraining the ABPN-light model. The DIV2K dataset consists of 800 training images and 100 validation images. This pretraining process is conducted once and typically takes approximately 6 h to complete. For each video, ABUV employs transfer learning to fine-tune the models and enable content-aware neural enhancement. However, for videos longer than 5 min, the total number of frames can exceed 9000. Training the models using all frames can become excessively time-consuming and may lead to overfitting. To address this, ABUV adopts a strategy where one frame is selected for every 10 frames in short videos (<60 s) and 15 frames for normal videos (<10 min). This approach effectively reduces the training cost while maintaining reasonable training times and mitigating the risk of overfitting. Fig. B.14 illustrates the reason why we chose 10 and 15 as intervals for horizontal and vertical videos, respectively. We experimented with different frame selection ratios: 1/1, 1/2, 1/6, 1/10, 1/15, and 1/30 for both short and normal videos. Since shorts are shorter, a selection ratio of 1/15 is appropriate to achieve high PSNR gain and fast training speed. However, for normal videos, a selection ratio of 1/15 would result in a loss of about 0.2 dB PSNR gain, so we chose the ratio of 1/10. With the introduction of frame selection, the training process takes less than 40 s for short videos and less than 180 s for long videos.

B.3.2. RL agent training

The RL agent training process consists of two steps, as shown in Fig. 2. First, the RL agent is trained offline using predefined network environments and video samples. Then, when degradation occurs during online video streaming, the RL agent undergoes retraining. Fig. 12(a) compares the cumulative rewards of GreenABR and ABUV during training. For GreenABR and the pretraining stage of ABUV, we use the same network trace (0.4 Mbps∼32.0 Mbps). During the pretraining stage, ABUV is more likely to experience reward degradation due to its more complex action space than GreenABR, leading to slower convergence. However, during retraining, we choose a network trace with a different bandwidth distribution (2.3 Mbps∼4 Mbps), and ABUV converges in less than 10,000 epochs since it does not need to start from scratch.

Data availability

Data will be made available on request.

References

[1] DataReportal, Digital 2023: Local country headlines report, 2023, https://datareportal.com/reports/digital-2023-local-country-headlines.
[2] data.ai., State of mobile 2023, 2023, https://www.data.ai/en/go/state-of-mobile-2023/.
[3] Irina Kegishyan, Mobile video statistics, 2023, https://www.yansmedia.com/blog/mobile-video-statistics.
[4] Kelvin C.K. Chan, Shangchen Zhou, Xiangyu Xu, Chen Change Loy, Investigating tradeoffs in real-world video super-resolution, 2022, pp. 5962–5971, 2022.
[5] Kelvin C.K. Chan, Shangchen Zhou, Xiangyu Xu, Chen Change Loy, BasicVSR++: Improving video super-resolution with enhanced propagation and alignment, 2022, pp. 5972–5981, 2022.
[6] Xintao Wang, Liangbin Xie, Chao Dong, Ying Shan, Real-esrgan: Training real-world blind super-resolution with pure synthetic data, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 1905–1914.
[7] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, Dongsu Han, Neural adaptive content-aware internet video delivery, in: Operating Systems Design and Implementation 2018, 2018.
[8] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, Dongsu Han, Nemo: enabling neural-enhanced video streaming on commodity mobile devices, 2020, pp. 1–14, 2020.
[9] Yinjie Zhang, Yuanxing Zhang, Yi Wu, Yu Tao, Kaigui Bian, Pan Zhou, Lingyang Song, Hu Tuo, Improving quality of experience by adaptive video streaming with super-resolution, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020a, pp. 1957–1966.
[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347 2017.
[11] Iraj Sodagar, The mpeg-dash standard for multimedia streaming over the internet, IEEE Multimed. 18 (2011) (2011) 62–67, 4.
[12] Roger Pantos, William May, HTTP Live Streaming, Technical Report, 2017.
[13] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, Hui Zhang, Oboe: Auto-tuning video ABR algorithms to network conditions, in: Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, 2018, pp. 44–58.
[14] Junchen Jiang, Vyas Sekar, Hui Zhang, Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE, in: Conference on Emerging Network Experiment and Technology 2012, 2012.

[15] Hongzi Mao, Ravi Netravali, Mohammad Alizadeh, Neural adaptive video streaming with pensieve, in: ACM Special Interest Group on Data Communication 2017, 2017.
[16] Bekir Oguzhan Turkkan, Ting Dai, Adithya Raman, Tevfik Kosar, Changyou Chen, Muhammed Fatih Bulut, Jaroslaw Zola, Daby Sow, GreenABR: energy-aware adaptive bitrate streaming with deep reinforcement learning, 2022, pp. 150–163, 2022.
[17] Kevin Spiteri, Rahul Urgaonkar, Ramesh K. Sitaraman, BOLA: Near-optimal bitrate adaptation for misc videos, 2016.
[18] Kevin Spiteri, Ramesh Sitaraman, Daniel Sparacio, From theory to practice: Improving bitrate adaptation in the DASH reference player, ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) 15 (2019) (2019) 1–29, 2s.
[19] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, Bruno Sinopoli, 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, 2015, pp. 325–338.
[20] Pattathal V. Arun, Krishna Mohan Buddhiraju, Alok Porwal, Jocelyn Chanussot, CNN-based super-resolution of hyperspectral images, IEEE Trans. Geosci. Remote Sens. 58 (2020) (2020) 6106–6121, 9.
[21] Rushi Lan, Long Sun, Zhenbing Liu, Huimin Lu, Cheng Pang, Xiaonan Luo, MADNet: a fast and lightweight network for single-image super resolution, IEEE Trans. Cybern. 51 (2020) (2020) 1443–1453, 3.
[22] Juhyoung Lee, Jinsu Lee, Hoi-Jun Yoo, SRNPU: An energy-efficient CNN-based super-resolution processor with tile-based selective super-resolution in mobile devices, IEEE J. Emerg. Sel. Top. Circuits Syst. 10 (2020) (2020) 320–334, 3.
[23] Xining Zhu, Lin Zhang, Lijun Zhang, Xiao Liu, Ying Shen, Shengjie Zhao, GAN-based image super-resolution with a novel quality loss, Math. Probl. Eng. 2020 (2020) (2020) 1–12.
[24] Tairan Liu, Kevin De Haan, Yair Rivenson, Zhensong Wei, Xin Zeng, Yibo Zhang, Aydogan Ozcan, Deep learning-based super-resolution in coherent imaging systems, Sci. Rep. 9 (2019) (2019) 1–13, 1.
[25] Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, Yueting Chen, Srdiff: Single image super-resolution with diffusion probabilistic models, Neurocomputing 479 (2022) (2022) 47–59.
[26] Royson Lee, Stylianos I. Venieris, Łukasz Dudziak, Sourav Bhattacharya, Nicholas D. Lane, Mobisr: Efficient on-device super-resolution through heterogeneous mobile processors, in: Computer Vision and Pattern Recognition 2019, 2019, arXiv.
[27] Qian Yu, Qing Li, Rui He, Gareth Tyson, Wanxin Shi, Jianhui Lv, Zhenhui Yuan, Peng Zhang, Yulong Lan, Zhicheng Li, BiSR: Bidirectionally optimized super-resolution for mobile video streaming, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 3121–3131.
[28] J.M.K. Freeskates, First time at the skatepark, 2022, https://www.youtube.com/shorts/1L423DSEebI.
[29] Zongcai Du, Jie Liu, Jie Tang, Gangshan Wu, Anchor-based plain net for mobile image super-resolution, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2494–2502.
[30] Google, Monitor the battery level and charging state, 2023, https://developer.android.com/training/monitoring-device-state/battery-monitoring.
[31] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, William T. Freeman, Video enhancement with task-oriented flow, Int. J. Comput. Vis. 127 (2019) (2019) 1106–1125, 8.
[32] Netflix Technology Blog., Toward a practical perceptual video quality metric, 2016.
[33] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, Lei Zhang, Toward real-world single image super-resolution: A new benchmark and a new model, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3086–3095.
[34] Mustafa Ayazoglu, Extremely lightweight quantization robust real-time single-image super resolution for mobile devices, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2472–2479.
[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
[36] Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson, How transferable are features in deep neural networks, Adv. Neural Inf. Process. Syst. 27 (2014) (2014).
[37] Eirikur Agustsson, Radu Timofte, Ntire 2017 challenge on single image super-resolution: Dataset and study, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 126–135.
[38] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, Pieter Abbeel, High-dimensional continuous control using generalized advantage estimation, 2015, arXiv preprint arXiv:1506.02438 2015.
[39] J.J. Allaire, Dirk Eddelbuettel, Nick Golding, Yuan Tang, TensorFlow: Large-scale machine learning on heterogeneous systems, 2022, https://tensorflow.org.
[40] TensorFlow Lite C++ API reference, 2023, https://www.tensorflow.org/lite/api_docs/cc.
[41] Zongcai Du, Jie Liu, Jie Tang, Gangshan Wu, Github: SR mobile quantization, 2021, https://github.com/NJU-Jet/SR_Mobile_Quantization.
[42] Youtube Help, YouTube recommended upload encoding settings, 2023, https://support.google.com/youtube/answer/1722171.
[43] U.S. broadband, Measuring broadband America mobile data, 2022, https://www.fcc.gov/reports-research/reports/measuring-broadband-america/measuring-broadband-america-mobile-data.
[44] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, Luc Van Gool, Ai benchmark: Running deep neural networks on android smartphones, in: Proceedings of the European Conference on Computer Vision Workshops, 2018.
[45] Google, ExoPlayer, 2023, https://exoplayer.dev/.
[46] Zhengfang Duanmu, Abdul Rehman, Zhou Wang, A quality-of-experience database for adaptive video streaming, IEEE Trans. Broadcast. 64 (2018) (2018) 474–487, 2.

Yichen Lu is currently pursuing his graduate studies at Nanjing University, where his research interests lie at the intersection of artificial intelligence and mobile video streaming technology. He has dedicated his studies to improving the user experience of mobile video streaming through cutting-edge techniques such as super-resolution algorithms and reinforcement learning (RL).

His work aims to significantly enhance the efficiency and quality of video streaming on mobile platforms, addressing the challenges posed by limited bandwidth and varying network conditions. Lu's commitment to innovation is reflected in his contributions to the field, where he seeks to push the boundaries of video technology for mobile devices.