Article:
Ren, J, Gao, L, Wang, X et al. (5 more authors) (2021) Adaptive Computation Offloading for Mobile Augmented Reality. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 5 (4). 175. ISSN 2474-9567

https://doi.org/10.1145/3508492

Adaptive Computation Offloading for Mobile Augmented Reality

JIE REN, Shaanxi Normal University, China
LING GAO, Northwest University, China
XIAOMING WANG, MIAO MA, GUOYONG QIU, Shaanxi Normal University, China
HAI WANG, JIE ZHENG, Northwest University, China
ZHENG WANG, University of Leeds, United Kingdom

Augmented reality (AR) underpins many emerging mobile applications, but it increasingly requires more computation power for better machine understanding and user experience. While computation offloading promises a solution for high-quality and interactive mobile AR, existing methods work best for high-definition videos but cannot meet the real-time requirement for emerging 4K videos due to the long uploading latency. We introduce Actor, a novel computation-offloading framework for 4K mobile AR. To reduce the uploading latency, Actor dynamically and judiciously downscales the mobile video feed to be sent to the remote server. On the server-side, it leverages image super-resolution technology to scale back the received video so that high-quality object detection, tracking and rendering can be performed on the full 4K resolution. Actor employs machine learning to predict which of the downscaling resolutions and super-resolution configurations should be used, by taking into account the video content, server processing delay, and user expected latency. We evaluate Actor by applying it to over 2,000 4K video clips across two typical WiFi network settings. Extensive experimental results show that Actor consistently and significantly outperforms competitive methods for simultaneously meeting the latency and user-perceived video quality requirements.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**.

Additional Key Words and Phrases: Mobile Augmented reality, Adaptive Computation Offloading, Super-resolution, 4K Video Processing

ACM Reference Format:
Jie Ren, Ling Gao, Xiaoming Wang, Miao Ma, Guoyong Qiu, Hai Wang, Jie Zheng, and Zheng Wang. 2021. Adaptive Computation Offloading for Mobile Augmented Reality. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 37, 4, Article 111 (December 2021), 29 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Augmented reality (AR) is quickly emerging as a key enabling technology for new mobile applications. By overlaying digital content and information onto the physical world, AR promises immersive experience in application areas like information retrieval, e-commercial, entertainment, education, communication and healthcare. Indeed, market research projects that the global market for mobile AR (MAR) will reach $198 billion by 2025 [90].

Authors' addresses: Jie Ren, School of Computer Science, Shaanxi Normal University, Xian, China, renjie@snnu.edu.cn; Ling Gao, School of Information Science & Technology, Northwest University, Xian, China; Xiaoming Wang, Miao Ma, Guoyong Qiu, School of Computer Science, Shaanxi Normal University, Xian, China; Hai Wang, Jie Zheng, School of Information Science & Technology, Northwest University, Xian, China; Zheng Wang, School of Computing, University of Leeds, United Kingdom, z.wang5@leeds.ac.uk.

2474-9567/2021/12-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

Although it is currently possible to perform lightweight AR tasks like simple object tracking and rendering locally on the end-user mobile device [2, 28], next-generation MAR applications are likely to require having the ability to detect and classify more complex objects and perform more complex rendering tasks for better usability and improved user experience [52]. Furthermore, MAR use cases are not limited to local usage. New applications like remote collaboration apps [88, 95] also leverage AR to enhance user interaction experience. Such interactive applications are sensitive to the responsive time. Concurrent to this development is the broader adaptation of high-resolution video standards like the 4K UHD (3840×2160) resolution [23, 92, 108]. We also observe this technology trend in the smartphone industry. For example, the 2K resolution screen becomes ubiquitous on recent smartphones like iPhone 12, Galaxy S21 and XiaoMi 11, with some equipped with a 4K resolution screen [10]. As the hardware specification is improved, application developers will look at ways to utilize the high resolutions provided by the device. Furthermore, research also shows that AR applications with a 4K resolution and high frame rates can provide a more immersive and engaging experience [89, 114]. All these recent hardware and software movements require significantly more computation power and resources than what can be offered by a battery-powered mobile system, imposing a significant challenge on resource-constrained devices.

State-of-the-art methods for object detection and AR rendering typically employ a large deep neural network (DNN) for accurate environmental understanding and high-quality rendering. However, it is notoriously difficult to run large models on mobile devices while providing fast response time [118]. Take the speed-optimized Faster RCNN model [26] on the performance-tuned TensorFlow Lite [96] runtime as an example. It can take over a second to execute Fast RCNN to process a single video frame on a high-end mobile device [52], making it difficult to achieve the typically 30 frames per second (FPS) required for smooth video playback and interactive AR.

A promising solution for achieving real-time MAR is to offload all or some of the MAR computational pipeline to the edge, or cloud [52, 119]. By shifting the computation-intensive tasks to a more powerful remote server, computation offloading offers the potential of interactive and responsive MAR on a wide range of mobile devices. Unfortunately, computation offloading remains challenging due to the stringent requirements on high processing accuracy and low end-to-end latency. To achieve high accuracy, the object detection and tracking system must work on video frames (or images) of adequate resolutions, because the lack of details on a low-resolution image can hinder the ability of machines to classify and locate objects of interests [70]. However, sending a high-resolution video will incur significant uploading delay, which in turn leads to longer processing latency. Prior studies show that even turn-around-time of 100ms can significantly compromise the useability of computation offloading because of changes in the user's view - the frame locations where the object was detected may no longer match its current location [119]. This uploading and processing delay is a particular problem for a typical mobile network where the uplink bandwidth is often much narrower than that of the downlink [86]. As future MAR applications are likely to produce virtual elements in much higher quality, we are left with less latency budget for MAR computation offloading.

Existing work on computation offloading for mobile computer vision has considered a range of techniques. These include trading DNN accuracy for lower latency [48, 69], offloading the entire [16, 45] or part of the object detection task [52, 119] to the cloud. Although these approaches were effective for the traditional high definition (HD) video format with a resolution of up to 1,920 × 1,080, they would require over 300ms offloading latency even using a high-speed 5G network when processing a 4K video, and are thus ill-suited for emerging 4K MAR. Video buffering techniques offer little help as the long buffering delay can seriously violate the interactive constraint of many AR scenarios.

We present ACTOR[1], a novel computation offloading framework for MAR. ACTOR is designed to support 4K video processing by leveraging the recent advancement in super-resolution (SR) [101] for low offloading latency and responsive MAR. ACTOR achieves this by first downscaling the input 4K video feed to a suitable lower-resolution

[1] ACTOR = **A**daptive **C**omputa**T**ion **O**ffloading for Augmented **R**eality.

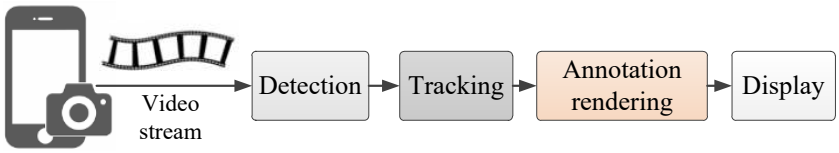


Fig. 1. A representative MAR processing pipeline. Object detection and rendering are often responsible for the bottleneck of the MAR processing pipeline.

video to reduce the uploading time. On the server-side, it employs a highly optimized DNN-based SR model to scale up a few, carefully selected video frames to the 4K resolution and then uses these SR-scaled frames to quickly interpolate the remaining frames to the 4K solution. By doing so, object detection, tracking and image rendering can be performed on full 4K video frames to preserve detection accuracy and rendering quality. The server then encodes the rendered, overlayed digital contents and sends them back to the user device, which will be merged with the original 4K video locally stored on the mobile device memory to be played back to the user. By utilizing the computation capability and resources of a remote server, Actor enables complex, more capable MAR on a wide range of resource-constrained mobile systems.

Translating our high-level idea to develop a practical system is not trivial. A major hurdle is how to preserve the user-perceived experience when downscaling the video feed. The challenge here is that the optimal downscaling resolution depends on the video content, the network environment, and user expectation that can vary from one user to the other. Actor addresses this challenge through *lightweight* predictive models learned *offline* and deployed on the user device. These models are trained to capture subtle interactions between video content, the network environment and user perception of delay and video quality. The models permit Actor to make adaptive scheduling decisions for any new video content *unseen* at design time. Such an approach avoids the pitfalls of using a hard-wired heuristics that require human modification when the computing environment changes.

We have implemented a prototype of Actor[2] and applied it to over 2,000 4K video clips from three public datasets [9, 60, 85]. We test Actor on commodity hardware under two typical WiFi networks and compare it against alternative offloading strategies, including the state-of-the-art MAR offloading method [52]. Experimental results show that Actor consistently outperforms all existing schemes by simultaneously delivering better user-perceived video quality without sacrificing the real-time constraint.

The core contribution of this paper is on combining super-resolution and computation offloading to optimize mobile AR by using machine learning as the enabling technology. This paper makes the following contributions. It is the first to:

- introduce a 4K MAR offloading framework that is built upon SR, which allows the mobile to provide real-time 4K AR service under the limited networking bandwidth and guarantee the high performance of object detection and user-perceived quality (Sec. 4);
- train a generic QoE model off-line at first and employ transfer learning to address the QoE model portability issue across users (Sec. 5.3);
- show how an adaptive MAR offloading framework can be built using predictive modeling (Sec. 5.1);

2 BACKGROUND AND PROBLEM SCOPE

[2]Code and data are available at: *https://github.com/daydreamren/actor.*

2.1 Background

Mobile augmented reality. Figure 1 depicts the typical processing pipeline for MAR. The video stream captured by a mobile camera is fed into an object detection algorithm to find and track interesting objects from each video frame so that annotation rendering can be performed. For most MAR applications, object detection, tracking and image rendering are computationally intensive.

Super resolution (SR). This is a technique for enhancing the resolution of an imaging system. It takes as input a low resolution image and produces a corresponding image with a high resolution. SR is based on a key assumption that much of the high-frequency data of an image is redundant and thus can be accurately reconstructed from low-frequency components [83]. Recent studies have demonstrated that SR builds upon DNNs can generate high-quality, high-resolution images from low-resolution inputs [22, 39, 101]. This work leverages the advances in SR to optimize MAR for latency.

Video multimethod assessment fusion (VMAF). This is a statistical metric to estimate the subjective quality of video streams. This metric has been successfully applied to evaluate the video stream quality from different domains, including live sports, video chats, gaming, 360 videos, and user generated content [50]. A recent study on the 4K VMAF model shows that the VMAF score has a strong correlation with the user's subjective score on 4K video quality [62, 72]. This metric is also used to evaluate the VR quality [66]. In this work, we follow the practice in the recent studies in 4K video and AR optimization by choosing VAMF to quantify the user perception of MAR video quality. In this work, the reference is the MAR video performed on the original, full 4K video stream captured by the mobile camera, and the distorted one, which is the video generated using remote MAR offloading. VMAF takes a value between 0 and 100, higher is better. Prior studies show that a VAMF score over 70 is considered to be good for many users [62].

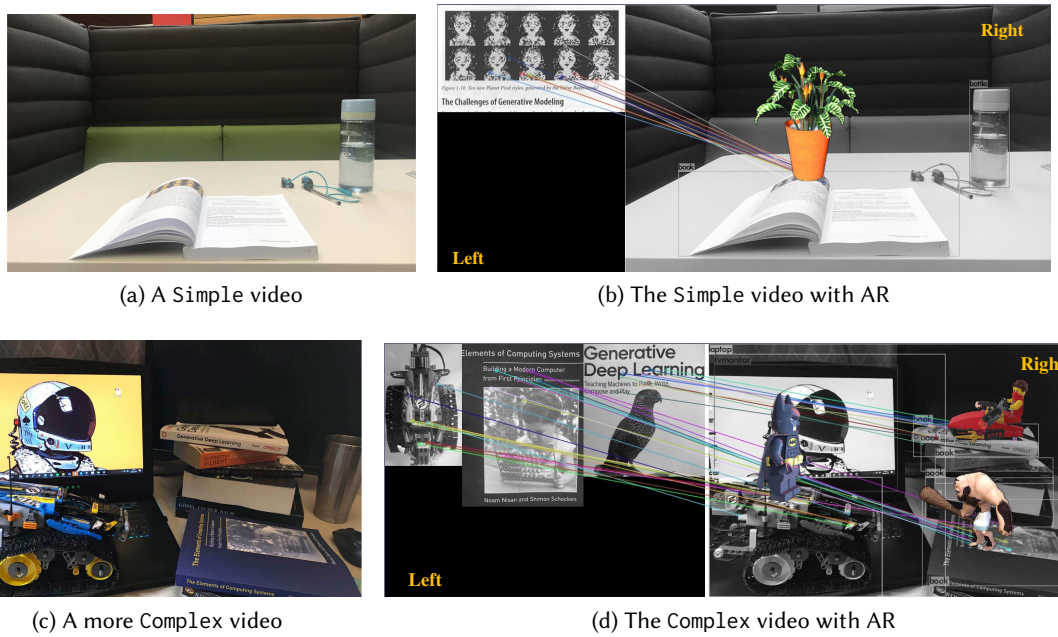
2.2 Problem Scope

Actor leverages the computation capability of a remote server to support high-quality 4K MAR while reducing the processing latency. Offering such capabilities will enable better user experience by supporting complex, computation-intensive MAR on a wide range of mobile devices with a variety of computing capabilities and resources. Actor is designed to trade user-perceived video quality for reduced offloading latency. Our work is motivated by recent studies, which show that a high frame rate is essential for ensuring an immersive and engaging experience for AR application [89, 114]. By reducing the turn-around-time, Actor can then improves the FPS to ensure the user-perceived experience. Furthermore, the improvements of video quality beyond "good enough" often do not enhance mobile user experience [81, 121]. Instead of uploading a full 4K video stream to a remote server, we dynamically and adaptively generate a lower-resolution video with a smaller size. The remote server uses SR to upscale the received video to the full 4K resolution for object detection, tracking and rendering. As a result, macroblocks[3] that contain the overlayed AR contents will be then sent back to the user device to replace the original macroblocks of the raw 4K video frames. Our approach is useful for many networking environments where the uplink bandwidth is typically 2-10x narrower than that of the downlink [86]. While privacy-preserving for computation offloading is not a focus of this work, our approach can be easily integrated with existing video privacy-preserving frameworks [20, 57, 78].

3 MOTIVATION EXAMPLE

As a motivation example, consider performing MAR offloading on the two videos illustrated in Figure 2.

[3]The macroblock is a processing unit in video encoding, which typically consists of 16×16 samples.



(a) A Simple video (b) The Simple video with AR

(c) A more Complex video (d) The Complex video with AR

Fig. 2. Motivation video samples (a & c) and the AR results performed on the video feeds (b & d).

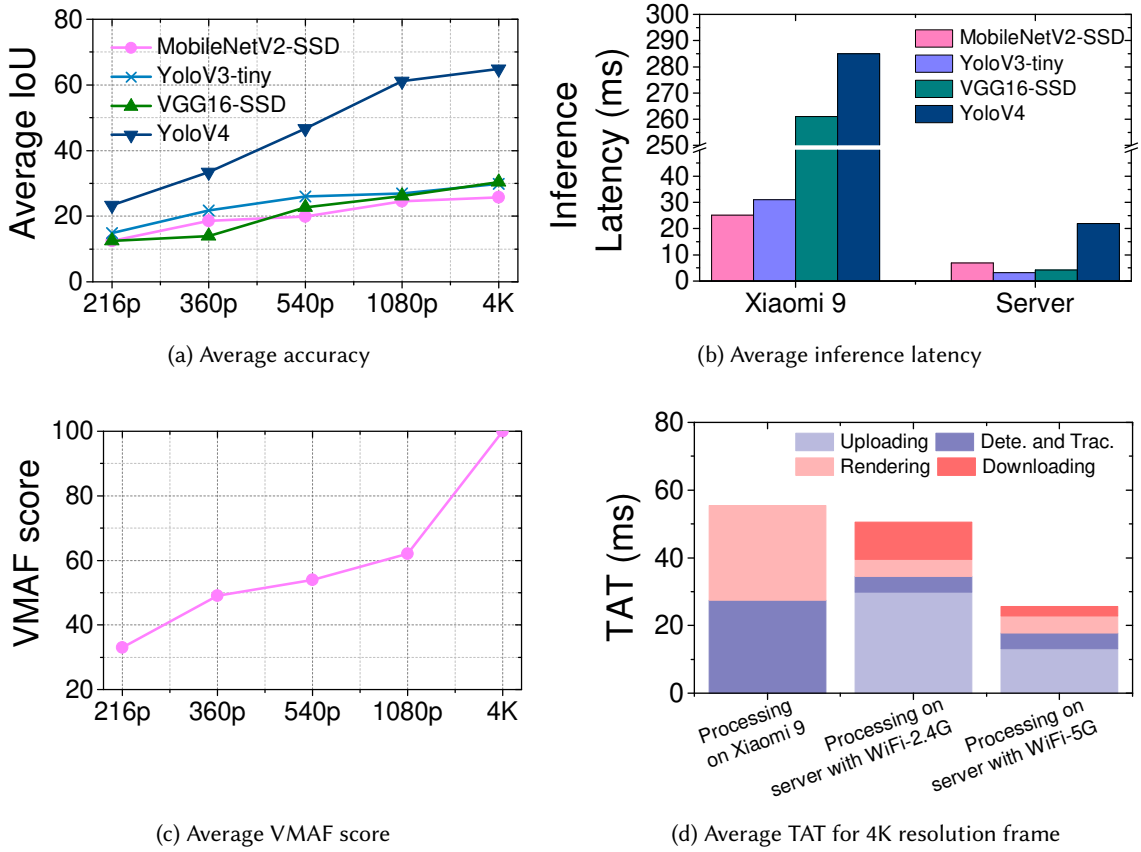
3.1 Setup

This experiment overlays digital contents on physical objects detected from the two video frames depicted in Figures 2(a) and 2(c). Each of the two test videos has 60 frames, lasting for 1 second, i.e., 60 FPS. We consider two different WiFi protocols (WiFi-2.4GHz and WiFi-5GHz). The network settings are based on a large-scale study of real-world network measurements [87].

In this experiment, we use a XiaoMi 9 smartphone as the user device. This device has an 8-core Qualcomm Snapdragon 855 CPU, an Adreno 640 GPU, a Hexagon 690 DSP and 64GB of RAM. Our remote server has a 2.4GHz Intel Xeon CPU and two NVIDIA P40 GPUs. More details of our evaluation platforms can be found at Section 6.1. On the server, we apply the EDVR super-resolution model [101] to upscale the received video to a 4K UHD resolution before further processing. From the upscaled video frame, we use an object detection model (Section 3.3) to detect the objects of interests. To do so, we first identify the targets Region of Interest (RoI) and extract each ROI's features. Next, we match features between the reference surface (the left image of Figures 2b and d) and the target image (the right image of Figures 2b and d) for AR rendering. Then, we apply a template-based algorithm for object tracking before overlaying 3D digital objects on the detected objects as shown on the right image of Figures 2b and 2d.

3.2 Evaluation Metrics

In this experiment, we consider three metrics, *turn-around-time (TAT)*, *FPS*, and the *video multi-method assessment fusion (VMAF) score* [62, 63]. TAT and FPS are used to evaluate if a strategy can provide real-time processing and smooth video play. VMAF is a perceptual video quality assessment method [55]. It quantifies user-perceived video quality by computing the difference between a reference and distorted video sequence (see also Section 2.1).



(a) Average accuracy

(b) Average inference latency

(c) Average VMAF score

(d) Average TAT for 4K resolution frame

Fig. 3. The average accuracy (a), inference latency (b) and VMAF score (c) for two motivation video samples under different resolutions, and the breakdown of average TAT (d) of each frame for two common AR pipelines when using 4k resolution videos.

3.3 Local vs Remote MAR Processing

In this experiment, we apply four representative object detection models, YoloV4 [5], YoloV3-tiny [73], MobileNetV2-SSD[80], and VGG16-SSD[84], to detect objects of interests from the motivation videos. We note that MobileNet and YoloV3-tiny are specifically tuned for embedded systems.

Subfigures (a) and (b) in Figure 3 respectively show the accuracy and processing latency when running each model on the mobile device and a remote server. We can see that YoloV4 achieves the best accuracy for all scale resolutions when compared with the other three models. The high resolution frame indeed helps improving the object detection performance. This finding is in line with prior studies [6, 52, 82, 119]. However, it is challenging to achieve low latency for YoloV4 on our mobile device. Running YoloV4 on XiaoMi 9's GPU requires a processing overhead of 286ms. Such processing latency is beyond the delay that can be tolerated by an average user. Offloading the object detection task to the server can significantly reduce the latency and achieve high accuracy, especially for processing high-resolution video contents.

Table 1. Video downscaling resolutions considered in this work and the resulting video size for the motivation examples in Figure 2.

Resolution	Simple (KB)	Complex (KB)
384 × 216	101	228
640 × 360	254	496
960 × 540	457	804

Table 2. Best-found SR configurations (video resolutions and SR frequency) for the motivation videos.

	Resolution - Upscaling factor - freq.	
	WiFi-2.4GHz	WiFi-5GHz
Simple	540p - 4X - 1/24	no action
Complex	216p - 10X - 1/12	540p - 4X - 1/17

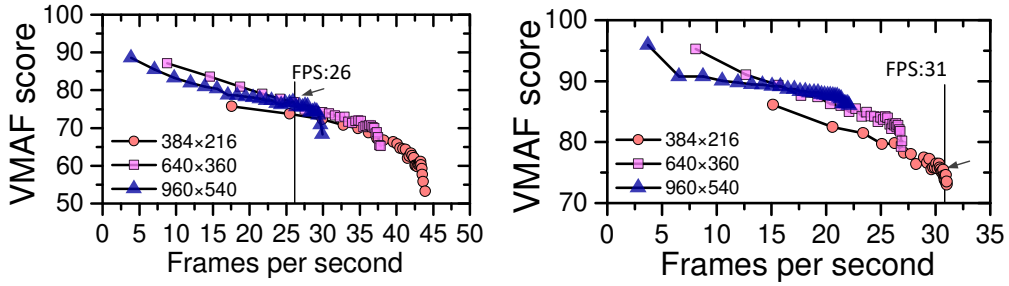
Figure 3(c) presents the VMAF score of the rendered 4K frame using the object detection results from the downscaled frames. Compared with rendering and performing object detection directly on the 4K frame, performing object detection on a down-scaled resolution can lead to insufficient, deteriorating rendering quality with a VMAF score of less than 70. Figure 3(d) compares the average TAT for processing two 4K motivation videos locally on XiaoMi 9 and remotely on the server. We observe that object detection, tracking and content rendering can be computationally expensive, leading to long TAT when they are performed on the mobile device. While computation offloading can greatly reduce the TAT, directly uploading a raw 4K video frame could incur significant communication latency. Even in a WiFi-5GHz environment, uploading a 4K video frame for remote processing still needs 35 ms that can not meet the typical 30 frame-per-second (FPS) camera feeding rate, due to the communication latency. To unlock the potential of remote 4K MAR processing, we thus need to find ways to reduce uplink communication latency.

3.4 Meeting User Expectations

Having shown that reducing the uploading latency is vital for MAR offloading, we now consider if SR can help on this endeavour; and if so, what SR configurations should be used.

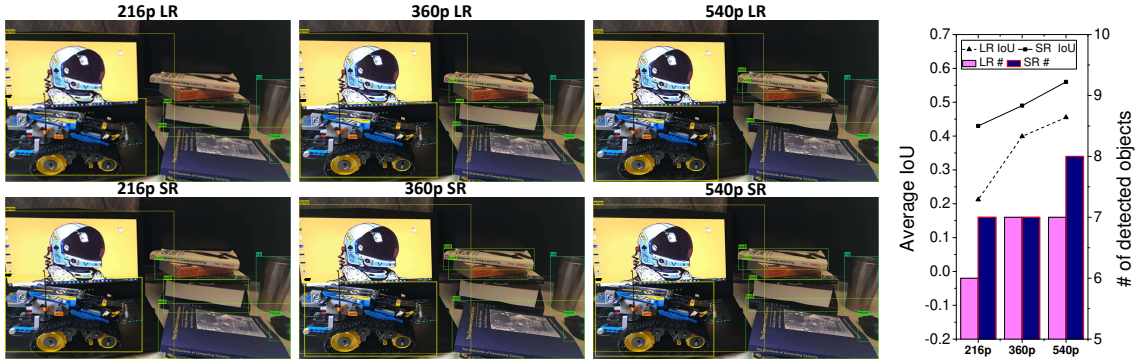
User expectation. We observe that the FPS requirement varies across users and viewing contents, which is consistent with prior research [75]. For most participants in our user study (see Sec. 6.2), the minimum acceptable FPS for the video shown in Figures 2(a) and 2(c) is 26 and 31, respectively. The difference in the FPS expectation is not surprising as the `Complex` video has a more complex background over `Simple`, it requires a higher FPS for meeting the perceptual requirement, which is consistent with previous study[75].

SR configuration. We now consider what SR configurations should be used to meet the minimum acceptable FPS while maximizing the video quality measured by VMAF. In this work, we consider two configurable parameters: the resolution of the video to be sent to the remote server and at what frequency we apply SR to the received video frames on the server-side. In this work, we consider the three downscaling resolutions given in Table 1. We did not consider 1080p, as the latency of 2x SR model consumes the benefit from the reduced latency by downsampling. We also leverage the sophisticated HEVC video encoding scheme [91] to avoid performing SR on every video frame. To this end, we consider 30 SR frequencies, $1/1, 1/2, ..., 1/30$. Here, an SR frequency of



(a) Simple. (b) Complex.

Fig. 4. The resultant VMAF (a) and FPS (b) of different downscaling resolutions in a 4G environment for the videos shown in Figure 2. The vertical line shows the best SR configuration that meets the minimum acceptable FPS while giving the highest VMAF score.



(a) Visualization results (b) Detection performance.

Fig. 5. Visual results(a) and detection performance (b) before and after using SR on the Complex video feed shown in Figure 2. Details of the texts printed on the textbooks vary under different resolutions, affecting the accuracy of object detection and tracking.

$1/n$ means we apply SR to upscale one of every n consecutive video frames and use the SR-upscaled frame to interpolate other frames to obtain a full 4K resolution.

Latency vs video quality. Figure 4 presents the FPS and VMAF for Simple and Complex videos under different SR configurations in a WiFi-2.4GHz network. Here, we wish to find an SR configuration that can satisfy the FPS requirement with the highest VMAF score. For example, the optimal configuration for Simple and Complex is to downscale the video to 540p and 216p, respectively. Table 2 summarizes the best-performing SR configurations. As can be seen from the table, the best configuration changes across videos and networking environments, and choosing the right configuration can obtain a smooth frame rate while maintaining the VMAF score at a good level ($\geq$ 70). Furthermore, the TAT of uploading the raw 4K simple video (32 FPS) does not violate the user acceptable FPS (26 FPS), so we do not need to downscale and perform SR on it (termed as no-action).

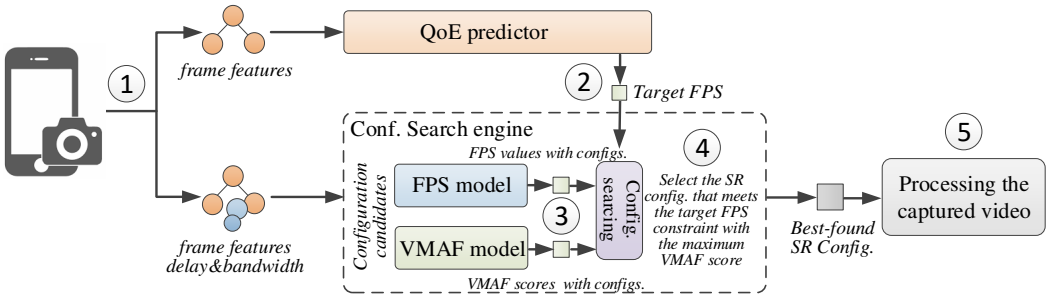


Fig. 6. Overview of our approach.

The effects of super-resolution on object detection performance. To evaluate image resolutions impact on object detection, we count the detected objects and compute the Intersection over Union (IoU) between the detected object bounding box and the ground truth. IoU takes a value between 0 and 1 - higher is more accurate. Figure 5 shows the visualized results and detection performance when we perform object detection and SR directly on the lower-resolution videos. One noticeable change for videos under different resolutions is the loss of details on the printed texts under lower-resolution video samples. As we can see from Figure 5 (b), low-resolution videos lead to a low average IoU, where miss detection can happen. By first applying SR to three low-resolution videos to the full 4K resolution, we can achieve highly accurate object detection, with an average IoU of over 0.4. This example shows that SR can significantly improve detection performance, which is in line with the findings reported in prior studies [6, 68, 82].

3.5 Lesson Learned

This experiment shows that choosing a proper SR configuration (downscaling resolution and SR frequency) is vital for MAR offloading. However, the right SR configuration can vary across users and video content. What we need is a technique that can automatically adapt to the diverse user expectation, video contents and networking environments. This work aims to provide such capabilities.

4 OVERVIEW OF OUR APPROACH

Figure 6 depicts the overall workflow of Actor. Our goal is to find the SR configuration that provides the best video quality while meeting the user's minimum FPS requirement. We use the VMAF score to quantify the video quality, but other metrics can also be used. Our SR configuration includes two parameters: (1) the downscaling resolution and (2) how often we apply SR to the received frames on the server (Sec. 3.4).

At the core of Actor are three models for estimating (1) the resulting FPS, (2) the minimum acceptable FPS and (3) the VMAF score for an SR configuration. Determining the resulting FPS for an SR configuration is straightforward as it depends on several metrics that are largely constant[4] within our one-second sampling window: the network delay and bandwidth (that varies little within our sampling window), the server processing time for a digital object and the video size (that strongly correlates with the video resolution) and section 5.2 gives the detail. Estimating the minimum acceptable FPS and VMAF score is more challenging as the two values change across users and video contents. To this end, we employ machine learning to build predictors for estimating the minimum acceptable FPS and the VMAF score, by taking into consideration the impact of dynamically changing

[4]The network is likely to be dynamic for a longer period, and our approach constantly monitors the network conditions to adjust our optimization accordingly.

Table 3. Network environment settings

	Minimum uplink bandwidth	Minimum downlink bandwidth	Maximum Delay
WiFi-2.4GHz	9.8Mbps	30.1Mbps	15ms
WiFi-5GHz	25.76 Mbps	123.7Mbps	15ms

video scenes and SR configurations. Specifically, the QoE (quality of experience) predictor estimates the minimum acceptable FPS target for a given user and video frame content. Finally, the configuration search engine selects the best suitable SR configuration that meets the minimum FPS constraint with the maximum VMAF score.

We organize the three decision models around two components, a QoE (quality of experience) predictor and an SR configuration search engine, described as follows.

4.1 QoE Predictor

The QoE predictor takes as input features of the raw video frames. It then predicts the minimum acceptable FPS. As the FPS requirement could change from one user to the other, we build a QoE for each target user. To reduce user involvement, we first use supervised learning to train a baseline predictor "at the factory" through a user study. We then tailor the baseline predictor for the target user during the first installation and continuously improve it over time at the end-user environment using a transfer learning approach inspired by [75] (see Sec.5.3).

4.2 SR Configuration Search Engine

The configuration search engine consists of two models for estimating the resultant FPS and VMAF score under an SR configuration for a given video scene. This is achieved by first using an analytical based FPS model (Sec. 5.2) to estimate the TAT, which is then used to calculate the resultant FPS for a given SR configuration. The VMAF predictor takes as inputs an SR configuration (represented as labels for a combination of the downscaling resolution and SR frequency), the video content captured by the mobile camera within a configurable sampling window (one second in this work). It then predicts the expected VMAF score for the upscaling video. By estimating the expected FPS and VMAF score under a given SR setting, the search engine can then quickly search for a Pareto efficiency configuration that meets the QoE requirement (i.e., the minimum FPS) but gives the highest VMAF score.

5 IMPLEMENTATION DETAILS

In this section, we first describe how to employ machine learning to train predictors to estimate the minimum acceptable FPS and the VMAF score. We then describe our simple yet effective analytical model for estimating the resulting FPS of an SR configuration, before discussing how we can combine the three models to perform MAR offloading.

5.1 Predictive Modeling

5.1.1 Model structure. We use the artificial neural network (ANN) to build our predictors, as it can model both linear and non-linear relationships and supports transfer learning [75]. Specifically, we use a fully connected ANN with three hidden layers, 20 neurons per layer. The number of nodes of the input layer is determined by the dimensionality of the model input. It is our intention to keep the model structure simple to minimize the inference (or execution) time on the user device.

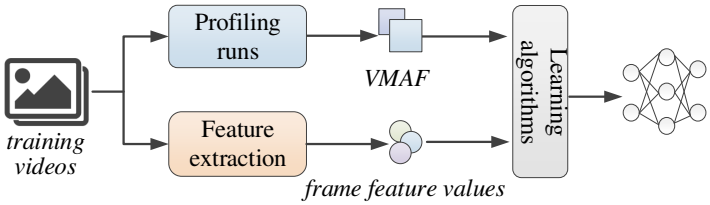


Fig. 7. Overview of our training process for predictors.

5.1.2 Training data generation. We apply cross-validation to train and test our models (see also Sec. 6.4). Training data are generated by profiling a set of training videos under a variety of networking environments and SR configurations. We have considered two networking conditions given in Table 3. Note that since our models take as input the minimum network bandwidth and maximum delay, the trained models are generalized to any other networking environment.

QoE training data. Training data for the QoE predictor are collected through a user study in this work. In practice, this can be done through a crowdsourcing platform like Amazon Mechanical Turk. Our user study involved 20 paid users (10 females) who were studying at our institution. To minimize user involvement, we apply the k-means clustering algorithm [4] to the video feature space to choose a small number of videos as QoE training data. We set $k = 7$ and randomly pick three videos from each cluster, resulting in 21 training videos (see Sec. 5.3). The user study was performed in a controlled WiFi-2.4GHz and a WiFi-5GHz environments (Sec. 6.1). During the experiment, each user watches the screen update of each rendered video with various video frame rates. For each training video sample, we replay the video under different FPS settings, from high to low and ask a user to select the FPS when he/she feels uncomfortable or noticeable drop in video perception quality. We then record the corresponding minimum acceptable FPS on a per-video and per-user basis. Note that training is a "one-off" cost, and the learned model can be applied to "*new, unseen*" videos and mobile platforms. We also stress that the trained QoE predictor can be easily ported to predict the requirement of *a new user* during deployment using low-cost transfer learning techniques by asking user feedback on a few video contents [75].

VMAF training data. Figure 7 depicts the process of using training videos to build the baseline VMAF predictor. To generate training data for estimating the VMAF score, we extract each of the high-resolution frames from a training video corpus of around 2,000 videos (see Sec. 6.2), where each video stream is in a 4K UHD resolution. This training dataset contains videos of 19 categories, including sports, landscapes, home settings, etc., which thus represents a wide range of video workloads. For each 4K video frame, we apply a standard bicubic downscaling algorithm to generate the lower-resolution frame. In this work, we target the three low-resolution settings given in Table 1[5]. We partition each training video to a sampling window of one second. For each low-resolution sampling window, we vary the SR frequency at each training networking environment and record the achieved VMAF score. We consider 30 SR frequencies, $1/1, 1/2, ..., 1/30$. In total, we train the VMAF predictor on over 180 thousand *automatically generated* training samples (~2000 video samples × 3 scaling resolutions × 30 SR frequencies). The learnt model can be applied to any video.

5.1.3 Features. In this work, we considered a total of 10 candidate features for building the QoE and the VMAF predictors, given in Table 4. These features were chosen based on previous work of image analysis [94] and our intuition, e.g. the spatial and temporal perceptual information that describes the complexity and mobility of a video frame.

[5]We found other downscaling resolutions either offer little benefit for reducing the TAT or often lead to a low VMAF score.

Table 4. Raw features in this work

Feature	Description
SI	Spatial perceptual Information
TI	Temporal perceptual Information
perceived_brightness_rms	Root mean square of perceived brightness
contrast	Standard deviation of the greyed image pixel intensities
n_keypoints	keypoints
edge_length	The distance between edge pixels detected
hue_rms	Root mean square of hue
value_rms	Root mean square of value
saturation_rms	Root mean square of saturation
frame_size	Frame size in KB

Table 5. Removed features

Kept Feature	Removed Feature	Cor.
brightness_rms	saturation_rms	0.844
SI	edge_length	0.837

Machine learning requires sufficient training data to learn an efficient model over a multi-dimensional feature space. To learn effectively over a small training dataset, we use correlation-based feature selection to reduce the feature dimensionality. We performed this by constructing a matrix of correlation coefficients using the Pearson product-moment correlation (PCC). The coefficient value falls between −1 and +1. The closer the absolute value is to 1, the stronger the correlation between the two features being tested. If a pair of features have a PCC greater than a threshold (0.75 in this work), we drop one of them and keep the other. Table 5 lists the features removed at this stage, leaving 8 features as a result.

5.2 Estimating the Resultant FPS

To compute the expected FPS for an SR configuration, we estimate the TAT for the video frames within our sampling window. TAT is computed as $t_{tf} + t_{server}$, where t_{tf} and t_{server} are the data transfer and server processing time respectively. We calculate t_{tf} as $vs_{reso}/us + macro_{4k}/ds + delay$, where the vs is the video size under a resolution, $macro_{4k}$ is the typical macroblock size for encoding the generated AR objects (determined through offline profiling), and us and ds are the measured upload and download speed respectively. Initially, we calculate the download latency by using the whole frame size, so our approach can always provide a higher actual FPS than the estimated FPS value at first, which guarantees the user QoE. Next, we use the actual macroblock size to calculate the FPS value. For each estimation, we compare the real FPS with the outputs of FPS model and adjust the parameters of FPS model.

We found that the video size, vs, is highly correlated with the first frame size and temporal perceptual information (TI) (see also Sec. 6.2). Based on this observation, we employ a linear regression model that takes in the size of the first video frame and the number of video frames to estimate the video size within a sampling window. Estimation of the video size is formulated as:

$$vs = frameSize + \alpha \times frameSize \times TI \times n$$

where weight parameter α is learned from our training dataset, and n denotes the number of video frames within the sampling window. We apply a curve-fitting regression algorithm by minimizing the least square error to derive α from our video training data.

The server processing time, t_{server}, is determined *offline* by profiling the model inference time for SR, object detection and tracking, and rendering. It is computed as:

$$t_{server} = (t_{detect} + t_{SR})_{SelectedFrame} + t_{track} + t_{render}$$

where t_{SR} and t_{detect} is time spent on the selected frames (determined by the SR frequency) when performing SR upscaling and object detection, and t_{track} and t_{render} respectively denote the time spent for tracking and rendering.

Finally, we compute the resultant FPS by dividing the TAT (e.g., in milliseconds) by the number of video frames, n, of our one-second sampling window, i.e., $FPS = 1000/TAT/n$. We compare the FPS given by our formula against the measurements of over 2,000 testing videos (Sec. 6.2) under all considered SR configurations and all networking environments in Table 3. We found that our FPS model is accurate, with an error rate of less than 5%.

5.3 QoE Predictor

ACTOR finds out the actual QoE target by seeking user feedback. This is done by automatically replaying the video under different FPS settings, from high to low. For each setting, ACTOR asks the user to rate the current video for being "acceptable" or not. It stops playing the video when the user indicates an FPS setting is unacceptable.

The expected QoE value is different for each user. Therefore, using a generic model across different users and hardware platforms is ineffective. To tune a model to match a specific user or mobile device, ACTOR employs transfer learning[75] to quickly port a baseline predictor to the target computing environment.

Prior work in other domains has shown that ANN models trained on similar inputs for different tasks often share useful commonalities[116]. Our work leverages this insight to speed up the process for tuning a model for a new user. This is because the first few layers (i.e., those close to the input layer) of our ANN are likely to focus on abstracting video features and are largely independent of the model output. Since we use the same network structure, transfer learning is achieved by copying the weights of a baseline model to initialize the new network. Then, we train the model as usual but using profiling information collected from fewer training videos.

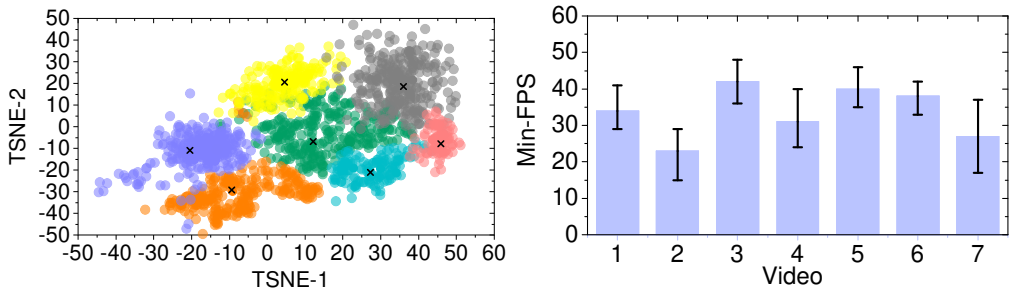
To select the representative videos for training, we group our training videos using the k-means clustering algorithm and leveraging the *Bayesian Information Criterion* (`BIC`) score to to determine the right number of clusters (i.e., K). The BIC measures if a selected K is the best fit for grouping data samples within a dataset. The larger the score is, the higher the chance that we find a good clustering number for the dataset. The BIC score is calculated as:

$$BIC_j = \hat{l}_j - \frac{p_j}{2} \cdot logR$$

where $\hat{l}_j$ is the likelihood of the data when K equals to j, R is the number of training samples, and the free parameter p_j is the sum of $K-1$ class probabilities – calculated as: $p_j = (K-1) + dK + 1$ for a d-dimension feature vector plus 1 variance estimate. $\hat{l}_j$ is computed as:

$$\hat{l}_j = \sum_{n=1}^{k} -\frac{R_n}{2} log(2\pi) - \frac{R_n \cdot d}{2} log(\hat{\sigma}^2) - \frac{R_n - K}{2} + R_n log(R_n/R)$$

where R_n is the number of points in a cluster and $\hat{\sigma}^2$ is the distance variance between each point to its cluster centroid.



(a) Using clustering to choose training examples for transfer learning. A cluster centroid is marked by a cross.

(b) Min. accpetable FPS for 20 participants across 7 aselected videos

Fig. 8. We cluster our dataset into 7 groups determined by using the BIC score, and apply t-SNE to project the data onto a two-dimensional space. A cluster centroid is marked by a cross (a), and the minimum accepted FPS of 7 centroid videos for 20 participants (b).

Figure 8(a) illustrates how one of our training dataset of 2,360 videos is grouped into 7 clusters determined using the BIC score. To aid the clarity, we apply t-SNE to project the data onto a two-dimensional space. Figure 8(b) presents the min-accepted FPS of 7 represents videos for 20 participants. The min-max bars show the variances across 20 users for each video, and it is also worth noting that the diversity of accepted FPS for each video. To speed up the training process of QoE model for a new user, we randomly select 3 videos around centroid of each cluster and apply transfer learning to the basic QoE model, which gives an average error rate of 11.8%.

When the QoE gives an inaccurate acceptable FPS prediction for the current video content, Actor then finds out the actual QoE target by seeking user feedback. This is done by automatically replaying the screen update under different FPS settings, from high to low. For each setting, Actor asks the user to rate the screen update for being "acceptable" or not. It stops playing the screen update when the user indicates an FPS setting is uncomfortable. When the device is charging, Actor runs the learning algorithm to update the QoE model.

5.4 Remote Server Processing

Figure 9 shows the processing pipeline on the remote server. The downscaled video and the SR frequency configuration are sent to a remote server for processing. Note that the server only performs SR on the i^{th} video frames (referred as *key video frames*) specified by the SR frequency, but it decodes the remaining video frames from the received video stream as usual. Next, it applies a bicubic algorithm to quickly upscale the non-key video frames, using the SR-scaled high-resolution frames as references. We apply an object detection algorithm to the key video frames to detect the object of interest. For the remaining upscaling frames, we use a simpler but faster object tracking algorithm to locate the object identified from the previous key video frames. Finally, we apply a rendering algorithm to the detected and tracked objects and send back the processed video macroblocks to the end-user. Note that we only send back macroblocks that contain the overlayed contents generated by AR. The rendered macroblocks will be used to replace those in the corresponding raw 4K video frame. By only sending back the changed macroblocks generated by AR, we minimize the download latency.

To further reduce the processing overhead, we also adopt a set of optimization techniques. First, we reduce the inference time of the SR model through model compression [61]. We also employ video caching [109] to leverage temporal locality of video frames to reuse the detection and localization results of previously frames. As we will

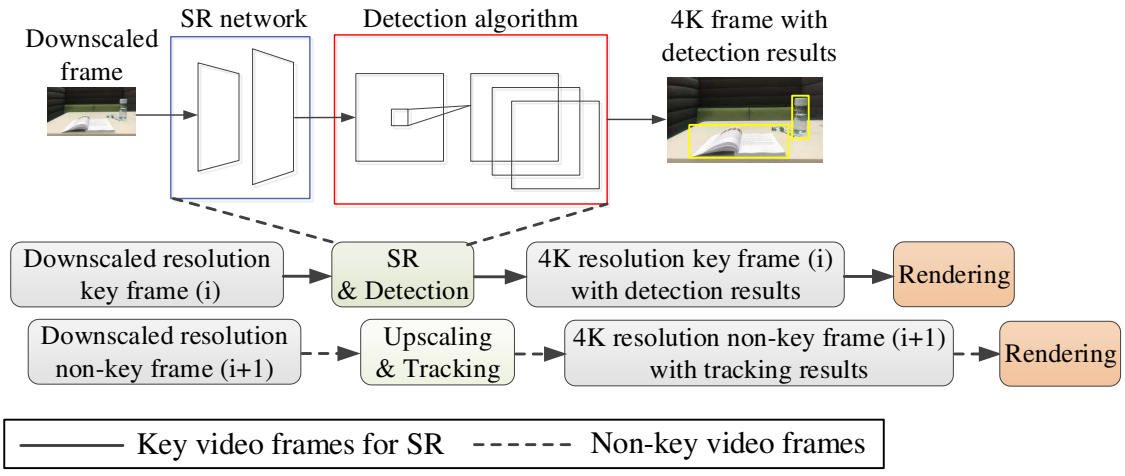


Fig. 9. Sever-side processing for remote MAR offloading.

show in Sec. 7.5.3, the server processing time accounts for less 20% of the TAT on a modest server configuration used in this work.

5.5 Putting All Together

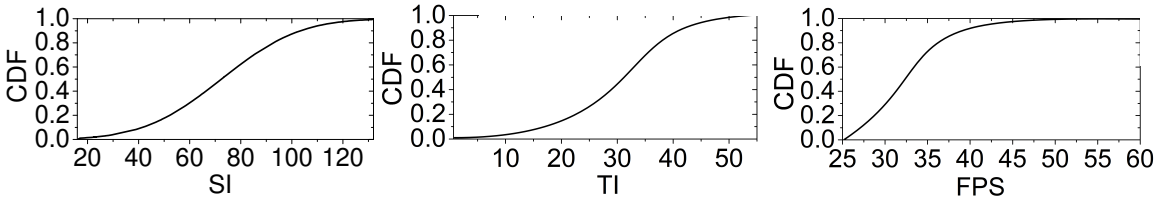
Once we have built the predictive models, we can use them for "*new, unseen*" videos and networking environments. On the user device, Actor runs a background network monitor to measure and report the bandwidth and delay for both the uplink and downlink between the user device and the remote server. To predict the optimal SR configuration, we extract features from the mobile video feed contents within every one second (as described in Sec. 5.1.3). As a cold start (i.e., for the first few video frames), we use a default downscaling resolution of 540p and SR frequency of 1/21, but this configuration will be dynamically adjusted over time. We use the user-specific QoE predictor (Sec. 4.1) to predict the minimum accepted FPS based on the video features. The predicted FPS requirement is passed to the SR configuration search engine (Sec. 4.2) to quickly find the optimal SR configuration. The chosen downscaling resolution is used to encode the video feed according to the HEVC standard [37], and the SR frequency is passed to the remote server together with the dowscaled video. In this work, we use H.265 [37] to encode and decode the video, but other encoding algorithms can be used too. Upon receiving the video, the remote server firstly applies the SR model to upscale the video frames specified by the SR frequency. The server then uses the upscaled video frame to scale up all other frames according to the method described in Sec. 5.4. Object detection, tracking and rendering are performed on the upscaled full 4K video frames. Macroblocks of each video frame that contain the overlayed contents generated by the render are then sent back to the user.

6 EVALUATION SETUP

6.1 Software and Hardware Systems

Hardware platforms. We use a XiaoMi 9 smartphone as the mobile device. The remote server has a 16-core 2.4GHz Intel Xeon CPU and two NVIDIA Tesla P40 GPUs.

Software implementation. Our predictive models build on PyTorch v1.2. For video processing, we use FFmpeg [25] to encode and decode the video stream on both the mobile device and the server. We developed a FFmpeg



(a) Spatial percep. info. (b) Temp. percep. info. (c) Min. FPS

Fig. 10. CDF of spatial (a), temporal (b) perceptual information and the minimum acceptable FPS (c) for test videos.

plugin to run on the user device to extract video features as described in Sec. 5.1.3. We applied a bicubic downscaling algorithm [59] (running on our mobile device) to the input 4K video feed to generate the lower-resolution videos as given in Table 1. *Note that we include the time for downscaling in all our experimental results.* On the server, we use the optimized (Sec. 5.4) EDVR model [101] for SR, a YoloV4 [5] for object detection, and a template matching algorithm from OpenCV [8] for object tracking. We use "leave-one-out" cross-validation (Sec. 6.4) to train the EDVR model. We use ARToolKit v5 [3] to generate various digital objects with different rendering times for object rendering. For simplicity, we render the same annotation on each detected object for all videos.

Networking environments. To ensure reproducibility, we evaluate all schemes in a controlled environment. Our mobile device and the server communicate through a WiFi connection. We use Netem [32] to control the network delay, packet loss and bandwidth to simulate the environments in Table 3. We add 20% of variances (following a normal distribution) to the bandwidths, delay (jitter) and packet loss to simulate a dynamic environment, but ensure that the variances are the same during the replay of a test video for reproducibility. Moreover, the network simulation parameters we used in this work is effective to evaluate the network performance[7, 40]. *This setup allows us to playback the network settings to ensure all approaches are tested in the same network environment for a fair comparison.*

6.2 Datasets and User Participants

Datasets. We use 2,360 4K video clips as the mobile video feeds in evaluation as detailed in Table 6. These videos are obtained from the SJTU [85], Ultra Video Group [60] and CableLabs [9] datasets. Figure 10 shows the CDF of the spatial (SI) and temporal (TI) perceptual information of our workloads, calculated according to [67]. SI quantifies the complexity of the spatial details present in a video sequence, whose score increases as the spatial complexity of the samples grows. TI quantifies the motion variation of the video sequences - more motion in adjacent frames will lead to a higher value of TI. We can see from the diagrams, the values range from small (18.6 for SI and 0.7 for TI) to large (132 for SI and 51.7 for TI), indicating that our test data cover a diverse set of video contents and characteristics.

FPS expectation. Figure 10(c) plots the minimum acceptable FPS given by our 20 participants for all testing videos (see also Sec. 5.1.2). As we can see from this figure, the minimum accepted FPS is typically above 25, and over 50% of users expect more than 30.

6.3 Competitive Approaches

We compare ACTOR to the following five schemes:

- `MAX`: This strategy selects the SR configuration (apart from performing SR on every 4K video frame) that maximizes the VMAF score. It gives the best video quality but can compromise the latency.

Table 6. Dataset description

Video source	Clips	Content Description
Ultra Video Group[60]	370	Plant, Animal activity, Human face, Horse racing, Boat,City alley, Architecture
SJTU[85]	230	Construction field, Traffic flow, Runners, Human, Building and Plant
CableLabs[9]	1760	Landscape, Human activity, Home settings, Animal activity, Car, Soccer, Skateboarding

- `CloudAR`: This prior work uses a cloud server to perform the object identification task, but object tracking and rendering is performed locally on the mobile [119].
- `EAR`: This state-of-the-art MAR offloading method uploads selected macroblocks of the target video frame to a remote server to perform object detection remotely [52].
- `CloudAR-SR and EAR-SR`: CloudAR-SR and EAR-SR leverage SR techniques on the original CloudAR and EAR basis, respectively. Both strategies downscale the 4K frame with 4x factor at first, and if they can not achieve the acceptable user FPS, they perform aggressive downscaling with 6 or 10 downscale factors.

6.4 Evaluation Methodology

Cross-validation. We use five-fold cross-validation to train all models. Specifically, we randomly partition our 2,360 videos into 5 sets where each set contains 472 videos. We keep one set as the validation data for testing, and the remaining 4 sets as training data to learn a model. We repeat this process five times (folds) to make sure that each sets used exactly once as the validation data. This is a standard methodology for evaluating the generalization ability of a machine learned model.

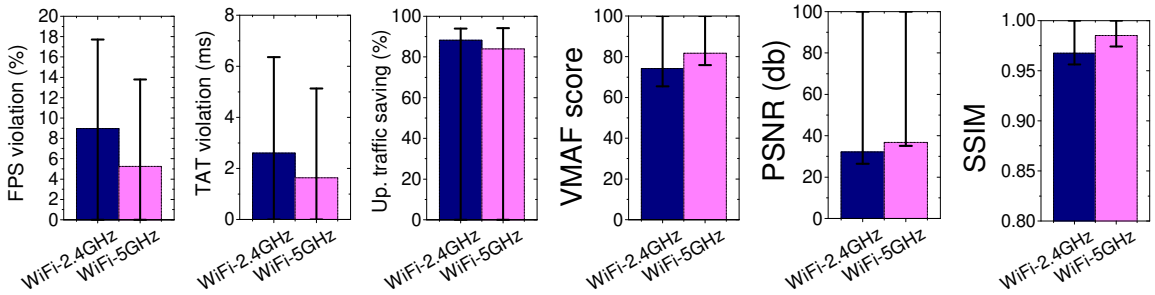
Evaluation metrics. To measure how often a scheme fails to meet the user expected FPS, we report the FPS violation ratio. This "*lower-is-better*" metric is calculated as δ/FPS_{min}, where δ is the number of FPS falls below the minimum acceptable FPS, FPS_{min}. We also present the violation in millisecond. To evaluate the user-perceived video experience, we consider three widely used video quality metrics: VMAF (see Sec. 3.2), the peak signal-to-noise ratio (PSNR)[103], and the structural similarity index (SSIM) [103]. To compute the metrics, we use the AR performed directly on the original 4k video frame as the baseline (i.e., the reference video).

Server processing delay. To evaluate the impact of server-side processing delay (e.g., due to the use of a larger DNN or more complex rendering), we also vary the processing time through increasing the complexity of the AR rendering tasks (i.e., by generating more digital objects with a higher degree of details). This leads to a processing delay ranging between 2ms and 20ms on our server platform.

Performance report. We use the geometric mean to compute the resultant FPS and all metrics. The geometric mean is widely seen as a more reliable performance metric over the arithmetic mean [24]. Unless state otherwise, we report the geometric mean across test videos, 20 users and the 2 networks given in Table 3, using cross-validation. To have statistically sound data, we run each approach on each test case repeatedly until the confidence-bound under a 95% confidence interval is smaller than 2%.

7 EXPERIMENTAL RESULTS

In this section, we first show that Actor significantly reduces the remote processing latency without incurring frequent FPS violations (Sec. 7.1). We then compare Actor against alternative methods (Sec. 7.2) before analyzing the working mechanism of our approach (Sec. 7.3 and Sec. 7.5) and discussing the limitations (Sec. 7.6).



(a) FPS violation (b) TAT violation (c) Uploading traffic saving (d) VMAF (e) PSNR (f) SSIM

Fig. 11. The FPS (a) and TAT (b) violation delivered by ACTOR, and improvement achieved by Actor over directly uploading 4K raw videos for the uploading traffic (b) and the corresponding VMAF (c), PSRN (d), and SSIM (e) scores.

7.1 Overall Results

Figure 11 presents the results of Actor under two networking environments, where the min-max bars give the variances across video datasets and 20 participants. From Figure 11a, we see that Actor rarely violates the FPS constraint with an average FPS violation rate of 6.7% (up to 17.8%, which translates into a fall of TAT with 6.1 ms). Our approach aims to deliver the best possible VAMF score while giving the least FPS violation. We note that a zero FPS violation is often impossible due to the high processing overhead of 4K AR. Prior studies suggested that FPS can have a significant impact on the user experience when viewing a video [75]. This is because a high frame rate help to maintain smooth motion and crisp details of the video. Our user study (Section 7.4) shows a low FPS violation ratio does lead to improved user experience. Therefore, our approach is designed to minimize FPS violation.

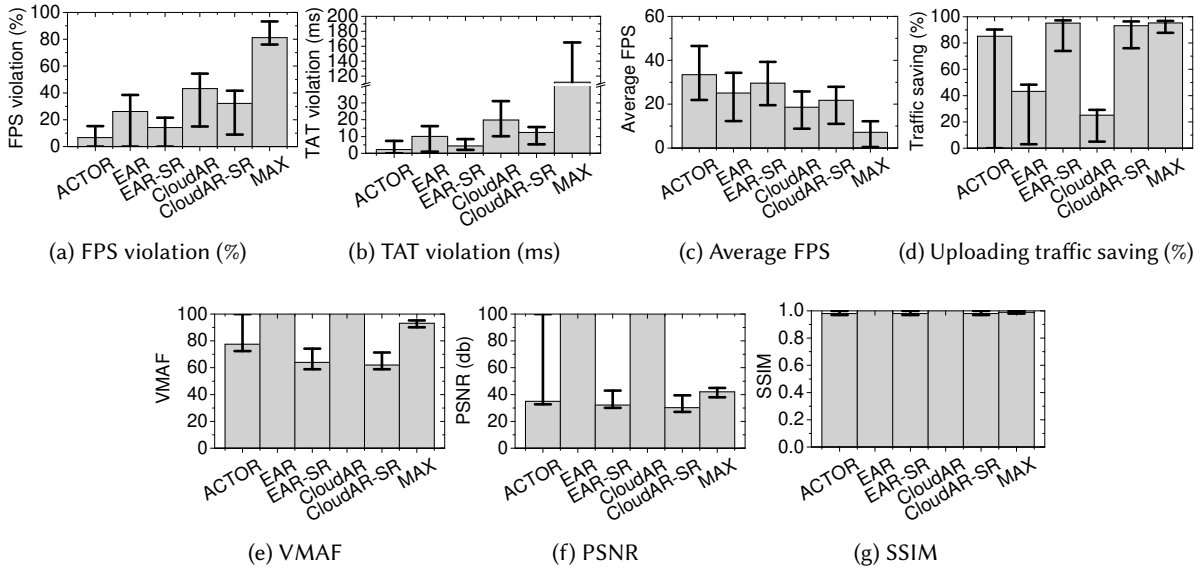
If we look at Figure 11c, we see that Actor can reduce the uploading data by over 80% across networking environments. This large saving in uplink communication does not compromise the user experience as Actor still maintains a good VMAF score of over 70. Similarly, it gives a high score on PSNR and SSIM, two alternative video quality evaluation metrics. Actor can also adapt to the change of networks. Specifically, Actor leverages the higher bandwidth and lower latency given by a higher-speed network to use a higher downscaling resolution to improve the user perceptual experience. For example, it obtains an average score of 83 for VMAF, 38.3 for PSNR and 0.98 for SSIM in the WiFi-5GHz network. This experiment shows that Actor can effectively adapt to video contents to trade user perceptual experience for real-time MAR computation offloading.

7.2 Comparison with Alternative Approaches

Figure 12 compares Actor with five alternative methods across network environments and videos. The min-max bars show the range of improvements achieved across videos.

By considering the resultant FPS as an optimization constraint, Actor gives the least frequent FPS violations with 6.75% FPS Violations (a fall of TAT with 2.28 ms). A low FPS violation ratio suggests a smoother video playing experience and is important for many interactive applications. By directly operating on the raw 4K videos, `EAR` and `CloudAR` give a high score on the perceptual quality metrics[6]. However, they suffer from long uploading

[6]The disparity of PSNR and VMAF is because human eyes do not notice small improvements that have little impact on the VMAF score but could give an artificially good PSNR.



(a) FPS violation (%) (b) TAT violation (ms) (c) Average FPS (d) Uploading traffic saving (%)

(e) VMAF (f) PSNR (g) SSIM

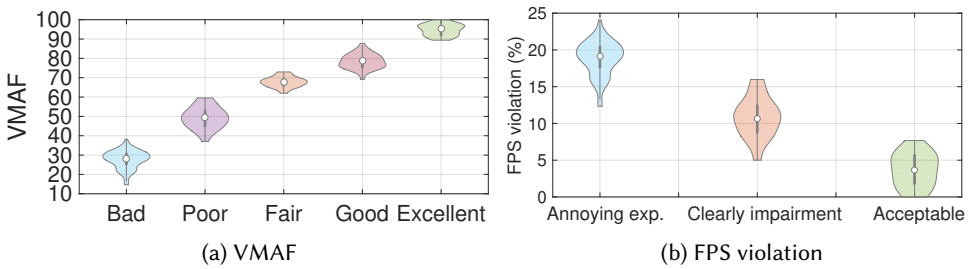
Fig. 12. Min-max bars showing the distribution for Actor, EAR, EAR-SR, CloudAR, CloudAR-SR and MAX in 2 different network environments. Actor gives the best performance for meeting both the user-perceptual experience and the user expected FPS.

latency and high rendering overhead on mobile when processing 4K videos and incur frequent QoE violations, often failing to meet the FPS requirement. Similarly, Max gives a high score on the video quality metrics at the cost of having the most frequent FPS violations, significantly affecting the smoothness of video playing.

In addition, we extend the EAR and CloudAR by leveraging SR technique, termed as EAR-SR and CloudAR-SR. In detail, EAR-SR and CloudAR-SR downscale the 4K raw frame with 4x downscale factor at first, if they can not achieve the min-fps, then perform aggressive downscaling with a 6x or 10x downscale factor. As we can see from Figure 12(a), both strategies reduce the FPS violation after applying SR, and EAR-SR only violates FPS by an average of 13.45%. EAR-SR and CloudAR-SR do not consider the current frame workloads and can not use SR adaptively like ACTOR, and they perform high-resolution frame rendering on the local mobile, thus violating the FPS more often than ACTOR. Specifically, Figure12(c) shows that Actor achieves the highest average FPS by 38 (up to 48) when compared with the other five alternative schemes. For user-perceptual experience, Actor achieves a high VMAF score of at least 73 (up to 88), which is not far away from that of MAX (92) – a strategy that solely aims to maximize the VMAF, but has much lower FPS violations (6.75% vs 81.2%). We also observe a similar trend for PSNR and SSIM, where Actor's video quality is close to MAX but with much fewer FPS violations. Furthermore, as a by-product, Actor reduces the energy consumption of the mobile system by 43.3% (up to 56.8%) compared to EAR and CloudAR.

7.3 QoE Model Evaluation

We divide the 20 participants of our user study into 3 groups based on their minimum acceptable FPS. The low-expectation group has 7 users with an averaged minimum-acceptable FPS target of under 30; the moderate-expectation group has 10 users with an averaged minimum-acceptable FPS target of between 28 and 40; and the high-expectation group has 3 users with an averaged minimum-acceptable FPS target of over 40. After applying transfer learning (with cross-validation) to port a QoE predictor to another user from the same or a different



(a) VMAF (b) FPS violation

Fig. 13. Impact of VMAF (a) and FPS violation (b) on user experience.

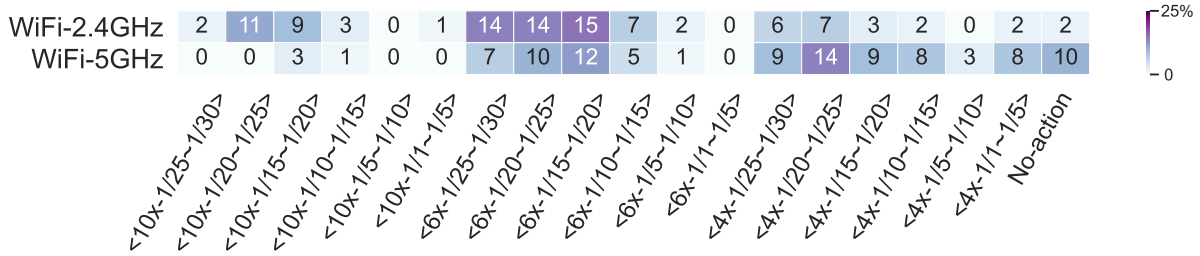


Fig. 14. The percentage of videos that belong to an SR configuration. The distribution of appropriate configuration changes across environments, showing the need of an adaptive scheme.

group. As expected, transfer learning within the same user group gives the lowest error rate of between 5.8% (2.56 FPS) and 8.7% (3.78 FPS). We see a slight increase in the error rate when applying transfer learning across user groups, but the average error rate is 12.6% (5.46 FPS).

7.4 User study on VMAF and FPS

Figure 13 presents the results of our user study regarding VMAF and FPS. Since the user experience of video and AR quality is a subjective matter, which is challenging to provide a rigorously quantified analysis. Nonetheless, our study provides some useful insight into how the VMAF score and FPS to correlate user experience in MAR.

To quantify the VMAF score with user experience, we ask our 20 participants to view the processed videos (from the seven selected videos in Section 5.3) and rank the video quality (measured by VMAF) from five categories of "bad", "poor", "fair", "good", and "excellent". As we can see from Figure 13 (a), a VMAF around 30 is considered to give a bad user experience, and a VMAF of above 90 delivers an excellent video quality. Our participants consider the video quality to be "good" when the VMAF score falls between 69 and 87, although this varies across users and video contents. This finds matches the observation reported by other researchers [62].

7.5 Model Analysis

7.5.1 SR configuration distributions. Figure 14 shows the distribution of the most appropriate SR settings in two typical networking conditions (Table 3). The x-axis gives the percentage of the videos that fall into an SR configuration. We use the notation <Downscaled times- SR frequency> to denote a SR configuration. For example, <4x-22> means that we downscale the origin 4K resolution frames in 4 times on mobile, and then apply SR to upscale 1 of every 22 consecutive video frames and use the SR-upscaled frame to interpolate other

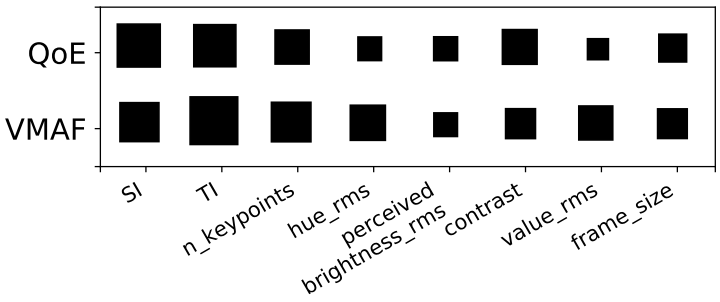


Fig. 15. Hinton diagram showing the importance of video features to the model accuracy. The larger the box, the more likely a feature affects the accuracy of the respective model.

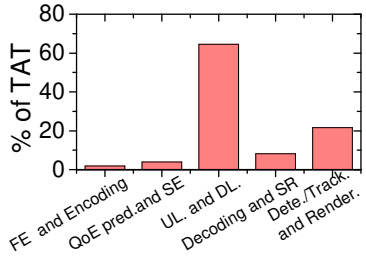


Fig. 16. Actor overhead to turn-around time.

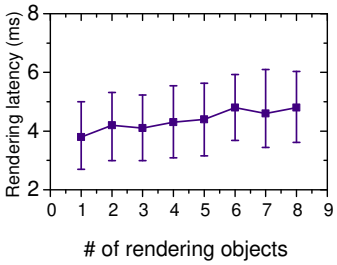


Fig. 17. Rendering latency with different rendering workload.

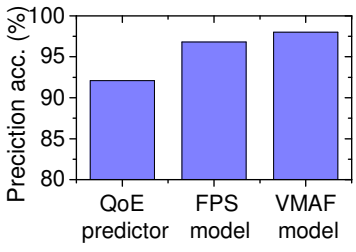


Fig. 18. Prediction accuracy.

frames to obtain a full 4K resolution. We can see that the appropriate SR configurations vary across networking environments and the change of distribution in frequencies when moving from a slow network to a fast one. For instance, it will cause an acceptable TAT to apply SR in 4 times for most video contents in a slow network, such as WiFi-2.4GHz networking environment with only 20% of video contents, while it is the desired SR configuration for 51% of video contents in a WiFi-5GHz environment. If we compare the distributions across networks and metrics, we find that the appropriate SR configuration varies across networking environments, video contents and user experience. The results reinforce our claim that the scheduling policy must be aware of the network, videos contents and user experience.

7.5.2 Feature importance. Figure 15 shows a Hinton diagram for video features (Table 4) that have an impact on the QoE and VMAF models. The larger the box, the more significance a feature contributes to the prediction accuracy. We calculate feature importance by first training a full model using all chosen *video features* and record the accuracy of our model. In turn, we remove each of our features, retraining and evaluating the two predictors on the remaining 8 features, noting the drop in the accuracy. We then normalize the values to produce a percentage of importance for each of our features. Features like `SI`, `TI` and `n_keypoints` are useful for quantifying the complexity of the video scene and are hence important for the predictive models. Feature `value_rms`, on the other hand, is important for estimating the VMAF but less important for estimating the user expected FPS requirement.

7.5.3 Overhead breakdown. Figure 16 shows breakdown of processing overhead of Actor. SR is the most computational expensive operation. However, as it is performed on the server, it only accounts for 8.1% of the end-end latency, and can be further accelerated using multiple CPUs and GPUs. The time spends on video

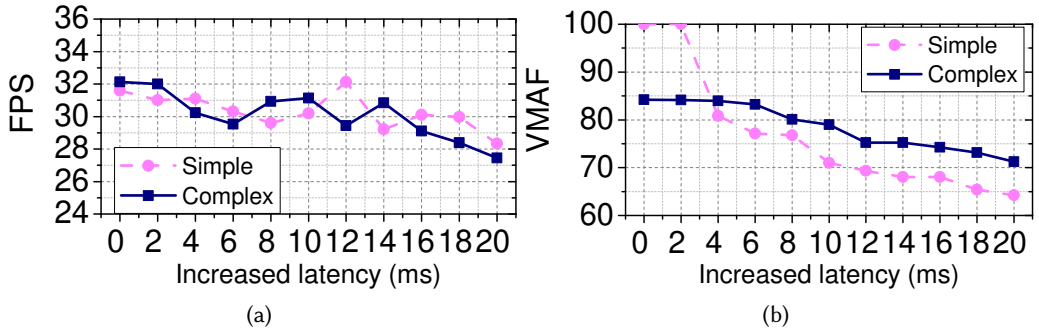


Fig. 19. Changes of FPS (a) and VMAF (a) as the server processing latency increases for each frame.

processing (including feature extraction and encoding) is small, contributes to less than 2% of the TAT. Similarly, QoE prediction and configuration search using the less optimized Python code accounts for less than 3% of the TAT. By contrast, video uploading and downloading domain the TAT, contributing to 63% of the TAT. This experiment shows the importance of reducing the communication overhead for MAR offloading. Figure 17 shows the rendering latency on the server when the number of needed rendering objects increases from 1 to 10. The results are averaged over 50 runs with standard derivation. After object detection and template matching, the powerful computing server can quickly render the object on the targets. To sum up, by reducing the uploading video size, Actor thus cuts down the communication overhead, allowing a smoother and higher-quality AR experience.

7.5.4 Prediction accuracy. Figure 18 shows that the predictive models of Actor are highly accurate, achieving an accuracy of 92.9%, 96.8% and 98% for estimating the user expected FPS, and the resultant FPS and VMAF, respectively. Overall, the performance of Actor is not far from the oracle.

7.5.5 Impact of server processing delay. In this experiment, we vary the server processing time to examine whether Actor can adapt to the change of server processing delay. This experiment also evaluates the impact of changing server time due to the use of more advanced and higher-quality AR rendering. In this experiment, we use the two motivation videos shown in Figure 2 and conduct the experiment in a WiFi-5GHz environment. We set the minimum acceptable FPS to 30. Figure19 shows the resultant FPS and the VMAF score as we increase the server processing delay from 2ms to 20ms for processing a video frame. Actor can adapt to such changes in server processing time without significantly compromising the QoE. As we increase the server processing time, it becomes harder to meet the user expected FPS. However, Actor is able to meet the FPS target at the cost of a small decrease in the VMAF score (but still maintaining the VMFA at good level with a score $\geq$ 70) via choosing an appropriate SR configuration. This experiment confirms that Actor can adapt to the change of server workloads.

7.6 Discussions and Future Work

Naturally, there is room for improvements. We discuss a few points here.

Changing network conditions. To ensure reproducibility, we deterministically replayed the network conditions during evaluation. We show that Actor can adapt to the variation of bandwidth, packet loss, and network delay. It achieves this through periodic sampling monitoring and adjustment. The adaptiveness of Actor can be further improved by considering other network parameters like the signal strength and utilizing the systems-wide

profiling information provided by the mobile base station. Doing so will allow Actor to react to drastic network changes proactively [99].

Apply to other application domains. Actor can be applied to other vision-based offloading tasks like cloud gaming [64]. This will require training our predictive models using domain-specific workloads, but the majority of our techniques can be re-used. While our techniques are evaluated in the smartphone settings, we believe our approach can be equally applied to other AR devices, including head-mounted devices (e.g., Dream Glass[27]) and TV. It would be an exciting challenge to see if the sensor data and the eye-tracking capability can be utilized to capture the user interested region to guide the optimization.

Privacy preserving. Privacy-preserving is an open problem for computation offloading. Work in video and image privacy-preserving [20, 42, 57, 78] is thus orthogonal to Actor. Our future work will look into this.

Reducing uploading latency. A possibility to further reduce the uploading latency is to first estimate the region of interest (RoI) of a video frame and only send macroblocks of ROI to the remote server [52] for SR. To reduce the computation overhead on the user device, the ROI estimation can be based on the object detection results performed by the remote server on a few most recent frames. We leave this as our future work.

Joint network for SR and object detection. Our current implementation uses two individual networks or SR and object detection. This can be improved by stacking two neural networks together to form a joint network. This will allow one to train an end-to-end network to learn how to perform SR to improve object detection. The state-of-the-art network for SR is typically trained under a generative adversary training framework [46, 54, 79], which is different from how an object detection network is typically trained. Combining both networks would require us to incorporate the object detection network into the generative adversary training framework and find a way to back-propagate the loss to update the weights. It would also require us to modify the output of the network to produce both scaled video (for rendering) and object detection results. We leave this as our future work.

Training overhead of predictive models. Gather user feedback to train the QoE model can be expensive. In this work, we employ clustering algorithms to minimize user involvement. This cost can be further reduced by using active learning [65] to only collect training samples, which are likely to improve the learnt model.

Model interpretability. Machine learning techniques, in general, have the issue of being a black box. This problem is just as true for our approach. A possible approach to gain insight into the model's decision making process is to train a simpler but interpretable model approximate the predictions of the underlying black-box model [77].

Model compression techniques. This work focuses on efficient computation offloading for MAR. Actor is designed to trade user-perceived video quality for smooth video play, accurate object detection, and high-quality rendering. It works by offloading computation-intensive tasks to a remote cloud server. There is an extensive body of work on reducing the running time and resource requirement of deep learning models on embedded devices, often through model compression techniques [52, 119]. These methods are orthogonal to our work. Our future work will look into how to integrate our approach with techniques for reducing deep learning model execution on embedded devices to exploit finer-grained computation offloading.

Out of distribution problem. Our trained predictive models can be applied to video contents that were not seen during the training phase. However, if the characteristics of the input video are significantly different from the training samples, the predictive models may give poor prediction performance. This is an issue known as "out-of-distribution" by the machine learning community [33, 51]. Therefore, our approach can benefit from techniques for detecting and improving ageing machine learning models [41, 43, 98].

8 RELATED WORK

Our work builds upon the following past foundations but is different from each.

Mobile AR optimization. ARCore [28] and ARKit [2] are two mainstream MAR platforms, but they do not support object detection due to the high computation requirement. Recent studies show that the computation resources of smartphones are inadequate to meet the processing latency requirement of MAR [13, 118]. Several works focus on running MAR locally on mobile devices by performing lightweight models or leveraging local inertial sensors to recognize and track multiple objects without using deep object detection models [1, 14]. However, these approaches are designed for lightweight AR on low-resolution rendering, but cannot meet the computation requirement of 4K MAR. Offloading frameworks like Glimpse offload the trigger frames to the cloud for object detection [16]. Although Glimpse can offer 30 FPS for a 640×480 resolution, it requires more significant computational resources on the mobile device for object tracking, leaving fewer resources for rendering high-quality AR. The work presented in [52] reduces the offloading latency by only uploading macroblocks for regions of interest for object detection and tracking. However, it fails to meet the real-time requirement when processing 4K videos and leaves the computation-intensive rendering task on the user device. None of these methods supports real-time MAR for 4K videos, nor models the impact of video content and network conditions on the user-perceived experience.

Mobile vision offloading. Due to the limited processing capacity on mobile devices, moving computation-intensive tasks from the mobile device to the cloud or edge infrastructures is a feasible way to enable continuous vision analytic[11, 52, 113, 119]. For example, DeepDecision [70] and MCDNN [29] dynamically determine if a DNN should be executed on the local mobile or a remote server. They are designed to optimize constraints like accuracy and latency for the current networking condition. LAVEA [112] distributes computation tasks among multiple edge nodes to provide low-latency video analytics. To minimize resource utilization while meeting the user-specified accuracy requirement, Chameleon [38] and VideoStorm [115] reduce the backend computation costs by profiling different configurations of pipeline knobs (e.g., video resolution, frame sampling rate, etc.). These systems are largely complementary to Actor. They can be used to further tune the SR configuration based on the accuracy requirement of object detection to improve the TAT. Moreover, Reducto [49] and Glimpse [16] perform selective data offloading based on the query accuracy and video content to minimize the latency. EAR [52] uses a motion vector based object tracking to adaptively offload those regions of interest. However, most existing solutions use low-resolution images through the entire pipeline, making the inference task lightweight. However, EAR lose the opportunity to leverage the rich content of high resolution (e.g., 2K or 4K) images/frames. Our work builds upon these past foundations by targeting 4K MAR. Unlike prior work, Actor considers the impact of video contents and network environments on user-perceived experience.

Deep learning acceleration. Deep learning techniques have shown astonishing success in various tasks, including object detection [21, 31, 71, 76]. Efforts have been devoted to accelerating the inference of deep learning models on mobile devices through model compression techniques[17, 36, 56, 69, 76]. Actor leverages the recent advances in deep inference acceleration to speed up the DNN model execution time on the remote server to reduce the latency.

Super resolution. Image and video super-resolution is the process of estimating a high resolution version of a low resolution image or video sequence, and various model architectures have been actively studied in recent years [30, 35, 101, 120].As the large memory requirements of advanced SR models, they are operated in the cloud generally. Prior studies [44, 47, 53] also seek to deploy SR models on mobile devices by compressing the large neural networks or relying on custom hardware. However, the existed on-device SR methods still take over 500 ms on each frame and can not deal with 4K resolution. Study in [44] is the first to achieve real-time performance from 2K to 4K, while the algorithm was deployed on the dedicated hardware (FPGA). Our work

utilizes DNN-based SR techniques [92] to reconstruct much of the information required for MAR. Unlike prior works, ACTOR is the first offloading framework to leverage the HECV encoding scheme to adaptively choose a few key video frames to perform SR. Our work utilizes DNN-based SR techniques [101] to reconstruct much of the information required for MAR.

Predictive modeling. Machine learning has been used to model power consumption [34], task scheduling [74, 93, 105] of mobile systems, program tuning and modeling in general [12, 15, 18, 19, 58, 65, 97, 100, 102, 104, 106, 107, 110, 111, 117]. Our work is the first to employ machine learning for 4K MAR offloading. Furthermore, our work tackles an outstanding problem of porting a model to a new computing environment. Transfer learning was recently used for wireless sensing [116] through randomly chosen samples. ACTOR improves [116] by carefully choosing representative tracing examples for transfer learning.

9 CONCLUSIONS

We have presented ACTOR, a novel computation offloading framework for 4K mobile argument reality (MAR). ACTOR leverages deep learning-based super-resolution techniques to trade the user-perceived experience for low latency and real-time processing. ACTOR dynamically chooses an appropriate resolution to downscale the mobile video feed to reduce the uploading latency. The downscaling resolution is determined based on the video content, the network environment and user expectation of the video quality and latency. ACTOR employs machine learning to develop an adaptive offloading approach, where decision models are first trained offline, and the trained model can be applied to any video content and network environments. We evaluate ACTOR by applying it to over 2,000 4K video clips across two typical mobile network environments. Experimental results show that ACTOR consistently outperforms competitive schemes for meeting real-time processing requirements while maintaining high user-perceived video quality.

ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation China (NSFC) under grant agreements 61902229 and 61872294; and an International Science and Technology Cooperation Grant of Shannxi Province (grant agreement 2020KW-006). For any correspondence, please contact Zheng Wang (E-mail: z.wang5@leeds.ac.uk).

REFERENCES

[1] Kittipat Apicharttrisorn, Xukan Ran, Jiasi Chen, Srikanth V Krishnamurthy, and Amit K Roy-Chowdhury. 2019. Frugal following: Power thrifty object detection and tracking for mobile augmented reality. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*. 96–109.
[2] Apple. 2020. Apple ARKit. https://developer.apple.com/arkit/.
[3] ATToolKit. 2017. ARToolKit. http://www.artoolkit.org
[4] Christian Bauckhage. 2015. K-means clustering is matrix factorization. *arXiv* (2015).
[5] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* (2020).
[6] Christoph Borel-Donohue and S Susan Young. 2019. Image quality and super resolution effects on object recognition using deep neural networks. In *Proceedings of the Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, Vol. 11006. 110061M.
[7] El Hocine Bouzidi, Duc-Hung Luong, Abdelkader Outtagarts, Abdelkrim Hebbar, and Rami Langar. 2018. Online-based learning for predictive network latency in software-defined networks. In *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.
[8] G. Bradski. 2000. The OpenCV Library. (2000).
[9] CableLabs. 2020. 4K VIDEO. https://www.cablelabs.com/4k.
[10] Gines Campagna. 2021. All The Best Phones With 4K Display Listed. https://www.thephonetalks.com/all-best-phones-with-4k-display/.
[11] Wei Chang, Yang Xiao, Wenjing Lou, and Guochu Shou. 2020. Offloading Decision in Edge Computing for Continuous Applications Under Uncertainty. *IEEE Transactions on Wireless Communications* 19, 9 (2020), 6196–6209.

[12] Donglin Chen, Jianbin Fang, Shizhao Chen, Chuanfu Xu, and Zheng Wang. 2019. Optimizing sparse matrix–vector multiplications on an armv8-based many-core architecture. *International Journal of Parallel Programming* 47, 3 (2019), 418–432.
[13] Huixiang Chen, Yuting Dai, Hao Meng, Yilun Chen, and Tao Li. 2018. Understanding the characteristics of mobile augmented reality applications. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 128–138.
[14] Kaifei Chen, Tong Li, Hyung-Sin Kim, David E Culler, and Randy H Katz. 2018. MARVEL: Enabling Mobile Augmented Reality with Low Energy and Low Latency. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. 292–304.
[15] Shizhao Chen, Jianbin Fang, Donglin Chen, Chuanfu Xu, and Zheng Wang. 2018. Adaptive optimization of sparse matrix-vector multiplication on emerging many-core architectures. In *2018 IEEE 20th International Conference on High Performance Computing and Communications*. IEEE, 649–658.
[16] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 155–168.
[17] Matthieu Courbariaux et al. 2015. BinaryConnect: training deep neural networks with binary weights during propagations. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*. 3123–3131.
[18] Chris Cummins, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. 2017. End-to-end deep learning of optimization heuristics. In *2017 26th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, 219–232.
[19] Chris Cummins, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. 2017. Synthesizing benchmarks for predictive modeling. In *2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 86–99.
[20] Simon Da Silva, Sonia Ben Mokhtar, Stefan Contiu, Daniel Négru, Laurent Réveillère, and Etienne Rivière. 2019. Privatube: Privacy-preserving edge-assisted video streaming. In *Proceedings of the 20th International Middleware Conference*. 189–201.
[21] Jifeng Dai et al. 2016. R-FCN: object detection via region-based fully convolutional networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 379–387.
[22] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. 2019. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11065–11074.
[23] deepframe. 2021. deepframe. https://www.realfiction.com/solutions/deepframe.
[24] Wolfgang Ertel. 1994. On the definition of speedup. In *International Conference on Parallel Architectures and Languages Europe*.
[25] FFmpeg. 2020. FFmpeg. https://ffmpeg.org/.
[26] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.
[27] Dream Glass. 2021. Dream Glass 4K/4K Plus. https://www.dreamworldvision.com/.
[28] Google. 2020. Google ARCore. https://developers.google.com/ar/.
[29] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. 2016. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. 123–136.
[30] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. 2019. Recurrent back-projection network for video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3897–3906.
[31] Kaiming He et al. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.
[32] Stephen Hemminger et al. 2005. Network emulation with NetEm. In *Linux conf au*. 18–23.
[33] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).
[34] Mohammad Ashraful Hoque et al. 2015. Modeling, Profiling, and Debugging the Energy Consumption of Mobile Devices. *ACM Computing Surveys (CSUR)* 48, 3 (2015), 1–40.
[35] Zheng Hui, Xiumei Wang, and Xinbo Gao. 2018. Fast and accurate single image super-resolution via information distillation network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 723–731.
[36] Forrest Iandola. 2017. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv: Computer Vision and Pattern Recognition* (2017).
[37] ITU-T. 2019. H.265 : High efficiency video coding. https://www.itu.int/rec/T-REC-H.265.
[38] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 253–266.
[39] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. 2018. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *CVPR*. 3224–3232.
[40] Audrius Jurgelionis, Jukka-Pekka Laulajainen, Matti Hirvonen, and Alf Inge Wang. 2011. An empirical study of netem network emulation functionalities. In *2011 Proceedings of 20th international conference on computer communications and networks (ICCCN)*. IEEE, 1–6.
[41] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. 2016. Fine-tuning deep neural networks in continuous learning scenarios. In *Asian Conference on Computer Vision*. Springer, 588–605.

[42] Youssef Khazbak, Junpeng Qiu, Tianxiang Tan, and Guohong Cao. 2020. TargetFinder: A Privacy Preserving System for Locating Targets through IoT Cameras. *ACM Transactions on Internet of Things* (2020).
[43] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. 2020. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490* (2020).
[44] Yongwoo Kim, Jae-Seok Choi, and Munchurl Kim. 2018. A real-time convolutional neural network for super-resolution on FPGA with applications to 4K UHD 60 fps video services. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 8 (2018), 2521–2534.
[45] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. 2020. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. *IEEE Transactions on Mobile Computing* 19, 7 (2020), 1586–1602.
[46] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.
[47] Royson Lee, Stylianos I Venieris, Lukasz Dudziak, Sourav Bhattacharya, and Nicholas D Lane. 2019. Mobisr: Efficient on-device super-resolution through heterogeneous mobile processors. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.
[48] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).
[49] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 359–376.
[50] Zhi Li, Kyle Swanson, Christos Bampis, Lukás Krasula, and Anne Aaron. 2020. Toward a Better Quality Metric for the Video Community. https://netflixtechblog.com/toward-a-better-quality-metric-for-the-video-community-7ed94e752a30.
[51] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690* (2017).
[52] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.
[53] Xin Liu, Yuang Li, Josh Fromm, Yuntao Wang, Ziheng Jiang, Alex Mariakakis, and Shwetak Patel. 2021. SplitSR: An End-to-End Approach to Super-Resolution on Mobile Devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–20.
[54] Alice Lucas, Santiago Lopez-Tapia, Rafael Molina, and Aggelos K Katsaggelos. 2019. Generative adversarial networks and perceptual losses for video super-resolution. *IEEE Transactions on Image Processing* 28, 7 (2019), 3312–3327.
[55] Zhengyi Luo, Yan Huang, Xiangwen Wang, Rong Xie, and Li Song. 2019. VMAF Oriented Perceptual Optimization for Video Coding. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
[56] Ningning Ma et al. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*. 116–131.
[57] Xiaojing Ma, Bin Zhu, Tao Zhang, Sixing Cao, Hai Jin, and Deqing Zou. 2018. Efficient privacy-preserving motion detection for HEVC compressed video in cloud video surveillance. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 813–818.
[58] Vicent Sanz Marco, Ben Taylor, Zheng Wang, and Yehia Elkhatib. 2020. Optimizing deep learning inference on embedded systems through adaptive model selection. *ACM Transactions on Embedded Computing Systems (TECS)* 19, 1 (2020), 1–28.
[59] MathWorks. 2020. Nearest Neighbor, Bilinear, and Bicubic Interpolation Methods. https://www.mathworks.com/help/vision/ug/interpolation-methods.html.
[60] A. Mercat et al. 2020. UVG dataset: 50/120fps 4K sequences for video codec analysis and development. In *ACM Multimedia Syst. Conf.*
[61] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440* (2016).
[62] Netflix. 2018. VMAF: The Journey Continues. https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12.
[63] Netflix. 2020. VMAF - Video Multi-Method Assessment Fusion. https://github.com/Netflix/vmaf.
[64] NVIDIA. 2020. Tencent Games Partners with NVIDIA to Launch START Cloud Gaming Service. https://nvidianews.nvidia.com/news/tencent-games-partners-with-nvidia-to-launch-start-cloud-gaming-service.
[65] William F Ogilvie, Pavlos Petoumenos, Zheng Wang, and Hugh Leather. 2017. Minimizing the cost of iterative compilation with active learning. In *2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 245–256.
[66] Marta Orduna, César Díaz, Lara Muñoz, Pablo Pérez, Ignacio Benito, and Narciso García. 2019. Video multimethod assessment fusion (VMAF) on 360vr contents. *IEEE Transactions on Consumer Electronics* 66, 1 (2019), 22–31.
[67] ITU-T Recommendation P.910. 2007. Methodology for the subjective assessment of video quality in multimedia applications. (2007).
[68] Yanwei Pang, Jiale Cao, Jian Wang, and Jungong Han. 2019. JCS-Net: Joint classification and super-resolution network for small-scale pedestrian detection in surveillance images. *IEEE Transactions on Information Forensics and Security* 14, 12 (2019), 3322–3331.

[69] Qing Qin, Jie Ren, Jialong Yu, Hai Wang, Ling Gao, Jie Zheng, Yansong Feng, Jianbin Fang, and Zheng Wang. 2018. To Compress, or Not to Compress: Characterizing Deep Learning Model Compression for Embedded Inference. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications*. 729–736.
[70] Xukan Ran, Haolianz Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1421–1429.
[71] Rajeev Ranjan et al. 2017. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE PAMI* (2017).
[72] Reza Rassool. 2017. VMAF reproducibility: Validating a perceptual practical video quality metric. In *2017 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)*. IEEE, 1–2.
[73] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *arXiv* (2018).
[74] Jie Ren, Xiaoming Wang, Jianbin Fang, Yansong Feng, Dongxiao Zhu, Zhunchen Luo, Jie Zheng, and Zheng Wang. 2018. Proteus: Network-aware web browsing on heterogeneous mobile systems. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 379–392.
[75] Jie Ren, Lu Yuan, Petteri Nurmi, Xiaoming Wang, Miao Ma, Ling Gao, Zhanyong Tang, Jie Zheng, and Zheng Wang. 2020. Camel: Smart, adaptive energy optimization for mobile web interactions. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 119–128.
[76] Shaoqing Ren, Kaiming He, Ross B Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*.
[77] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.
[78] Michael S Ryoo, Brandon Rothrock, Charles Fleming, and Hyun Jong Yang. 2017. Privacy-preserving human activity recognition from extreme low resolution. In *Thirty-First AAAI Conference on Artificial Intelligence*.
[79] Mehdi SM Sajjadi, Bernhard Scholkopf, and Michael Hirsch. 2017. Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of the IEEE International Conference on Computer Vision*. 4491–4500.
[80] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.
[81] Volker Seeker, Pavlos Petoumenos, Hugh Leather, and Björn Franke. 2014. Measuring qoe of interactive workloads and characterising frequency governors on mobile devices. In *2014 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 61–70.
[82] Jacob Shermeyer and Adam Van Etten. 2019. The effects of super-resolution on object detection performance in satellite imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.
[83] Wenzhe Shi, Jose Caballero, Ferenc Huszár, 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1874–1883.
[84] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[85] Li Song, Xun Tang, Wei Zhang, Xiaokang Yang, and Pingjian Xia. 2013. The SJTU 4K video sequence dataset. In *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 34–35.
[86] Speedtest. 2019. How 5G is Changing the Global Mobile Landscape. https://www.speedtest.net/insights/blog/5g-changing-global-mobile-landscape-2019/.
[87] SpeedTest. 2020. THE STATE OF MOBILE 5G IN THE UNITED KINGDOM. https://www.speedtest.net/insights/blog/5g-united-kingdom-2019/.
[88] Splunk. 2020. Splunk AR. https://www.splunk.com/.
[89] Kashyap Kammachi Sreedhar, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. 2016. Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 583–586.
[90] statista. 2020. Augmented reality market size worldwide 2017-2025. https://www.statista.com/statistics/897587/world-augmented-reality-market-value/.
[91] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668.
[92] Rami Tabari. 2021. The best 4K laptops in 2021. https://www.laptopmag.com/articles/best-4k-laptops.
[93] Ben Taylor, Vicent Sanz Marco, and Zheng Wang. 2017. Adaptive optimization for OpenCL programs on embedded heterogeneous systems. *ACM SIGPLAN Notices* 52, 5 (2017), 11–20.
[94] Ben Taylor, Vicent Sanz Marco, Willy Wolff, Yehia Elkhatib, and Zheng Wang. 2018. Adaptive deep learning model selection on embedded systems. *ACM SIGPLAN Notices* 53, 6, 31–43.
[95] TeamViewer. 2020. Augmented Reality-Powered Remote Support. https://www.teamviewer.com/.
[96] TensorFlow. 2020. TensorFlow Lite. https://www.tensorflow.org/lite/.

[97] Georgios Tournavitis, Zheng Wang, Björn Franke, and Michael FP O'Boyle. 2009. Towards a holistic approach to auto-parallelization: integrating profile-driven parallelism detection and machine-learning based mapping. *ACM Sigplan notices* 44, 6 (2009), 177–187.
[98] Vladimir Vovk, Valentina Fedorova, Ilia Nouretdinov, and Alexander Gammerman. 2016. Criteria of efficiency for conformal prediction. In *Symposium on conformal and probabilistic prediction with applications*. Springer, 23–39.
[99] Ermias Andargie Walelgne, Alemnew Sheferaw Asrese, Jukka Manner, Vaibhav Bajpai, and Jörg Ott. 2020. Understanding data usage patterns of geographically diverse mobile users. *IEEE Transactions on Network and Service Management* (2020).
[100] Huanting Wang, Guixin Ye, Zhanyong Tang, Shin Hwei Tan, Songfang Huang, Dingyi Fang, Yansong Feng, Lizhong Bian, and Zheng Wang. 2020. Combining graph-based learning with automated data collection for code vulnerability detection. *IEEE Transactions on Information Forensics and Security* 16 (2020), 1943–1958.
[101] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. 2019. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.
[102] Zheng Wang et al. 2014. Integrating profile-driven parallelism detection and machine-learning-based mapping. *ACM TACO* (2014).
[103] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
[104] Zheng Wang, Dominik Grewe, and Michael FP O'boyle. 2014. Automatic and portable mapping of data parallel programs to opencl for gpu-based heterogeneous systems. *ACM Transactions on Architecture and Code Optimization (TACO)* 11, 4 (2014), 1–26.
[105] Zheng Wang and Michael O'Boyle. 2018. Machine Learning in Compiler Optimization. *Proc. IEEE* (2018).
[106] Zheng Wang and Michael F.P. O'Boyle. 2009. Mapping Parallelism to Multi-cores: A Machine Learning Based Approach. In *PPoPP '09*.
[107] Yuan Wen, Zheng Wang, and Michael FP O'boyle. 2014. Smart multi-task scheduling for OpenCL programs on CPU/GPU heterogeneous platforms. In *2014 21st International conference on high performance computing (HiPC)*. IEEE, 1–10.
[108] WondAR. 2020. The WondAR is a world-class 3D augmented reality system specialized for large screens. https://www.thewondar.com/.
[109] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. 2018. DeepCache: Principled cache for mobile deep vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 129–144.
[110] Guixin Ye, Zhanyong Tang, Shin Hwei Tan, Songfang Huang, Dingyi Fang, Xiaoyang Sun, Lizhong Bian, Haibo Wang, and Zheng Wang. 2021. Automated conformance testing for JavaScript engines via deep compiler fuzzing. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. 435–450.
[111] Guixin Ye, Zhanyong Tang, Huanting Wang, Dingyi Fang, Jianbin Fang, Songfang Huang, and Zheng Wang. 2020. Deep program structure modeling through multi-relational graph-based learning. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*. 111–123.
[112] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. 2017. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–13.
[113] Ayman Younis, Brian Qiu, and Dario Pompili. 2020. Latency-aware Hybrid Edge Cloud Framework for Mobile Augmented Reality Applications. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.
[114] Tao Zhan, Kun Yin, Jianghao Xiong, Ziqian He, and Shin-Tson Wu. 2020. Augmented reality and virtual reality displays: Perspectives and challenges. *Iscience* (2020), 101397.
[115] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. 2017. Live video analytics at scale with approximation and delay-tolerance. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*. 377–392.
[116] Jie Zhang, Zhanyong Tang, Meng Li, Dingyi Fang, Petteri Nurmi, and Zheng Wang. 2018. CrossSense: Towards cross-site and large-scale WiFi sensing. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 305–320.
[117] Peng Zhang, Jianbin Fang, Tao Tang, Canqun Yang, and Zheng Wang. 2018. Auto-tuning streamed applications on intel xeon phi. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 515–525.
[118] Wenxiao Zhang, Bo Han, and Pan Hui. 2017. On the networking challenges of mobile augmented reality. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*. 24–29.
[119] Wenxiao Zhang, Sikun Lin, Farshid Hassani Bijarbooneh, Hao Fei Cheng, and Pan Hui. 2017. Cloudar: A cloud-based framework for mobile augmented reality. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*. 194–200.
[120] Yulun Zhang et al. 2018. Image Super-Resolution Using Very Deep Residual Channel Attention Networks. In *Proceedings of the European conference on computer vision (ECCV)*. 286–301.
[121] Agustin Zuniga, Huber Flores, Eemil Lagerspetz, Petteri Nurmi, Sasu Tarkoma, Pan Hui, and Jukka Manner. 2019. Tortoise or hare? quantifying the effects of performance on mobile app retention. In *The World Wide Web Conference*. 2517–2528.