OmniLive: Super-Resolution Enhanced 360° Video Live Streaming for Mobile Devices

Seonghoon Park*, Yeonwoo Cho*, Hyungchol Jun, Jeho Lee, and Hojung Cha†
Department of Computer Science, Yonsei University, Seoul, Republic of Korea
{park.s, yeonwoo.cho, hyungchol.jun, jeholee, hjcha}@yonsei.ac.kr

ABSTRACT

The live streaming of omnidirectional video (ODV) on mobile devices demands considerable network resources; thus, current mobile networks are incapable of providing users with high-quality ODV equivalent to conventional flat videos. We observe that mobile devices, in fact, underutilize graphics processing units (GPUs) while processing ODVs; hence, we envisage an opportunity exists in exploiting video super-resolution (VSR) for improved ODV quality. However, the device-specific discrepancy in GPU capability and dynamic behavior of GPU frequency in mobile devices create a challenge in providing VSR-enhanced ODV streaming. In this paper, we propose OmniLive, an on-device VSR system for mobile ODV live streaming. OmniLive addresses the dynamicity of GPU capability with an anytime inference-based VSR technique called Omni SR. For Omni SR, we design a VSR deep neural network (DNN) model with multiple exits and an inference scheduler that decides on the exit of the model at runtime. OmniLive also solves the performance heterogeneity of mobile GPUs using the Omni neural architecture search (NAS) scheme. Omni NAS finds an appropriate DNN model for each mobile device with Omni SR-specific neural architecture search techniques. We implemented OmniLive as a fully functioning system encompassing a streaming server and Android application. The experiment results show that our anytime VSR model provides four times upscaled videos while saving up to 57.15% of inference time compared with the previous super-resolution model showing the lowest inference time on mobile devices. Moreover, OmniLive can maintain 30 frames per second while fully utilizing GPUs on various mobile devices.

CCS CONCEPTS

• Human-centered computing → Ubiquitous and mobile computing systems and tools; • Computing methodologies → Neural networks

* Co-primary authors with equal contribution.
† Corresponding author.

KEYWORDS

Mobile omnidirectional video live streaming; On-device video super-resolution; Multi-exit deep neural network; Neural architecture search

ACM Reference format:
Seonghoon Park, Yeonwoo Cho, Hyungchol Jun, Jeho Lee, and Hojung Cha. 2023. OmniLive: Super-Resolution Enhanced 360° Video Live Streaming for Mobile Devices. In *The 21st Annual International Conference on Mobile Systems, Applications and Services (MobiSys '23), June 18–22, 2023, Helsinki, Finland.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3581791.3596851

1 INTRODUCTION

Omnidirectional videos (ODVs), also known as 360-degree or panoramic videos, deliver spherical views around omnidirectional cameras. Various video platforms, such as YouTube [1] and Facebook [2], provide ODV live streaming services for immersive and interactive viewing experiences. In particular, *mobile* ODV live streaming, which delivers live panoramic videos from mobile sources to mobile devices, has drawn attention because of the nomadic nature of both content generation and viewing activities. Mobile ODV live streaming can easily be integrated into extended reality (XR)-based interactive applications. For instance, an XR live tour can be provided, as illustrated in Figure 1, where users immersively watch a streamer's activities or trip experiences using their virtual reality (VR) headsets.

Unfortunately, mobile ODV live streaming faces inherent difficulties because of the bandwidth restrictions of mobile networks. End users are simply unable to enjoy high-quality streaming of ODVs that would normally be experienced with conventional flat videos. Only a small fraction of the entire ODV is played in the viewport of a mobile device, while conventional videos show the entire part. Therefore, to provide the same resolution in a viewport, ODVs require significantly more data than flat videos. For instance, when the viewport resolution is 1920×1080, the overall resolution of ODVs is 7680×3840 (3840s). Note that commercial omnidirectional cameras are currently unable to support 3840s ODVs for live streaming. In addition, the bandwidth of current mobile networks is insufficient to support 3840s ODVs. Although 1080p flat videos are casually streamed, ODV streaming cannot be provided with the same video quality.

One possible solution for providing enhanced ODV quality is to upscale the videos on mobile devices with super-resolution. The super-resolution techniques are based on deep neural network

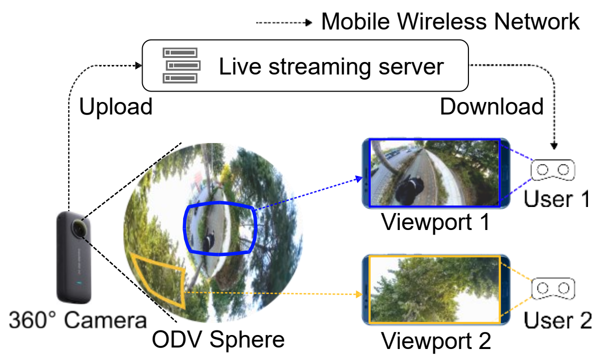


Figure 1. Mobile ODV streaming-based XR live tour.

(DNN) models and require heavy graphics processing unit (GPU) resources. Our preliminary experiments show that mobile devices barely utilize GPU resources while rendering ODVs. This indicates that there is an opportunity to exploit video super-resolution (VSR) for mobile ODV live streaming. Several studies, such as PARSEC [3] and SR360 [4], have presented super-resolution-based techniques to improve the quality of ODV streaming. However, the DNN models of these schemes require a heavy training phase with every incoming ODV, so the streaming servers take a long time to prepare streamable ODVs for incoming videos from the cameras. Therefore, these solutions are not adequate for *live* streaming, where low latency in video delivery is critical.

In this paper, we provide an on-device video super-resolution system for mobile ODV live streaming. For the low latency of video delivery, super-resolution models should be able to upscale ODVs at runtime. In addition, the system should exploit GPU resources as much as possible to maximize the quality of super-resolution while keeping up with the frame rates of the ODVs. Two key issues exist in developing such a system. The first is the dynamicity of GPU frequency in mobile devices. The GPU frequencies of mobile devices change dynamically under various runtime conditions, such as thermal throttling. The second issue is the varying capabilities of GPU resources among devices; that is, each device has a different computing capability by itself. Because of these problems, the inference time of DNN models is not predictable; thus, frame dropping or insufficient GPU utilization could happen during super-resolution operation in the device.

We propose OmniLive, which is an attempt to employ on-device super-resolution for mobile ODV live streaming. To address the dynamicity of GPU frequency, OmniLive introduces the Omni SR technique, which exploits the anytime inference of VSR. Anytime inference [5] is a technique that uses the intermediate outcomes of DNN models as the final results without passing through the entire layers. For Omni SR, we propose a lightweight anytime inference-based DNN model for VSR, called OmniSRNet, and a scheduler, called Omni SR Scheduler, to determine when to stop model inference. To address the discrepancy in GPU computing capability, we present a neural architecture search (NAS)-based Omni NAS. NAS [6] is a technique for finding the optimal neural architecture for the given

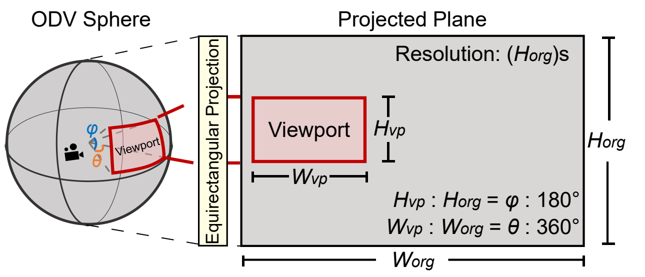


Figure 2. Entire sphere and viewport of ODV.

conditions. To date, no NAS approach has been proposed for anytime inference-aware VSR models. Omni NAS presents an automatic yet effective scheme for finding an optimal OmniSRNet architecture for each mobile device. We have implemented OmniLive as a complete system encompassing a streaming server and an Android application running on smartphones. The implemented system upscales low-resolution (LR) videos four times without degrading the frame rates during ODV live streaming on mobile devices.

The key contributions of this paper are as follows:

- We address the dynamicity of GPU frequency by designing an anytime inference-aware VSR DNN model and accompanying scheduler for early exit of the model.
- We address the discrepancy in GPU computing capability of mobile devices by exploiting a NAS method to find anytime inference-aware VSR models optimized for each device.
- To the best of our knowledge, the fully implemented OmniLive is the first working system that provides super-resolution enhanced videos for mobile ODV live streaming.

2 MOTIVATION AND CHALLENGES

We discuss the motivation for super-resolution techniques for mobile ODV live streaming. The technical challenges for upscaling ODVs on mobile devices are then discussed.

2.1 Motivation

2.1.1 *Problems with Mobile ODV Live Streaming*

Omnidirectional videos are spherical videos containing views in every direction around the cameras. Figure 2 illustrates how a spherical ODV is displayed in a viewport on a mobile device. Although the viewport in conventional videos displays the entire video, the viewport in ODVs shows only a small portion of the videos. Thus, when delivering the same number of pixels in a viewport, streaming ODVs requires significant network traffic compared with streaming flat videos.

We have analyzed how much of an entire ODV is displayed on a viewport and how high the resolution of ODVs should be to provide the same video quality in a viewport as flat videos. As shown in Figure 2, a spherical ODV is converted into a flat video via equirectangular projection (ERP). The equation below describes the relationship between the entire ODV and viewport, here according to fields of view (FoV):

Table 1. Network traffic for ODVs (30 FPS).

	1080s	**1920s**	**3840s**
Overall resolution	2160×1080	3840×1920	7680×3840
Viewport resolution	540×300	960×540	1920×1080
Traffic (Mbps)	8–15	25–47	100–188 (Estimated)

$$W_{vp} = \frac{\theta}{360°} \cdot W_{org},\ H_{vp} = \frac{\varphi}{180°} \cdot H_{org},$$

where W_{org} and H_{org} are the width and height of the entire ODV, respectively. W_{vp} and H_{vp} are the width and height of the viewport, respectively. θ and φ are the horizontal and vertical FoV, respectively.

The specific FoV values depend on applications and devices. To obtain the FoV values of general ODVs on mobile devices, we examined the cases with the FoV test pattern video available on YouTube [7], here for commercial smartphones—Galaxy S20, Pixel 6 Pro, and Galaxy S22. We observed that the FoV values are about 90°–100° and 50°–60°. According to the ERP equation, when θ is 90° and φ is 50°, W_{org} and H_{org} are 4 times W_{vp} and 3.6 times H_{vp}, respectively. This means that, compared with flat videos, ODVs require about 14.4 times more data to provide the same number of pixels in a viewport. That is, to provide the same number of pixels in a viewport as with 1920×1080 (1080p) flat videos, the resolution of ODVs should be 7680×3840 (3840s). However, there are currently no commercial omnidirectional cameras that support this high resolution for live streaming.

We further examined how much network traffic is required for mobile ODV live streaming with a commercial omnidirectional camera, for instance, Insta360 One X2 [8]. Table 1 shows the required traffic when the frame rate is 30. Note that Insta360 One X2 does not support a 3840s resolution; thus, we estimated the required traffic of 3840s ODV with the traffic of the lower-resolution ODVs. According to the Speedtest Global Index [9], the median uplink and downlink bandwidth of global mobile networks are 9.05 and 33.43 Mbps, respectively. As shown in the table, 3840s ODVs require significantly more traffic than the median bandwidth; thus, ODVs are difficult to stream over current mobile networks. In conclusion, mobile ODV live streaming can provide only a relatively low viewport resolution due to limits in network bandwidths.

2.1.2 *Opportunities for Super-Resolution*

One possible approach to overcome low-resolution in the viewport of ODVs is upscaling the video with on-device super-resolution. Super-resolution is a technique that upscales LR images to high-resolution ones using DNN models. VSR tasks were originally processed with image super-resolution (ISR) DNN models. EVSRNet [10], which delivered the shortest inference time in the MAI 2021 challenge [11], is a representative ISR DNN model for on-device super-resolution. Recently, researchers have proposed VSR DNN models that consider interframe information to enhance super-resolution quality. MAI-Diggers [12], the winner of the MAI 2021 challenge, is a representative VSR DNN model for on-device super-resolution. Unfortunately, these DNN models demand a significant amount of GPU computation.

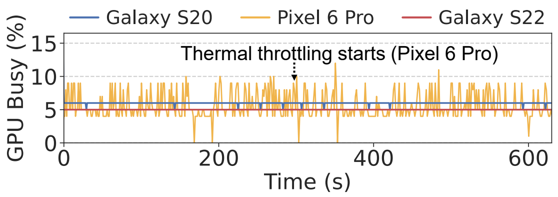


Figure 3. GPU traces while displaying ODVs on mobile devices.

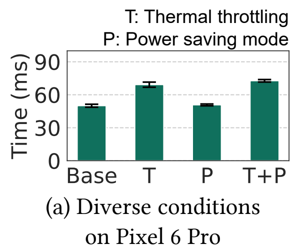
To see the opportunity for super-resolution for mobile ODV live streaming, we conducted an experiment to observe how much GPU resources are used by mobile devices when rendering ODVs. In the experiment, we collected GPU busy percentages while mobile devices played ODVs. Figure 3 shows traces of GPU busy percentage during ODV streaming on Galaxy S20, Pixel 6 Pro, and Galaxy S22 for 10 minutes. Each device utilizes less than 10% of the GPU's computing power when displaying the ODVs. Interestingly, in the case of Pixel 6 Pro, we observed that the traces fluctuate, probably because of vendor-specific implementation of GPU-related metrics. We also imposed thermal throttling on Pixel 6 Pro after 300 seconds in the experiment. Despite thermal throttling, however, the frame rate of the video was not degraded because the GPU busy percentage was low enough. Our observation insight is that there is an opportunity for the super-resolution to improve the quality of mobile ODV live streaming by making the most of available GPU resources.

2.2 Challenges

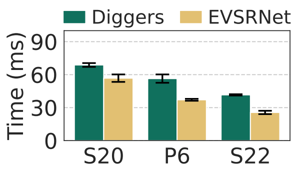
Although there is an opportunity to do so, it is not trivial to develop a system utilizing super-resolution for mobile ODV live streaming. Two key challenges exist: handling the dynamicity of a device's GPU computing power and addressing the discrepancies in computing power among devices.

Dynamicity in a device's computing power. The GPU capability of mobile devices fluctuates depending on various conditions, such as power saving mode or thermal throttling. To see how dynamicity affects super-resolution DNN models' inference time on mobile devices, we tested four conditions (base, thermal throttling, power saving mode, and thermal throttling + power saving mode) on the Pixel 6 Pro smartphone. The DNN model used in the experiment is MAI-Diggers [12]. Figure 4(a) shows the average inference time per frame in each condition when conducting super-resolution tasks of upscaling 540×300 videos to 2160×1200 videos. As shown in the figure, thermal throttling has a large effect on the inference time, while the power saving mode is not. Thermal throttling easily occurs when mobile devices continuously play videos. The increased inference time caused by thermal throttling can lead to frame drops in ODVs.

Discrepancy in computing power among devices. The frame-level inference time of DNN models for VSR diverges depending on the device's hardware specifications. We compared



(a) Diverse conditions on Pixel 6 Pro



(b) Various devices

Figure 4. Inference time of super-resolution models.

the inference time of DNN models among three devices—Galaxy S20, Pixel 6 Pro, and Galaxy S22. For the experiment, we upscaled videos from 540×300 to 2160×1200 resolution with two DNN models—EVSRNet [10] and MAI-Diggers [12]. Figure 4(b) shows the average inference time per frame for each device, indicating huge differences among the devices. This result suggests that if only a single global DNN model is used for super-resolution, the model causes frame drops of ODVs on some devices, while the model insufficiently utilizes GPU resources on other devices.

3 OMNILIVE OVERVIEW

Motivated by preliminary experiments, we propose the OmniLive system, which provides on-device VSR for mobile ODV live streaming. The system overcomes the low-resolution problem caused by mobile network bandwidths. OmniLive aims to provide the same resolution in a viewport as conventional videos, which is done by upscaling ODVs. Particularly, OmniLive focuses on 1080s ODVs whose viewport resolution is 540×300. With a 4x super-resolution, OmniLive upscales 540×300 videos to 2160×1200 videos to provide a similar quality of user experience with 1080p flat videos. Also, OmniLive's super-resolution technique makes the most available GPU resources without causing frame drops of ODVs. OmniLive concentrates on 30 frames per second (FPS) videos; thus, the inference time of the DNN model for super-resolution should not exceed 33.3 milliseconds per frame (ms/f). To achieve this goal, OmniLive solves the challenges discussed in Section 2.2.

OmniLive addresses the dynamicity of the device's computing power by devising an anytime inference-based VSR technique that is named Omni SR. Anytime inference is a technique that allows DNN models to early exit the inference, here depending on the runtime conditions [5]. Different from conventional DNN models that have a single exit, for anytime inference, the DNN model has multiple exits. As far as we are aware, there is no anytime inference-aware DNN model for VSR tasks. We propose a VSR DNN model with multiple exits for anytime inference called OmniSRNet. We also present an anytime inference scheduler called the Omni SR Scheduler. The scheduler decides when the OmniSRNet model should terminate the inference. Section 4 describes Omni SR in detail.

OmniLive also addresses the discrepancy issue in computing power among devices by presenting a NAS method called Omni NAS. NAS [6] is a method that automatically finds the optimal DNN architecture for the given conditions. It is simply not

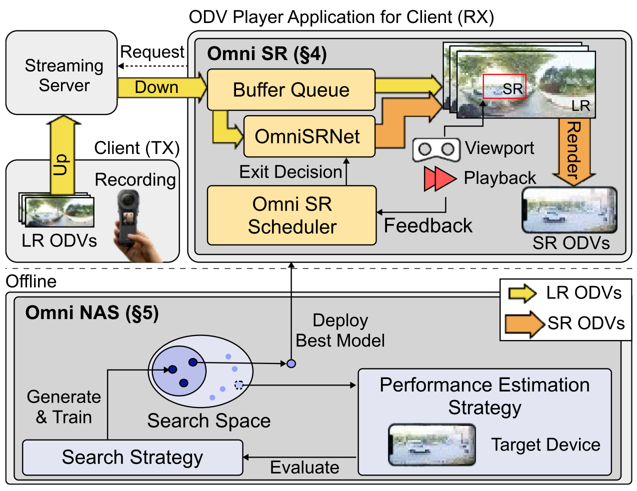


Figure 5. Overview of OmniLive.

possible to manually design the optimal DNN structures for each device whose inference time does not exceed 33 ms/f to maintain 30 FPS. We develop a NAS method by searching for optimal OmniSRNet architectures for each mobile device. Section 5 describes the details of this method.

Figure 5 shows the overall architecture of the OmniLive system. OmniLive is a system covering server streaming ODVs and a mobile device displaying super-resolution enhanced ODVs. First, adequate VSR models (i.e., OmniSRNet variants) are obtained for each device and trained offline with Omni NAS. The obtained VSR models are then embedded into the ODV player application of mobile devices. When streaming ODVs, the server receives LR ODVs from transmitters, i.e., omnidirectional cameras, and transmits them to mobile receivers. Using Omni SR, the ODV player application upscales the streamed LR ODVs to super-resolution enhanced ODVs (SR ODVs).

4 OMNI SR

We propose Omni SR for on-device VSR, which takes the dynamicity of GPU frequencies into account. Omni SR employs an anytime inference strategy to adaptively adjust the DNN inference workload according to varying device conditions. Specifically, we design a new multi-exit DNN model and a runtime scheduler for deciding the exit path of the model. We also present a three-level buffer queue architecture for synchronizing VSR tasks.

4.1 OmniSRNet

OmniSRNet is a video super-resolution model with multiple exit paths, enabling anytime inference for on-device VSR. Unlike ISR tasks, no existing DNN models exploit anytime inference capabilities for VSR tasks. Because ISR tasks take a single frame as the input, existing anytime inference-enabled ISR models [13] rely mainly on simple 2D convolutional neural networks (CNNs). VSR models, on the other hand, take a sequence of frames as the input and aggregate temporal information between consecutive

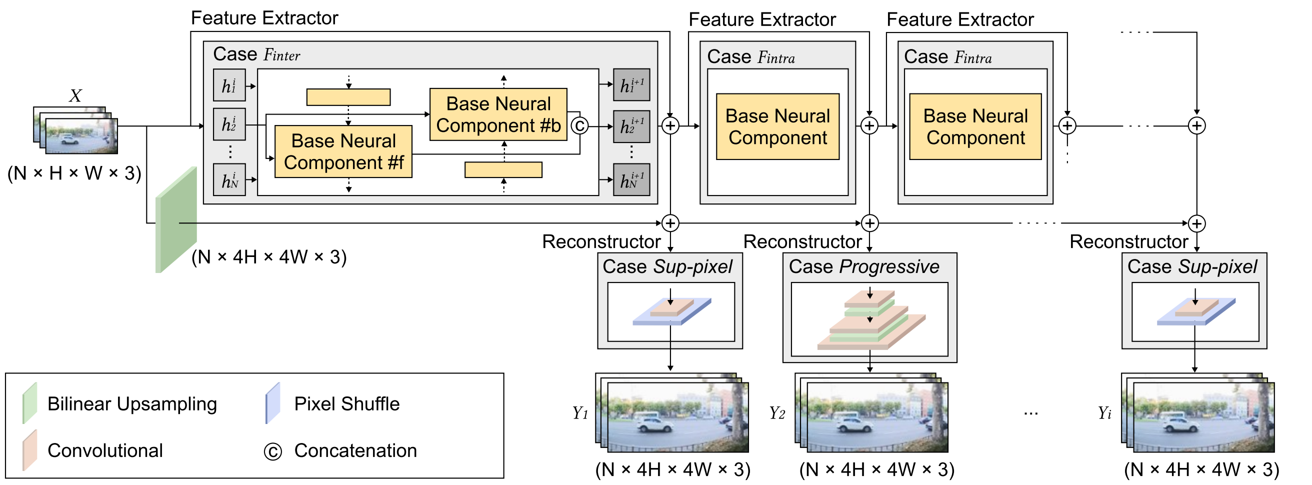


Figure 6. OmniSRNet base architecture.

frames. With these methods, the temporal consistencies from consecutive frames and the upscaling quality are improved compared with the image counterparts. To exploit the anytime inference scheme with the multi-exit VSR model, the proposed OmniSRNet is modularized into feature extractor units and reconstructor units. Figure 6 illustrates the base architecture of OmniSRNet. As shown in the figure, there is an exit for each feature extractor unit, and every exit has its own reconstructor unit because the optimal unit can be different.

Feature extractor unit. Super-resolution tasks usually generate upscaled frames by aggregating blurry and crisp features extracted from the original frames. Thus, the feature maps extracted from shallow feature extractor units must be shared with deeper ones. To this end, we design feature extractor unit F to have skip connections [14], which merge the feature maps extracted from the shallow feature extractors into the feature map extracted from F. Given multiple frames X in the LR video, a feature map H_i generated by the i-th feature extractor unit F_i is computed as follows:

$$H_i = \begin{cases} F_1(X), & if\ i = 1 \\ F_i(H_{i-1}) + H_{i-1}, & if\ i > 1. \end{cases} \tag{1}$$

Let Y_i denote an output of the i-th exit in OmniSRNet, which is an upscaled version of LR X. Y_i is calculated as follows:

$$Y_i = R_i(H_i) + Biup(X), \tag{2}$$

where R_i is a reconstructor unit, and $Biup$ is a bilinear upsampling operation.

A feature extractor unit is divided into two types: interframe feature extractor F_{inter} for gathering the temporal information between frames and intraframe feature extractor F_{intra} for obtaining the image-level information. F_{inter} is a bidirectional recurrent neural network (RNN) that collects forward-propagated features h^f and backward-propagated features h^b from the frame sequence. F_{intra} is a base neural component that extracts the features from image pixels. F_{inter} and F_{intra} are defined as follows:

$$F_{intra} = W_i(x), \;\; F_{inter} = \bigcup_{t=0}^{N} W_t([h_t^f; h_t^b]), \tag{3}$$
$$h_i^f = W_f(x_i, x_{i-1}, h_{i-1}^f), \;\; h_i^b = W_b(x_i, x_{i+1}, h_{i+1}^b),$$

where W represents an operation of the base neural component and x is a single input frame in X in which there are N frames. The base neural component combined with convolution layers and leaky rectified linear units (ReLUs) can be one of three blocks: a residual block [14], a dense block [15], or a variant of information multiple distillation (IMD) blocks [16] Each base neural component optionally has a channel attention module [17]. With these types of neural components, rich linear and nonlinear features can be extracted from the input video, which benefits the super-resolution performance [18].

Reconstructor unit. A reconstructor unit R attached to the end of the feature extractor F is constructed by either the subpixel upscaling operation [19] or the progressive upscaling operation. The former, which consists of a single convolution layer, utilizes a pixel shuffle that reduces the amount of computation by delaying the expansion of the width and height of the input tensors to the end of the operation. The latter, which comprises two or more convolution layers, progressively expands the height and width of input feature maps through the two 2× upscaling operations, which may have more trainable parameters than the former.

4.2 Omni SR Scheduler

The goal of the Omni SR Scheduler is to enhance the quality of LR videos without any frame drops while fully utilizing the mobile GPU. Because of the computational policies in mobile devices, such as thermal throttling, the clock cycles of GPUs continuously fluctuate, which changes the devices' maximum computational capabilities. Unfortunately, depending on the permission policies of the underlying operating systems or hardware, device status information that affects OmniSRNet's inference time, such as the current GPU temperature, is often not exposed to running applications [20]. Thus, instead of inspecting the current

Algorithm 1. Dynamic Anytime Inference Policy

Given Values: OmniSRNet $M = \bigcup_{i=1}^{k}\{F^i, R^i\}$, number of exits k, input frames X, deadline T_d

```
    /* main VSR */
1:  procedure SuperResolute($M, X, T_d$):
2:      $T_{in}$ ← call CopyToInputBuffer($X$)
3:      $T_d \leftarrow T_d - T_{in} - T^*_{out}$
4:      $Y$ ← call InvokeAnytimeModel($M, T_d$)
5:      $T_{out}$ ← call CopyToOutputBuffer($Y$)

    /* anytime inference */
6:  procedure InvokeAnytimeModel($M, T_d$):
7:      for $i = 1$ to $k$ do:
8:          $T_{F^i}$ ← call InvokeNeuralUnit($F^i$)
9:          $T_d \leftarrow T_d - T_{F^i}$
10:         $a_i$ ← call DecideStopOrContinue($i, T_d$)
11:         if $a_i = 0$ then break
12:     $T_{R^i}$ ← call InvokeNeuralUnit($R^i$)
13:     return upscaled frames

    /* decide which exit to use */
14: procedure DecideStopOrContinue($i, T_d$):
15:     if $i = k$ then return 1
16:     $\gamma \leftarrow T_{F^i}/T^0_{F^i}$
17:     $T^*_{F^{i+1}} \leftarrow \gamma \cdot T^0_{F^{i+1}}$, $T^*_{R^{i+1}} \leftarrow \gamma \cdot T^0_{R^{i+1}}$
18:     if $T^*_{F^{i+1}} + T^*_{R^{i+1}} \leq T_d$ then return 1
19:     else return 0
```

computing abilities of the mobile processors, Omni SR Scheduler tracks the historical execution time of each neural component in OmniSRNet to indirectly estimate the varying inference time of the model. In the case of VSR, model inference is performed for all consecutive frames, with a high temporal correlation among them. We effectively utilize these temporal characteristics to build a lightweight scheduler without any additional DNN components.

In addition, Omni SR Scheduler should know the expected quality level of the upscaled frames for each exit, to select the best exit for each input frame. The quality level can be a function of the quality measurements of upscaled frames compared with the original ones, for example, the peak signal-to-noise ratio (PSNR). However, we cannot use this direct method because the quality of the final upscaled video cannot be measured in advance at runtime. Instead, we set integer numbers, from 1 to k, representing the exit numbers in OmniSRNet as the discrete quality levels of the upscaled video. This is based on the assumption that the quality level of upscaled video and the number of executed network layers, that is, the number of exits until the end of OmniSRNet inference, have a positive linear relationship.

Algorithm 1 shows the pseudocodes of the scheduler. The SuperResolute is the main procedure for handling VSR tasks (lines 1–5). Given the LR video, the procedure takes the input X containing N frames of the video, the deadline T_d, and the given OmniSRNet M. The deadline T_d is the remaining time until the upscaled version of the current X is played in the client's viewport, which is managed in video decoders. The OmniSRNet M with k exits contains k sets of the feature extractor F and reconstructor R. Let T_i denote the time taken to obtain the i-th exit's output, which is the upscaled input frames Y. T_i is calculated as follows:

$$T_i = \sum_{t=1}^{N}(T_{R^i} + \sum_{j=1}^{i} T_{F^j}) + T_{in} + T_{out}, \tag{4}$$

where T_{F^j} is the execution time of the j-th feature extractor, T_{R^i} is the execution time of the i-th reconstructor, and T_{in} and T_{out} are the times taken to store frames in the input and output buffers, respectively (lines 2 and 5). To play the output Y without any frame drops, T_i should not exceed T_d (i.e., $T_i \leq T_d$), representing the time constraint of OmniSRNet execution. Unfortunately, since the actual T_{out} is calculated after the execution of OmniSRNet, the algorithm estimates T^*_{out} with a few previous T_{out} values (line 3). Then, InvokeAnytimeModel handles the OmniSRNet execution with given T_d (line 4).

While InvokeAnytimeModel is executed (lines 6–13), the algorithm decides whether to stop or continue the execution of OmniSRNet, that is, where to exit, to meet the time constraint while maximizing the quality of Y. All T_{F^i} and T_{R^i} are measured and recorded (lines 8 and 12), and the decision is made by DecideStopOrContinue (lines 10–11) with the recorded values. While in the process of anytime inference, the inference times of the next feature extractor unit $T_{F^{i+1}}$ and reconstructor unit $T_{R^{i+1}}$ need to be predicted to decide whether to stop at the current exit. Assuming that the number of operations of each neural block is fixed, the inference time of the next neural block can be estimated via the ratio γ of the last inference time to the initial inference time of the current neural block (lines 16–17). Note that the scheduler measures the inference time of all the feature extractor and reconstructor units before streaming starts. $T^0_{F^i}$ and $T^0_{R^i}$ denote the initial inference time, and $T^*_{F^i}$ and $T^*_{R^i}$ the estimated inference time. The algorithm stops at the current exit if the estimated inference time of the neural blocks in the next exit exceeds the deadline T_d (line 18). This decision-making process is performed at every executed exit, and the total time complexity is $O(k)$, where k is the number of total exits in a given OmniSRNet M. The computational cost of the process is about 5 to 10 µs, which is negligible.

4.3 Frame Synchronization

Because Omni SR upscales only the viewport area of ODV frames with OmniSRNet, it is important for Omni SR to synchronize the time between the upscaled and non-upscaled areas. Omni SR solves this problem by introducing a three-level buffer queue architecture, which has an additional queue compared with the case for general media codecs [21]. Figure 7 shows the overall flow of frame synchronization based on the three-level buffer queue. Incoming ODV frames are first enqueued into the low queue and then forwarded to the mid queue by Omni SR as soon as the queue is empty. Omni SR forwards a set of frames—not a single frame—

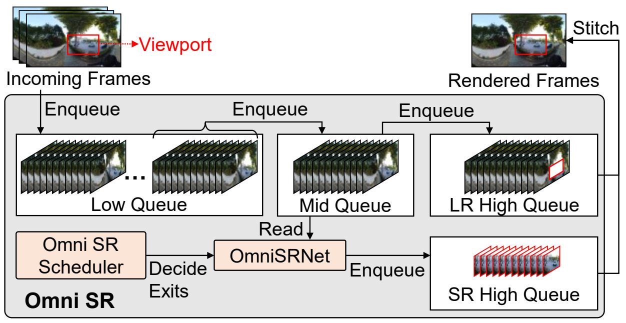


Figure 7. Omni SR's three-level buffer queue.

at one time because OmniSRNet requires multiple frames for interframe information. For the frames in the mid queue, OmniSRNet reads only the viewport regions of the frames and upscales the regions. Next, the original and upscaled frames are transmitted to the LR and SR high queues, respectively. The two high queues are then synchronized in a time scale, and the frames are merged for rendering.

5 OMNI NAS

We propose Omni NAS, which is a scheme that finds the optimal OmniSRNet in mobile devices. The problem of determining the optimal architecture among different anytime inference-aware VSR models is challenging because there are too many possible candidates. Appending multiple exits into DNN models for anytime inference simply increases the number of cases. VSR models take multiple frames as an input, unlike ISR models, which leads to considerably increased complexity for NAS. Omni NAS finds an optimal model for a given device. The scheme consists of three components: search space, search strategy, and performance estimation strategy. Omni NAS sets the appropriate search space for the given task, sets up an appropriate strategy for a given goal, searches efficiently within a search space, and estimates the performance of the model. Based on the search space, search strategy, and performance estimation strategy, Omni NAS finds and trains the OmniSRNet models offline and tests them on real devices with a benchmark application.

Search space. Table 2 shows the search space template of Omni NAS, listing all the configurable parameters of each module within the OmniSRNet architecture. The developer can set the search space of Omni NAS using the template to obtain an optimized OmniSRNet for the target device. There are various options for searching, such as the types of neural components, number of channels, and number of units. Considering all the options, possible configurations are in the magnitude of 60 trillion, making it impossible to conduct a complete search. Thus, an efficient search strategy must be devised.

Search strategy. Because the search space of Omni NAS is vast, we must set an efficient search strategy. We base our strategy on the aging evolution search method [22], which repeats generation, transformation, and search because our search space is nondifferentiable. The aging evolution search is well known when it comes to exploring the nondifferentiable search space

Table 2. Template for Omni NAS search space.

Degree of Freedom	Options
(1) Neural component type	Residual, IMD, Dense
• Channel attention module	Yes, No
◦ Number of reduction channels	{1, 2, 3, ..., 8}
• Bottleneck input	Yes, No
◦ Number of bottleneck channels	{8, 12, 16, ..., 32}
• Depth of convolution layers	{1, 2, 3}, if Residual
	{2, 3, 4}, if IMD
	{2, 3, ..., 6}, if Dense
(2) Number of feature extractors*	{2, 3, 4, 5}
• Feature extractor type	Bidirectional RNN, 2D CNN
• Number of block channels	{4, 6, 8, ..., 20}
(3) Number of reconstructor units*	{2, 3, 4, 5}
• Reconstructor unit type	Subpixel, Progressive
• Number of block channels	{4, 6, 8, ..., 20}
• Number of upscaling channels	{4, 6, 8, ..., 20}

* The numbers of feature extractor units, reconstructor units, and exits are the same for each model.

efficiently. Omni NAS allows for the exploration of a broader search space at a neural component level to reduce the complexity of the search. Because OmniSRNet is an anytime inference model, it is updated based on the loss computed between the ground truth video and super-resolution videos that are given from each exit. The goal is to produce higher quality videos, that is, higher PSNR scores, for the latter exits of the model. In particular, the search algorithm tries to make the exit number and the PSNR score have a positive linear relationship, as discussed in Section 4.2. As such, we define the loss function $\mathcal{L}$ as the weighted sum of each loss at i-th exit $\mathcal{L}_i$:

$$\mathcal{L} = \sum_{i=1}^{k} exp(10, \frac{\Delta PSNR}{10k} \cdot i) \cdot \mathcal{L}_i, \tag{5}$$

Performance estimation strategy. We present a performance estimation strategy for scoring models searched by Omni NAS. When Omni NAS completes the training of one architecture, the performance is estimated based on the quality of super-resolution using PSNR and structural similarity index measure (SSIM) metrics. Once estimated, the model is transformed into a target device model to measure the model size and estimate the runtime. Finally, we consider a model with a greater number of exits as a better one because the model would provide more flexible configurations of the OmniLive system. Based on the assumptions above, Omni NAS finds an architecture α^*, which is one of the possible transformations for the given architecture α, as follows:

$$\begin{aligned} \alpha^* &= argmax_{\alpha \in \mathcal{A}} \mathcal{S}(\alpha) \\ &= argmax_{\alpha \in \mathcal{A}} \{PSNR_{avg}(\alpha), SSIM_{avg}(\alpha), \\ &\quad -ModelSize_{max}(\alpha), N(Exit(\alpha)), \\ &\quad -Runtime_{min}(\alpha), |T_d - Runtime_{max}(\alpha)|\}. \end{aligned} \tag{6}$$

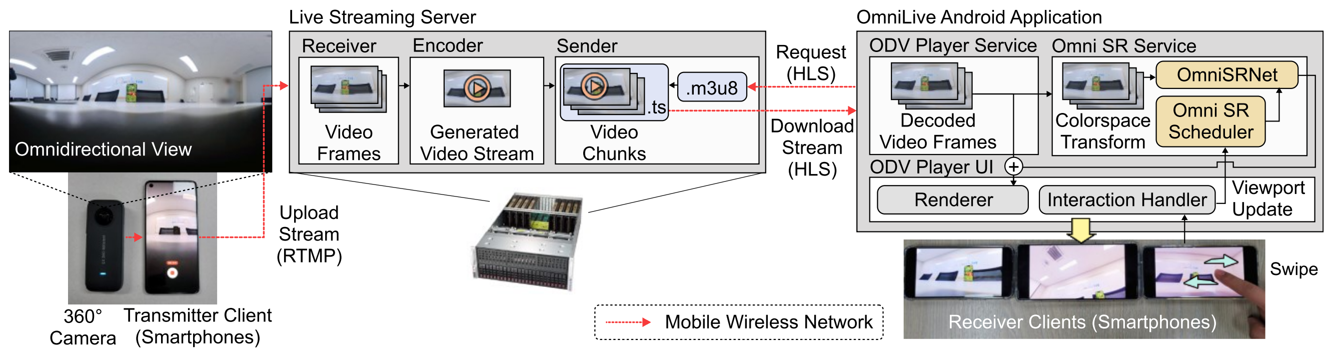


Figure 8. OmniLive implementation.

All the input parameters are normalized in order to make the variables comparable. Also, we applied random weights to the variables, which is a widely-used technique for NAS to avoid local optimum. The thermal throttling issue is also considered during performance estimation. This way, Omni NAS generates the device-specific and thermal throttling-aware OmniSRNet models.

6 IMPLEMENTATION

We implemented OmniLive as an end-to-end system for mobile ODV live streaming. OmniLive consists of three components: Omni NAS, which finds the appropriate OmniSRNet variant for a given device; a streaming server for mobile ODV live streaming; and the OmniLive Android application for runtime super-resolution in the receiving device. Figure 8 shows the flow of mobile ODV live streaming using the OmniLive implementation.

Omni NAS. Omni NAS implementation consists of one central server for the core NAS logic and multiple training servers for the parallel training of OmniSRNet candidates. We implemented the central server with TensorFlow [23]. The server generates OmniSRNet candidates with the Omni NAS search strategy, forwards the candidates to the training servers, and receives the trained candidates from the training servers. Then, the central server transforms the candidates into TensorFlow Lite [24] models, which are inserted into Android applications before scoring the models on mobile devices using the performance estimation strategy. Based on the model scores, the central server produces the next candidates.

Training servers are implemented with PyTorch [25]. Upon receiving OmniSRNet candidates, the servers train the candidate models using video datasets and return the trained models to the central server. For OmniSRNet training, AdamW [26] is used to optimize the hyperparameters. Epoch, batch size, learning rate, and learning rate decay per five epochs are set to 30, 4, 4e-3, and 0.5, respectively. The *ΔPSNR* for weight in each exit is set to 1.5. We trained the models using video data with 240×150 pixels per 10 frames randomly cropped from the REDS train dataset [27]. Because the REDS dataset is smaller than the other datasets, such as Vimeo90K [28], we use data augmentation methods such as flip, mirror, reverse, color permutation, and blend.

Live streaming server. We implemented the streaming server using FFmpeg [29]. The server comprises three parts: receiver, encoder, and sender. The receiver in the streaming server receives ODVs' frame data from the transmitter, that is, omnidirectional cameras, via the real-time messaging protocol (RTMP). Upon receiving the frame data, the encoder in the streaming server generates video transport stream (TS) files as downloadable video chunks using the H.264 video codec. Then, the streaming server delivers the video chunks to the receiver clients, that is, smartphones, with the HTTP live streaming (HLS) protocol.

OmniLive Android application. The OmniLive Android application featuring Omni SR is implemented using Java-based Android API and C++-based Android NDK v25. The detailed implementation falls into three parts: the ODV player service, ODV player user interface, and Omni SR service. First, the ODV player service handles the HTTP connection between the device and the streaming server, receiving video chunks from the server via HLS. Upon receiving the chunks, the service decodes the ODV frames in YUV 420 color space. Second, the Omni SR service is developed based on TensorFlow Lite. The service reads the viewport area of the decoded frames and transforms the area into RGB color space using NEON Intrinsics [30]. After this, Omni SR upscales the area with OmniSRNet according to the process discussed in Section 4.3. For the Omni SR Scheduler, the multi-exit of anytime inference is implemented using TensorFlow Lite C++ Subgraph API. Third, the ODV player user interface renders ODV frames using OpenGL, a YUV420 shader for the nonupscaled area, and RGB shader for the upscaled area of the frames. The user interface also handles user interactions—the viewport area of ODVs is changed according to the user's swipe gestures.

7 EVALUATION

We evaluated OmniLive in two ways. First, we conducted a benchmark test to measure the performance of the OmniSRNet models found by Omni NAS. Next, we investigated the efficacy of Omni SR on real mobile devices. Table 3 lists the three mobile devices used in the evaluation.

7.1 Adequacy of OmniSRNet

We designed a benchmark test to evaluate the OmniSRNet models which Omni NAS found for various devices. Using the test, we compared OmniSRNet with state-of-the-art super-resolution

Table 4. Performance comparisons with mobile real-time super-resolution models on Galaxy S22. ✓ meets the requirement of 30 FPS while ✗ does not.

		Interpolation	2D CNN		Bi-RNN	OmniSRNet (Galaxy S22)	
		Bicubic	ESPCN [19]	EVSRNet [10]	Diggers [12]	Exit 1	Exit 5
PSNR/ SSIM	REDS [27]	26.14/0.7292	27.27/0.7565	27.01/0.7483	27.96/0.7812	27.00/0.7483	27.75/0.7735
	Vid4 [32]	22.38/0.6095	22.89/0.6397	22.91/0.6440	23.65/0.6918	22.94/0.6507	23.46/0.6797
	Vimeo90K [28]	29.74/0.8482	30.18/0.8456	29.65/0.8382	31.99/0.8882	30.51/0.8610	31.70/0.8825
	SPMCS [33]	25.66/0.7242	26.24/0.7468	26.28/0.7528	27.25/0.7886	26.31/0.7549	27.07/0.7818
Model size (KB)		-	151	78	236	17	155
Inference time (ms/f)		-	42.00 ✗	25.60 ✓	41.2 ✗	10.97 ✓	30.52 ✓

Table 3. Device specifications.

	Galaxy S20	Pixel 6 Pro	Galaxy S22
Release	February 2020	October 2021	February 2022
AP	Snapdragon 865	Google Tensor	Snapdragon 8 Gen 1
GPU	Adreno 650 (587 MHz)	Mali-G78 (848 MHz)	Adreno 730 (818 MHz)
Benchmark*	220	341	1295
OS	Android 12	Android 12	Android 12

* The benchmark scores are measured using the AI benchmark application [31]. A higher score indicates faster inference of DNN tasks.

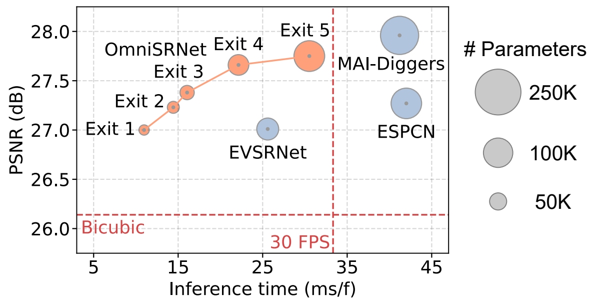


Figure 9. Comparisons of OmniSRNet with state-of-the-art super-resolution models (Galaxy S22, REDS dataset).

DNN models to show that our model outperforms others in terms of inference time and super-resolution quality.

7.1.1 *Experiment Setup*

We selected three state-of-the-art DNN models for super-resolution tasks: ESPCN [19], EVSRNet [10], and MAI-Diggers [12]. ESPCS serves as the base ISR option for FFmpeg [29], and EVSRNet and MAI-Diggers are the models used in Section 2.1. We trained the selected DNN models using the REDS train dataset [27], which is also used for OmniSRNet training. We excluded PARSEC [3] and SR360 [4] from our evaluation because they were unable to upscale videos from 540×300 to 2160×1200 resolution within 33.3 ms/f on any of our target devices. We observed that they took 88.5–568 ms/f to perform the super-resolution task.

We selected four different test datasets widely used for VSR tests to evaluate the chosen models and OmniSRNet: REDS validation [27], Vid4 [32], Vimeo90K-T [28], and SPMCS [33]. All the LR video inputs were preprocessed using the 4x bicubic downsampling method using TensorFlow. We evaluated the models based on super-resolution quality, inference time, and model size. The quality of super-resolution tasks was assessed using PSNR and SSIM in the RGB color space. The inference time was measured using the OmniLive Android application. The super-resolution tasks for the inference time test were upscaling videos from 540×300 to 2160×1200 resolution.

7.1.2 *Results*

Table 4 presents the results of our experiment using the Galaxy S22. Figure 9 visualizes the results using the REDS validation dataset. The super-resolution quality of OmniSRNet Exit 5 was similar to MAI-Diggers, which outputs the best quality among the models, showing slight differences between -0.29 dB and -0.19 dB. OmniSRNet Exit 5 reduced inference time by 25.93% compared to MAI-Diggers; thus, OmniSRNet met the requirement of 30 FPS while MAI-Diggers did not. The overall quality scores of OmniSRNet were higher than those of EVSRNet, which was the only model meeting the 30 FPS constraint among the three state-of-the-art models. Especially, the inference time of OmniSRNet Exit 1 was 57.15% shorter (10.97 ms/f compared to 25.60 ms/f) than that of EVSRNet, while PSNR and SSIM scores were higher on average. To sum up, OmniSRNet was able to achieve considerably faster inference than the previous model showing the lowest-inference time, EVSRNet. Also, OmniSRNet showed the best super-resolution quality among the models that satisfied real-time conditions.

In addition, we investigated the performance of Omni NAS. With Omni NAS, we conducted NAS tasks on all three target devices and selected the best-performing models for each device. We found that the OmniSRNet variant for Galaxy S22 was based on the bidirectional RNN architecture, while the variants for other devices were 2D CNN models. Note that the bidirectional RNN architecture generally shows a better super-resolution quality than the 2D CNN architecture but requires more computing power because of its complexity. This result indicates only the Galaxy S22 could take advantage of the bidirectional RNN architecture because the other devices could not meet the real-time constraint with the complex architecture.

Table 5 shows the PSNR and inference time of the models found by Omni NAS. The PSNR scores in the table were assessed

Table 5. OmniSRNet models' PSNR and inference time per frame on three mobile devices.

	Galaxy S20		Pixel 6 Pro		Galaxy S22	
	PSNR (dB)	Inf. time (ms/f)	PSNR (dB)	Inf. time (ms/f)	PSNR (dB)	Inf. time (ms/f)
Bicubic	26.14	-	26.14	-	26.14	-
Exit 1	26.80	27.72	27.01	18.21	27.00	10.97
Exit 2	27.13	32.64	27.32	23.14	27.23	14.43
Exit 3	27.34	36.81	27.80	28.07	27.38	16.06
Exit 4	-	-	-	-	27.66	22.14
Exit 5	-	-	-	-	27.75	30.52

with the REDS validation dataset. We tried to find OmniSRNet models with three or more exits where all the exits meet the 30 FPS constraint and show as good super-resolution quality as possible. Unfortunately, Omni NAS failed to find an appropriate OmniSRNet variant meeting the 30 FPS constraint for Galaxy S20 because the device has limited computation capability compared to the other devices. Instead, Omni NAS found a variant meeting the 24 FPS constraint for Galaxy S20—an OmniSRNet model with three exits (Exit 1, Exit 2, and Exit 3). On the other hand, Omni NAS found the best-performing models for Pixel 6 Pro and Galaxy S22, where all the exits of the models satisfy the 30 FPS constraint. The model for Pixel 6 Pro had three exits, and the model for Galaxy S22 had five. To sum up, the existing models either sacrificed super-resolution quality or failed to meet the real-time condition as described in Table 4. However, Omni NAS was able to find OmniSRNet variants that provide high super-resolution quality while satisfying the real-time constraint, as shown in Table 5. Also, the OmniSRNet models found by Omni NAS have multiple exits; therefore, the models can process adaptive VSR tasks, which the existing models are unable to handle.

7.2 Efficacy of Omni SR Scheduler

To validate the efficacy of the Omni SR Scheduler, we set up real-world ODV streaming environments. In a live streaming scenario, the optimal operation is to maximize GPU utilization with the real-time constraint under various runtime conditions. We evaluated the performance of OmniLive under dynamic resource conditions, such as thermal throttling.

7.2.1 Experiment Setup

During live streaming, the video content may affect system performance, which may change the experimental results. The experiments for each scenario should be conducted with the same video data for a fair comparison with the baselines. Thus, we prerecorded a 30-minute 2160×1080-resolution ODV with the Insta360 One X2 camera at the lab and used the video for each scenario in our experiments. A desktop server with a Xeon(R) Silver 4214 CPU was used as a streaming server, and the three mobile phones listed in Table 3 were used as receivers. The mobile devices were connected to Wi-Fi networks, and the network capacities were limited to 10 Mbps for uplink and 30 Mbps for downlink, which are typical mobile network conditions, by controlling the Wi-Fi router in the lab. To minimize the effect of charging and temperature issues, the phones were fully charged and cooled down before the experiments. The experiments were conducted in the lab environment of typical room temperature (24 ℃).

With the setup, each device could receive the ODV transmitted from the streaming server and perform at a super-resolution. Specifically, mobile devices converted 540×300 resolution videos in the viewport region to 2160×1200 resolution using the OmniLive Android application with Omni SR. In this scenario, we validated the efficacy of our technique (*OmniLive*) based on a multi-exit scheme by comparing it to three other policies based on a single-exit scheme. The first baseline was selecting the earliest exit (*Exit 1*). The second was selecting the next-to-last exits (*Exit 2* for Galaxy S20 and Pixel 6 Pro and *Exit 4* for Galaxy S22). The third was choosing the last exits (*Exit 3* for Galaxy S20 and Pixel 6 Pro and *Exit 5* for Galaxy S22). The target FPS was 30 for Pixel 6 Pro and Galaxy S22 and 24 for Galaxy S20, which cannot meet the 30 FPS constraint, even with Exit 1.

7.2.2 Results

Figure 10 shows the traces of GPU temperature, GPU frequency, FPS, and inference time per frame for ODV streaming with the three devices. We measured the traces on Galaxy S20 and S22 for 800 seconds and on Pixel 6 Pro for 1600 seconds to see the effect of thermal throttling—the starts of thermal throttling were vendor-specific, and Pixel 6 Pro had a tendency for late thermal throttling. Overall, the GPU temperature quickly rose at first and then dropped back because of thermal throttling. Here, GPU frequency was limited, depending on the GPU temperature. GPU operated at a maximum frequency while the temperature rose but operated at a significantly lower frequency after the temperature started to decrease. In the case of the Pixel 6 Pro, where the GPU frequency fluctuated, the maximum frequency was capped according to thermal throttling.

The FPS traces in the figure show the effectiveness of our technique in resource-constrained situations caused by thermal throttling. In the case of Pixel 6 Pro and Galaxy S22, with the policy of selecting the last exits, the frame rates could not meet the 30 FPS constraint. Although the last exits showed an inference time shorter than 33.3 ms per frame in the benchmark test, as shown in Table 5, the exact inference time exceeded the constraint in real usage scenarios. The next-to-last exit policy showed 30 FPS but notably decreased as thermal throttling occurred and inference time increased. Meanwhile, with OmniLive, 30 FPS was consistently achieved, even after thermal throttling invocation. In the case of the Galaxy S20, 24 FPS was achieved by the policies, except for Exit 2 and Exit 3, after thermal throttling started. This means that OmniLive was able to meet the frame rate constraint by adaptively selecting the DNN exit. We can see that the Exit 1 policy also achieved 30 FPS or 24 FPS. However, the video quality with this policy was lower than our technique because selecting the exit compromises super-resolution performance (see Table 5).

Figure 11 shows the exit numbers' moving average traces with a one-second window for the OmniLive policy. OmniLive selected the exit adaptively, depending on runtime conditions. OmniLive

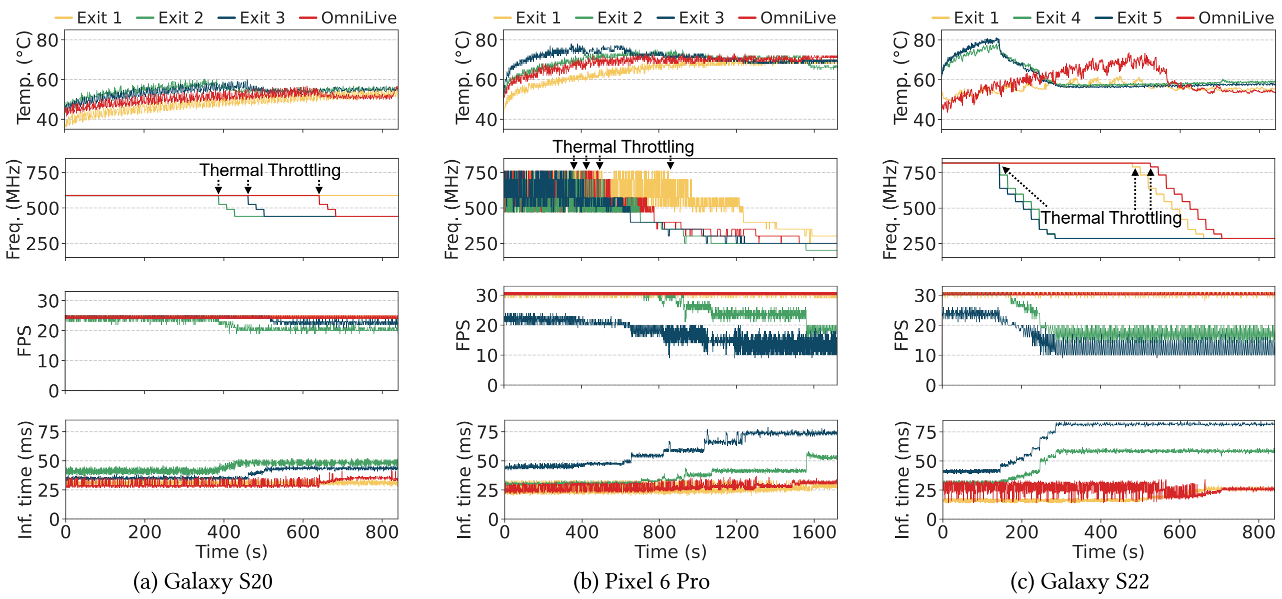


Figure 10. Traces of GPU temperature, GPU frequency, FPS, and inference time per frame.

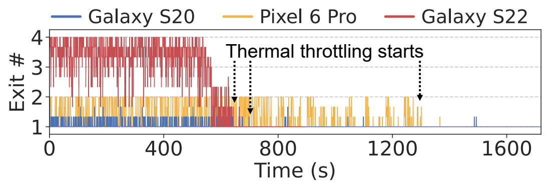


Figure 11. Exit number traces.

mainly selected the latter exits before thermal throttling occurred, whereas Exit 1 was selected after thermal throttling. Such dynamic selection of exit number indicates that OmniLive maximized super-resolution performance by fully utilizing the resources within a 33.3 ms period per frame. As shown in Figure 10, in the case of the Exit 1 policy, super-resolution was performed within 20 ms before thermal throttling occurred. The extra time left (13 ms) shows the potential for improving super-resolution performance with further computation. In conclusion, our experimental results show that OmniLive effectively upscaled ODVs on various devices with on-device VSR techniques. This indicates that OmniLive could overcome the low number of pixels in a viewport problem caused by mobile network limitations.

8 RELATED WORK

We discuss previous work related to OmniLive from four perspectives: DNN models for on-device VSR, multi-exit architecture and NAS, and super-resolution enhanced video streaming.

8.1 DNN Models for On-Device VSR

The DNN models for super-resolution tasks were typically developed for server-level computers, which have rich GPU resources. As the computing ability of mobile devices has improved, several studies have recently been conducted to develop lightweight models for super-resolution tasks running on mobile devices. Previous studies employed lightweight models which were originally developed for ISR tasks due to their simple DNN architecture. For instance, EVSRNet [10], ESPCN [19], and SRCNN [18] introduced 2D CNN-based ISR models. Such ISR-based methods are fundamentally limited in exploiting the temporal information of video, which is a key feature of VSR schemes.

Recent studies have focused more on the VSR DNN models which utilize videos' temporal information, especially based on the basic RNN or bidirectional RNN architecture. Using the additional information, the VSR DNN models generate better quality videos on an equal amount of calculation than the ISR DNN models. RRN [34] first suggested an RNN-based VSR approach. BasicVSR [35] and MAI-Diggers [12] are derived from bidirectional RRN, which has shown the highest super-resolution quality. The VSR DNN models usually exhibit longer inference time than the ISR DNN models, as shown in Table 4; thus, the models fail to meet the 30 FPS constraint even on recent mobile devices. Meanwhile, our OmniSRNet enhances super-resolution quality by adopting the VSR model architecture while meeting the real-time constraint.

8.2 Multi-Exit Architecture and NAS

Most of the DNN models for mobile VSR tasks are based on the single-exit DNN architecture, and therefore they are unable to provide adaptive inference. To overcome the limitation, LarvaNet [13] presented a multi-exit DNN model for super-resolution, but it still has limitations. LarvaNet was developed for ISR tasks, not VSR tasks, leading to low super-resolution quality for videos. Moreover, LarvaNet did not target mobile devices; hence the scheme is difficult to be utilized for mobile ODV live streaming. Meanwhile, recent work for super-resolution has adopted NAS to find optimal DNN models. MobiSR [36] proposed a technique for performing on-device super-resolution with NAS, but the scheme is limited to images and not applicable to videos. EVSRNet [10] also used NAS to obtain the optimal DNN model, but it presented a single-exit ISR DNN model.

To the best of our knowledge, no previous attempts have been made to design a multi-exit VSR DNN model for mobile devices. Also, there has not been a study for NAS finding the optimal multi-exit VSR model. OmniLive is the first attempt to provide adaptive on-device VSR using the multi-exit neural network and NAS.

8.3 Super-Resolution Enhanced Streaming

Researchers have exploited the potential of super-resolution techniques to enhance video quality or reduce network usage of video streaming services, especially focusing on regular videos. NAS [37] and DeepStream [38] introduced video-on-demand (VOD) streaming using on-device super-resolution for desktop computers. SRAVS [39] and NEMO [40] proposed a video streaming technique enabling real-time super-resolution on mobile devices. LiveNAS [41] and NeuroScaler [42] adopted server-side super-resolution for live video streaming. However, these regular video-based solutions are not feasible for ODV live streaming, as ODVs require considerably more computing and network resources than regular videos.

Concerning ODV streaming, attempts have been made to provide DNN-based super-resolution on client devices, but they are limited to VOD services. Sophon [43] presented super-resolution enhanced ODV streaming for desktop computers using a prefetch approach. PARSEC [3] and SR360 [4] proposed mobile-aware solutions with tiling methods, which selectively apply super-resolution to a part of the ODV tile viewed by the user. Unfortunately, these methods require a preprocessing phase, prior to distributing a video, which requires nearly 20 times the length of the given video. Hence, the solutions are inadequate for live streaming where the low latency characteristic is important. In contrast, OmniLive is a suitable technique for *live* streaming because videos are delivered directly to a mobile device without preprocessing on the streaming server, and super-resolution is performed on the fly on mobile devices.

9 DISCUSSION

We discuss several issues on the current design of OmniLive. The first is how OmniLive addresses the heavy training issue of super-resolution DNN models. When it comes to the ODV, previous solutions for on-demand streaming, such as PARSEC [3] and SR360 [4], are not suitable for *live* streaming. The servers of these solutions require high-resolution videos and significant time to generate per-content DNN models. For instance, if a one-minute video is uploaded to the streaming server, the server needs more than 20 minutes to train a DNN model using the contents of the video as high-resolution ground-truth data. For on-demand streaming, this complexity of model training is acceptable, as training can be conducted before the content distribution, and high-resolution videos can be transmitted to the servers. For live streaming, however, near real-time delivery is crucial for the service, and high-resolution videos cannot be uploaded to the servers due to limited uplink bandwidths; therefore, online model training is virtually impossible. Our NAS-based approach, therefore, seeks to find the best-performing *generic* models a priori.

The second issue is transmitting all the ODV pixels in OmniLive, instead of cropping the videos. The ODV streaming services typically transmit the entire ODV (in the ERP format) to the user's device to provide an instant transition of view selection for the users. OmniLive targets this typical format of ODV services. This functionality certainly requires network overhead. One solution to address the network traffic is to transmit specific parts of an ODV to the user's device. For example, VP-Only [44], Flare [45], and PARIMA [46] optimized network transmission by predicting the viewport of end users. The approach is promising, but many challenges exist in determining the size and target of sub-areas upon which the responsiveness of view transition is not hampered. Besides, the approach's effect is limited to the downlink side because streamers must send the entire pixels to the streaming server to cover numerous viewers' various viewports. Note the limited uplink bandwidth is the main constraint of the ODV streaming, as discussed in Section 2.1.1. Our current work does not address this issue, but the issue is worth investigating in future work.

Can this be related to first 360 degree paper we read?

10 CONCLUSION AND FUTURE WORK

To the best of our knowledge, OmniLive is the first attempt to provide adaptive on-device VSR for ODV live streaming. ODV is often downscaled to mitigate network bandwidths, and OmniLive successfully restores video quality with super-resolution at runtime, fully exploiting the available GPU resources. OmniLive addresses challenges by introducing anytime inference-based Omni SR and Omni NAS. Our results have shown that OmniLive maximizes super-resolution quality while maintaining 30 FPS across a wide range of mobile devices. We hope that OmniLive will be usefully integrated with many upcoming XR applications.

OmniLive can be expanded in two ways. First, as mobile devices have adopted various hardware chips for machine learning, such as digital signal processors, tensor processing units, and neural processing units, our GPU-focused solution can be extended to exploit these processors. Second, we can improve OmniLive by partially offloading super-resolution tasks from mobile devices to streaming servers. This is possible because a downlink is relatively unrestricted compared with an uplink in mobile networks.

ACKNOWLEDGEMENTS

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00532, Development of High-Assurance (≥EAL6) Secure Microkernel).

REFERENCES

[1] Encoder settings for Live 360 degree videos - YouTube Help. https://support.google.com/youtube/answer/6396222?hl=en

[2] Bringing Live 360 to everyone. https://www.facebook.com/formedia/blog/bringing-live-360-to-everyone

[3] Mallesham Dasari, Arani Bhattacharya, Santiago Vargas, Pranjal Sahu, Aruna Balasubramanian, and Samir R. Das. 2020. Streaming 360-Degree Videos Using Super-Resolution. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications (INFOCOM '20)*, IEEE, 1977–1986. DOI:https://doi.org/10.1109/INFOCOM41043.2020.9155477

[4] Jiawen Chen, Miao Hu, Zhenxiao Luo, Zelong Wang, and Di Wu. 2020. SR360: Boosting 360-degree video streaming with super-resolution. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '20)*, ACM, New York, NY, USA, 1–6. DOI:https://doi.org/10.1145/3386290.3396929

[5] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D Lane. 2021. Adaptive Inference through Early-Exit Networks: Design, Challenges and Directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning* (EMDL '21), ACM, New York, NY, USA, 1–6. DOI:https://doi.org/10.1145/3469116.3470012

[6] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. *J. Mach. Learn. Res.* 20, 1 (January 2019), 1997–2017.

[7] FOV test pattern YouTube. https://www.youtube.com/watch?v=sCONzcPxJf0

[8] Insta360 ONE X2. https://www.insta360.com/kr/product/insta360-onex2

[9] Speedtest Global Index. https://www.speedtest.net/global-index

[10] Shaoli Liu, Chengjian Zheng, Kaidi Lu, Si Gao, Ning Wang, Bofei Wang, Diankai Zhang, Xiaofeng Zhang, and Tianyu Xu. 2021. EVSRNet: Efficient video super-resolution with neural architecture search. In 2021 *IEEE/CVF Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '21)*, IEEE, 2480–2485. DOI:https://doi.org/10.1109/CVPRW53098.2021.00281

[11] Andrey Ignatov, Andres Romero, Heewon Kim, Radu Timofte, Chiu Man Ho, Zibo Meng, Kyoung Mu Lee, Yuxiang Chen, Yutong Wang, Zeyu Long, Chenhao Wang, Yifei Chen, Boshen Xu, Shuhang Gu, Lixin Duan, Wen Li, Wang Bofei, Zhang Diankai, Zheng Chengjian, Liu Shaoli, Gao Si, Zhang Xiaofeng, Lu Kaidi, Xu Tianyu, Zheng Hui, Xinbo Gao, Xiumei Wang, Jiaming Guo, Xueyi Zhou, Hao Jia, and Youliang Yan. 2021. Real-time video super-resolution on smartphones with deep learning, mobile AI 2021 challenge: Report. In 2021 *IEEE/CVF Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '21)*, IEEE, 2535–2544. DOI:https://doi.org/10.1109/CVPRW53098.2021.00287

[12] MAI-VSR-Diggers. https://github.com/Feynman1999/MAI-VSR-Diggers

[13] Geun Woo Jeon, Jun Ho Choi, Jun Hyuk Kim, and Jong Seok Lee. 2020. LarvaNet: Hierarchical Super-Resolution via Multi-exit Architecture. In *Computer Vision – ECCV 2020 Workshops*, Springer, Berlin, Heidelberg, Germany, 73–86. DOI:https://doi.org/10.1007/978-3-030-67070-2_4

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16)*, IEEE, 770–778. DOI:https://doi.org/10.1109/CVPR.2016.90

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17)*, IEEE, 2261–2269. DOI:https://doi.org/10.1109/CVPR.2017.243

[16] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. 2019. Lightweight Image Super-Resolution with Information Multi-distillation Network. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19)*, ACM, New York, NY, USA, 2024–2032. DOI:https://doi.org/10.1145/3343031

[17] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. 2020. ECA-Net: Efficient channel attention for deep convolutional neural networks. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition *(CVPR '20)*, IEEE, 11531–11539. DOI:https://doi.org/10.1109/CVPR42600.2020.01155

[18] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2016. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 2 (February 2016), 295–307. DOI:https://doi.org/10.1109/TPAMI.2015.2439281

[19] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16)*, IEEE, 1874–1883. DOI:https://doi.org/10.1109/CVPR.2016.207

[20] Android permissions for system developers. https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/permission/Permissions.md

[21] MediaCodec | Android Developers. https://developer.android.com/reference/android/media/MediaCodec

[22] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI '19)*, AAAI Press, 4780–4789. DOI:https://doi.org/10.1609/aaai.v33i01.33014780

[23] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI '16)*, USENIX Association, Savannah, GA, 265–283.

[24] TensorFlow Lite. https://www.tensorflow.org/lite

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury Google, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf Xamla, Edward Yang, Zach Devito, Martin Raison Nabla, Alykhan Tejani, Sasank Chilamkurthy, Qure Ai, Benoit Steiner, Lu Fang Facebook, Junjie Bai Facebook, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS '19)*, Curran Associates Inc., Red Hook, NY, USA. DOI:https://doi.org/10.5555/3454287.3455008

[26] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations (ICLR '19)*. DOI:https://doi.org/10.48550/arxiv.1711.05101

[27] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. 2019. NTIRE 2019 challenge on video deblurring and super-resolution: Dataset and study. In 2019 *IEEE/CVF Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW '19)*, IEEE, 1996–2005. DOI:https://doi.org/10.1109/CVPRW.2019.00251

[28] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T. Freeman. 2019. Video Enhancement with Task-Oriented Flow. *Int. J. Comput. Vis.* 127, 8 (August 2019), 1106–1205. DOI:https://doi.org/10.1007/S11263-018-01144-2

[29] FFmpeg. https://ffmpeg.org/

[30] Neon. https://developer.arm.com/Architectures/Neon

[31] AI-Benchmark. https://ai-benchmark.com/

[32] Ce Liu and Deqing Sun. 2014. On bayesian adaptive video super resolution. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 2 (February 2014), 346–360. DOI:https://doi.org/10.1109/TPAMI.2013.127

[33] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. 2017. Detail-Revealing Deep Video Super-Resolution. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV '17)*, IEEE, 4482–4490. DOI:https://doi.org/10.1109/ICCV.2017.479

[34] Takashi Isobe, Fang Zhu, Xu Jia, and Shengjin Wang. 2020. Revisiting Temporal Modeling for Video Super-resolution. In *The 31st British Machine Vision Conference (BMVC '20)*. DOI:https://doi.org/10.48550/arxiv.2008.05765

[35] Kelvin C.K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. 2021. BasicVSR: The Search for Essential Components in Video Super-Resolution and Beyond. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '21)*, IEEE, 4945–4954. DOI:https://doi.org/10.1109/CVPR46437.2021.00491

[36] Royson Lee, Stylianos I. Venieris, Lukasz Dudziak, Sourav Bhattacharya, and Nicholas D. Lane. 2019. MobiSR: Efficient On-Device Super-Resolution throu

gh Heterogeneous Mobile Processors. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)*, ACM, New York, NY, USA, 1–16. DOI:https://doi.org/10.1145/3300061.3345455

[37] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural Adaptive Content-aware Internet Video Delivery. In *Proceedings of the 13th USENIX conference on Operating Systems Design and Implementation (OSDI '18)*, USENIX Association, Carlsbad, CA, 645–661.

[38] Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. 2022. DeepStream: Video Streaming Enhancements using Compressed Deep Neural Networks. *IEEE Trans. Circuits Syst. Video Technol.* (2022). DOI:https://doi.org/10.1109/TCSVT.2022.3229079

[39] 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 (INFOCOM '20)*, IEEE, 1957–1966. DOI:https://doi.org/10.1109/INFOCOM41043.2020.9155384

[40] 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 (MobiCom '20)*, ACM, New York, NY, USA, 1–14. DOI:https://doi.org/10.1145/3372224.3419185

[41] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. Neural-Enhanced Live Streaming: Improving Live Video Ingest via Online Learning. In *Proceedings of the 2020 Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*, ACM, New York, NY, USA, 107–125. DOI:https://doi.org/10.1145/3387514.3405856

[42] 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 (SIGCOMM '22)*, ACM, New York, NY, USA, 795–811. DOI:https://doi.org/10.1145/3544216.3544218

[43] Jianxin Shi, China Lingjun Pu, China Xinjing Yuan, China Qianyun Gong, China Jingdong Xu, Lingjun Pu, Xinjing Yuan, Qianyun Gong, and Jingdong Xu. 2022. Sophon: Super-Resolution Enhanced 360° Video Streaming with Visual Saliency-aware Prefetch. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22)*, ACM, New York, NY, USA, 3124–3133. DOI:https://doi.org/10.1145/3503161.3547750

[44] Ching Ling Fan, Jean Lee, Wen Chih Lo, Chun Ying Huang, Kuan Ta Chen, and Cheng Hsin Hsu. 2017. Fixation prediction for 360° video streaming in head-mounted virtual reality. In *Proceedings of the 27th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '17)*, ACM, New York, NY, USA, 67–72. DOI:https://doi.org/10.1145/3083165.3083180

[45] Feng Qian, Bo Han, Qingyang Xiao, Vijay Gopalakrishnan, and Vijay 2018 Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18)*, ACM, New York, NY, USA, 99–114. DOI:https://doi.org/10.1145/3241539.3241565

[46] Lovish Chopra, Sarthak Chakraborty, Abhijit Mondal, and Sandip Chakraborty. 2021. PARIMA: Viewport adaptive 360-degree video streaming. In *Proceedings of the Web Conference 2021 (WWW '21)*, ACM, New York, NY, USA, 2379–2391. DOI:https://doi.org/10.1145/3442381.3450070