AdaDSR: Adaptive Configuration Optimization for Neural Enhanced Video Analytics Streaming

Sheng Cen, Miao Zhang, Yifei Zhu, *Member, IEEE,* and Jiangchuan Liu, *Fellow, IEEE*

***Abstract*—Neural-based super-resolution (SR) has achieved great success in enhancing image or video quality, creating new opportunities for building bandwidth-efficient and high-accuracy video analytics (VA) systems. Intuitively, with the help of SR techniques, cameras only need to send downsampled low-quality frames to the server in a canonical edge-assisted video analytics framework. The server-side SR model then upscales the quality of received frames for the subsequent video analytics tasks, incurring thus substantially reduced bandwidth consumption. Nonetheless, as revealed by our measurement results on real-world video clips, higher delivery quality does not necessarily lead to higher analysis accuracy. This motivates us to study the content-adaptive downsampling and upscaling ratio selection problem for video analytics streaming. We propose a SR-based video analytics framework, named *AdaDSR* that can dynamically select the optimal downsampling and upscaling ratios so that the system utility can be maximized. *AdaSDR* is configured to balance the tradeoffs among accuracy, network cost, and computational cost. It further leverages the temporal consistency of videos to skip trivial decisions so that the camera's processing overhead can be reduced. Experiments on real-world video datasets demonstrate that AdaDSR can improve the average utility by 7.2%-18.4% when compared with state-of-the-art approaches under diverse video scenes.**

***Index Terms*—super-resolution, video analytics, edge computing, configuration optimization**

I. Introduction

With the development of deep neural network (DNN)-based computer vision (CV) models and an increasing number of camera deployments, recent years have witnessed an explosive growth of video analytics tasks, such as object detection, and semantic segmentation [1] [2] [3]. Currently, most front-end devices stream their captured videos to resource-rich cloud servers to achieve high analysis accuracy. However, cloud servers typically reside far away from these front-end devices, which introduces significant network consumption for large-volume video transmission. Streaming filtered frames to the cloud server can affect the inference accuracy due to the loss of image details. On the other hand, analyzing videos directly on front-end devices could avoid the network cost, however, it cannot support high-accuracy video analytics at need due to the limited computational resources of most front-end devices. Therefore, it is urgent to find a solution to achieve high-accurate video analytics with an affordable network cost.

Super-resolution (SR) is an emerging CV technique aiming at recovering the resolution of low-quality images in order to achieve better visual quality [4]. It upsamples low-resolution images to an expected high resolution, by using neural networks like SRCNN [5], VDSR [6], ESPCN [7], etc., which are pre-trained to minimize the loss function of high/low resolution pairs of images in the given training set. Previous studies on SR mainly focus on improving human-perceived visual quality for human-oriented video streaming applications [8] [9]. In recent years, with the rise of machine-centric streaming systems, where videos are streamed and consumed directly by machines that execute intelligent models, analytics-aware SR models are further developed to improve inference accuracy. These models embed analytics-related objectives into the loss function and have shown promising results [10] [11].

However, existing analytics-aware SR systems for video analytics tasks are far from optimal. First, our later measurement study indicates that the SR operation dominates the computational cost and is affected by both the input frame size and the SR upscaling ratio. We further reveal that the inference accuracy may not drop when using an SR model with a lower upscaling ratio. Therefore, when the downsampled frames are recovered to their original quality, as current solutions do [10] [11] [12], unnecessary computation exists without bringing extra benefit in an accuracy improvement. Second, many works [12] choose to maintain a Pareto-frontier solution in the space of important system dimensions, e.g., accuracy and latency, without considering the effect of video content. Although it utilizes the temporal consistency of videos, it cannot adapt to fine-grained video content dynamics.

To overcome the aforementioned limitations, we propose *AdaDSR*, a content-adaptive configuration selection framework that fully exploits the potential of SR techniques in helping video analytics streaming tasks. *AdaDSR* intelligently adjusts the frame downsampling and SR upscaling ratios, referred to as configurations, to balance the tradeoff among inference accuracy, network cost, and computational cost. It is composed of a feature extractor, a network cost estimator, and a configuration selector. The feature extractor is constructed based on a CNN backbone to extract the frame-level foreground objects' high-level features; the network cost estimator is designed to estimate the video segment transmission volume based on a random forest regressor. The configuration selector predicts the optimal configuration for video frames based on

S. Cen and Y. Zhu are with the UM-SJTU Joint Institute, Shanghai Jiao Tong University, China. Y. Zhu is also with Cooperative Medianet Innovation Center (CMIC), Shanghai Jiao Tong University, China.

M. Zhang and J. Liu are with the School of Computing Science, Simon Fraser University, Canada.

Email: {cens98, yifei.zhu}@sjtu.edu.cn, mza94@sfu.ca, jcliu@cs.sfu.ca
Corresponding author: Yifei Zhu

the results from previous modules. When deployed online, *AdaDSR* further utilizes a frame filtering strategy by leveraging temporal consistency of videos to further save the network and computational cost. In summary, our contributions can be summarized as follows:

- We reveal the complex relationship between downsampling and SR upscaling ratio when helping the video analytics tasks. This motivates us to decouple the decisions of adjusting these two ratios to fully unleash SR's potential in system improvement.
- We formulate the SR-based video analytics problem as a utility maximization problem by controlling the configuration setting of SR models. Our comprehensive utility function considers the analytics performance, network cost, and computation cost, which are crucial factors to video analytics.
- We design a novel utility-driven configuration selector to learn the complex relationship between video content and the optimal configuration. An online frame-skipping mechanism is further designed to reduce the camera-side processing overhead.
- Extensive experiments on real-world video clips show that our method can improve the performance by 7.2% to 18.4% on average, compared with state-of-the-art baselines.

The remainder of this paper is organized as follows. We conduct a literature review on SR-based video analytics systems in Section II. Section III presents the measurement results and the resulting motivation. Section IV presents the system models and problem formulation. Section V proposes our design in making the adaptive decisions for selecting SR configuration. Section VI presents the evaluation of methods. We finally draw a conclusion in Section VII.

II. Related Work

A. SR for Improving Quality of Experience (QoE)

SR is most widely used to improve the frame image quality and thus improving users' visual experience for video watching. In the video streaming scenario, the bandwidth constraint and the need for higher visual quality are the major motivation for SR-based streaming systems. Many works design or retrain an SR network to enhance the quality (always evaluated as reconstructed error) of the images under strict network requirements. For example, a light-weighted SR is designed in [13] to reconstruct the image and reduce video stalling time in the scenario of HTTP video adaptive streaming [14] [13]. Liu *et al.* [15] deploy actor-critic reinforcement learning to jointly decide the configuration of light-weighted and large-scale SR models in the mobile devices and edge servers respectively, to improve users' QoE. Zhang *et al.* [8] select the frame bitrate and SR reconstruction factor to reduce the frame reconstruction loss as well as the rebuffering time. SR is also used in [9] [16] to improve the client-side video resolution and reduce the stalling time under bandwidth constraints for 360-degree video streaming. SR technique is also employed [17] in the traditional edge caching system where users could have access to high-quality video with a higher data hit ratio through adaptive bitrate (ABR) algorithms, considering the bandwidth fluctuation. However, these above works do not necessarily improve video analytics tasks, as shown in Sec. II-B, despite their ability to enhance the human-oriented visual quality.

B. SR for Enhancing Video Analytics

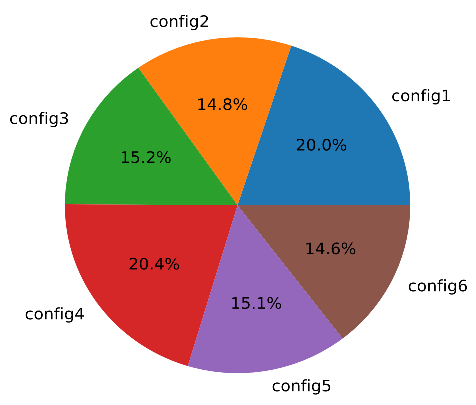
Recently, many researchers have been dedicated to applying the SR technique to machine-centric video streaming tasks, like object detection [18] [19], semantic segmentation [20] [21], etc. Shan *et al.* [22] utilize online learning to adaptively train the SR models under various network requirements. Aguilar *et al.* [23] use SR to improve the small-object detection accuracy in video clips. Wang *et al.* [24] train customized object detection models and evaluate the inference accuracy with different upscaling factors. Mu *et al.* [25] improve the SR process by replacing the motion estimation and compensation with a pyramid deformable CNN network to improve the overall inference accuracy. Temporal consistency inter-frame learning is applied [26] in SR to make the reconstructed videos consistent.

In addition to these efforts from the computer vision field, a growing number of system researchers have started to apply SR to increase the inference accuracy of video analytics tasks under the constraint of network conditions. Wang *et al.* [12] train the SR model with different upscaling factors to increase the average accuracy. It constructs a Pareto frontier between frame compression/downsampling rate and inference accuracy to optimize the configuration selection under the bandwidth constraint, by applying the controlling mechanism designed in AWStream [27]. Zhang *et al.* [10] follow the work and further improve the tail inference accuracy. They are concerned more about the objects which are difficult to detect. However, all of the works above neither consider the SR model computational cost, nor decouple the selection of frame downsampling and SR upscaling ratio, which wastes a lot of server-side computational resources and lead to sub-optimal analytics results.

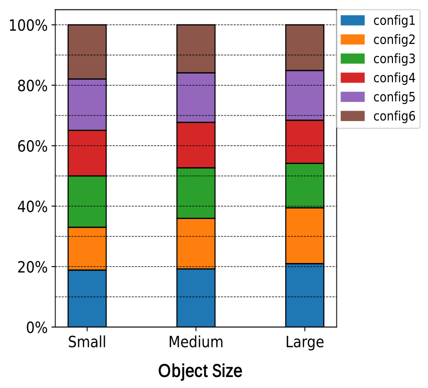
III. Motivation

In this section, we explore real-world video datasets, and reveal the non-trivial relationship between the inference accuracy and the configuration selected for utilizing super- resolution. We select six different configurations to analyze three video clips [28] [29] [30] (downloaded with 720p, about 3,600 frames are analyzed) captured by surveillance cameras in the US, posted on YouTube. Each configuration refers to a specific joint selection of segment downsampling and SR upscaling ratio as shown in Table I. For example, configuration 2 means that the camera first downsamples the frames from 720*1280 to 180*320 (downsampling ratio 4×), then the server upscales the downsampled frames to 360*640 (upscaling ratio 2×).

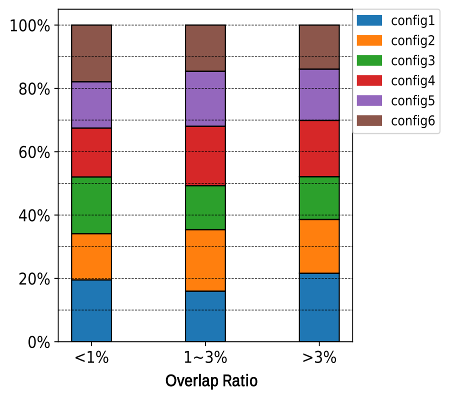
The analyzed video frames cover a variety of real-world scenarios (e.g. Town Square, People Square, and crossroads) and illuminations (e.g. day and night). The three most common objects (cars, traffic lights, and pedestrians) are chosen as the detection targets. We apply the pre-trained SR models introduced in [31], which is widely used in the CV community



(a)



(b)



(c)

Fig. 1. Probability distribution of each configuration hitting the highest accuracy for each frame. (a) Overall distribution; (b) Distribution with regard to object size; (c) Distribution with regard to object overlap ratio.

TABLE I
SR INFERENCE TIME OF TESTED CONFIGURATIONS (UPSCALING RATIO 1× MEANS THAT NO SR MODELS ARE USED, AND THE CORRESPONDING TIME IS MARKED AS '-'.)

# Config.	Downsampling	Upscaling	T_{SR} (ms)
1	4×	1×	-
2	4×	2×	763 ± 26
3	4×	4×	1011 ± 58
4	3×	1×	-
5	3×	2×	1089 ± 15
6	3×	4×	1539 ± 87

due to their invulnerability to alternative blurring kernels. We use yolov7 [32] and mAP@0.75 as our object detection model and accuracy metric, respectively. Table I also lists out the average SR inference time of a single frame[1] in the last column. We summarize the following insights observed from the measurement study:

(1) Different configurations lead to different network and server-side computational costs. Table I shows that when we downsample the frame to a smaller resolution and apply an SR model with a larger upscaling ratio, it takes longer to run the SR inference model. For example, config. 6 costs the highest running time, since it operates on a larger frame resolution, and uses a higher SR upscaling ratio. Therefore, various configurations could lead to dramatically different server-side GPU computational costs, represented by SR inference time. Moreover, after downsampling the video segments (each comprises 60 frames) in our dataset to different resolutions, we find that the average data sizes of transferred video segments for downsampling ratio of 4×, 3×, and 2× are about 220 KB, 296 KB, and 352 KB, respectively. Therefore, streaming frames with a higher resolution over the network incurs a higher data transfer overhead and thus higher network costs.

[1]The experiment was tested on Ubuntu, with dual Tesla V100 32GB GPU cards, an Intel Xeon 6226 2.7 GHz CPU card, and 8GB memory.

(2) Separately considering the frame downsampling and SR upscaling ratio is necessary. Existing works [11] [10] tend to use the same ratio for downscaling and upscaling. While it seems intuitive that frames should be restored to their original visual quality to achieve the maximum accuracy performance, our measurement indicates the contrary, namely, restoring the image quality to a level lower than its original quality can achieve the highest accuracy in some cases. Fig. 1(a) shows the overall frame-level distribution of each configuration hitting the highest accuracy. We find that consistent frame downsampling and upscaling ratio does not always yield the highest accuracy (e.g., config. 3 does not always lead to the highest accuracy among config. 1, 2, and 3). About 84.8% of frames achieve the highest accuracy when using different ratios for downsampling and upscaling, necessitating adaptive separate optimization. Furthermore, when choosing the best and worst configurations, we get mAP@0.75 of 71.3% and 48.9%, respectively, revealing a huge impact of the chosen configuration on the detection accuracy.

When it comes to jointly considering the inference accuracy, network cost, and computational cost, we find that decoupling the selection of downsampling and SR upscaling ratio has more benefits. Config. 2 may outperform config. 3 (the one with the same downsampling and SR upscaling ratio) for both a higher inference accuracy sometimes, and a much lower computational cost. Similarly, a decoupling selection may lead to a higher accuracy and a much lower network cost.

The above findings reveal that a separate tuning of frame downsampling and SR upscaling ratios is necessary for optimizing inference accuracy, computational cost, and network consumption.

(3) Simple vision features are insufficient to unearth the optimal combination of downsampling and upscaling ratio. Recent related works on video analytics, such as [33], choose low-level features, such as foreground object size, to be the hint for configuration selection. But it remains a question on whether such features could provide useful information on the selection of our configuration knobs, i.e. frame downsampling

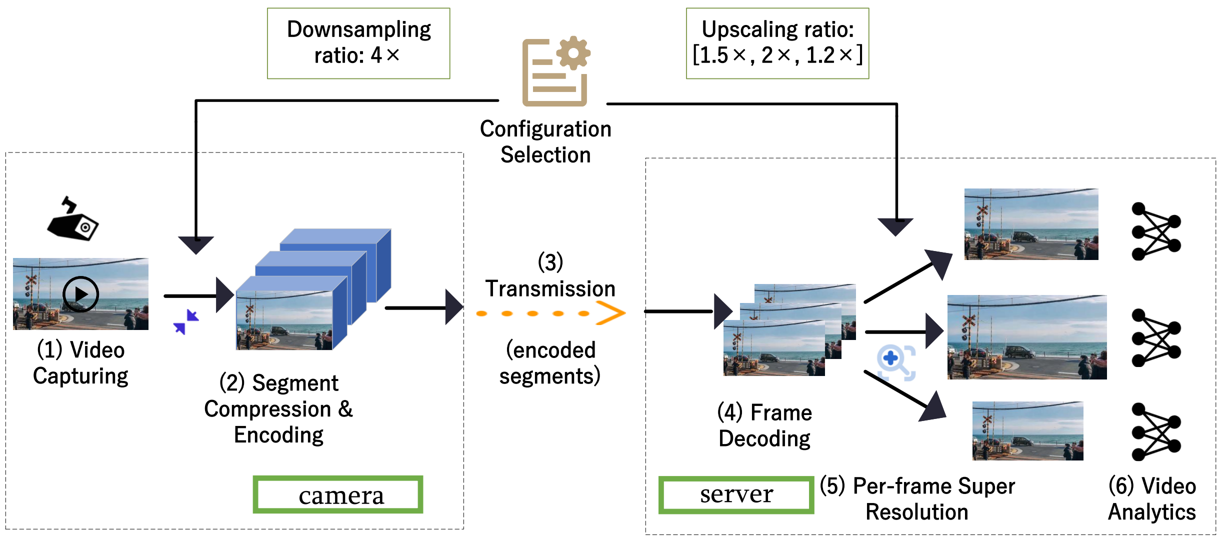


Fig. 2. SR-enhanced video analytics pipeline.

and SR upscaling ratio. We thus select two commonly used vision features, namely, object size and overlap ratio, to examine the possible relationship between the content and the optimal combination of downsampling and upscaling ratio. Object size refers to the average foreground object sizes, while overlap ratio refers to the average of the foreground objects' overlapping areas divided by total sizes. These features could be extracted by simply running a background extractor [34], or using an edge detector [35]. Frames are grouped according to three levels of object size (in terms of the ratio of areas in the image): small ($<1/64$), medium ($1/64 - 1/16$), and large ($>1/16$), as well as three levels of overlap ratio: $<1\%$, $1\% - 3\%$, and $>3\%$. The division is set to have a roughly equal proportion of samples. Fig. 1(b) and (c) show the proportion of accuracy-optimal configurations with regard to these two handcrafted features. Unfortunately, we find that optimal configurations are still evenly distributed in different groups, indicating that these low-level visual features alone cannot guide us to make appropriate configuration choices.

The above observation reveals a more comprehensive configuration selector is required to extract high-level frame features and capture the relationship between content and the SR configuration decision.

TABLE II
SUMMARY OF MAIN NOTATIONS

Notation	Description	Unit
$\mathcal{I}$	Frame index set of a segment.	-
$\mathcal{J}$	Segment index set of the analyzed video.	-
$\mathcal{D}$	Candidate of downsampling ratio.	-
$\mathcal{U}$	Candidate of SR upscaling ratio.	-
$\mathcal{T}$	Time period of observation.	s
$N_{j,d}$	Network cost for downsampling rate d of segment j.	\$
$\widetilde{N}_{j,d}$	Network cost estimation for downsampling rate d of segment j.	MB
$v_{j,d}$	Transmitted volume after selecting downsampling rate d of segment j.	MB
c	Cost charged per unit data volume.	\$ / MB
$C_{i,j}$	Server computational cost for frame i in segment j.	J
P	Working power of the server.	W
$t_{i,j}$	Total inference time on the server.	s
$s_{j,0}$,$s_{j,1}$,$s_{j,2}$	First, middle, and last frames of segment j	-
Ψ	Average utility for the analyzed video.	-
$\overline{A}$, $\overline{N}$, $\overline{C}$	Average normalized value of accuracy, network cost, and computational cost for the analyzed video.	-
ω_1, ω_2, ω_3	Coefficients serving for balancing accuracy, network cost, and computational cost.	-

IV. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we present the framework of our SR-enhanced video analytics pipeline, model the key factors in our system, and finally formulate the studied problem. We provide a summary of important notations and related meanings in Table II.

A. SR-enhanced Video Analytics Pipeline

We consider a typical video analytics system where stationary cameras with limited computing power continuously stream videos to the remote server for conducting accurate video analytics tasks, as shown in Fig. 2. Due to the high potentials of SR for video analytics, and limited computing resources equipped in smart cameras, we put the SR enhancing and video analytics processes to the server, which could be equipped with rich computational resources. We focus on retrospective video analytics applications [36] [37] such as incident investigation and traffic volume analysis, which care about transmission traffic cost rather than latency.

Specifically, the camera first encodes the buffered frames into segments (each of which consists of s frames) and downsamples them to the desired resolution by using a certain codec. It then transmits them to a cloud server through a network link for frame inference. When the downsampled segments reach the server, they are decoded into frames, and a customized SR model (if necessary) is utilized to enhance the frame quality. The frames are then fed into a video analytics model for final inference.

We consider the analyzed frames to be captured sequentially in a time window of observation $\mathcal{T}$. These frames are partitioned into $\mathcal{J}$ segments, each of which is composed of $\mathcal{I}$ frames. The configuration selector is deployed on the camera to determine the segment downsampling and frame SR upscaling ratio. $\mathcal{D}$ and $\mathcal{U}$ are the sets comprising all the candidate downsampling and SR upscaling ratios, respectively.

B. Network Cost Model

Typically, videos are compressed in the unit of segments comprising several consecutive frames, in order to save the compressed data size by leveraging the temporal consistency of videos. We can model the network cost $N_{j,d}^{n}$ after selecting the downsampling knob $d \in \mathcal{D}$, for the j^{th} segment as:

$$N_{j,d} = c \cdot v_{j,d} \tag{1}$$

where $v_{j,d}$ is the file size of the j^{th} segment after selecting the downsampling knob $d \in \mathcal{D}$, and c is the cost charged per unit data volume.

C. Server Computational Cost Model

When the server receives a compressed segment from the camera, it will decode the segment into frames and execute (if necessary) customized SR inference according to the instruction from the camera, and then feed the frame to the object detection model. We perform customized SR upscaling for each frame since we find that the choice of SR upscaling ratio leading to the highest inference accuracy is vulnerable to motion changes, and thus frame-level upscaling are conducted for better accuracy preservation.

The server computational cost is highly related to the server-side inference time. The total inference time is a sum of the SR inference time (if necessary), as well as the running time of the object detection model. So the total time for inferring the i^{th} frame is:

$$t_{i,j} = t_{i,j}^{SR} + t_{i,j}^{anal} \tag{2}$$

where $t_{i,j}^{SR}$ is the SR inference time, and $t_{i,j}^{anal}$ is the time for completing video analytics tasks.

The server computational cost (C) is a product of the working power of the server (where GPU serves as the dominant hardware for conducting the computational task), and the computational time. Then, for the i^{th} frame of the j^{th} segment, we can express the server computational cost as:

$$C_{i,j} = P \cdot t_{i,j} \tag{3}$$

where P is the average working power of the server which runs the inference tasks.

Then, we present several findings to simplify the server computational cost model.

- One finding is that the inference time of the video analytics tasks, specifically multi-object detection in our work, i.e. $t_{\cdot,\cdot}^{anal}$ in Eq. 2, is negligible compared with SR inference time $t_{\cdot,\cdot}^{SR}$. Therefore, the difference of $t_{\cdot,\cdot}^{anal}$ among configurations does not affect the relative utility much. We thus can neglect $t_{\cdot,\cdot}^{anal}$ in Eq. 2 when computing or comparing the utility function.
- Also, it has been shown in Table I that the standard deviation of SR inference time for each configuration is quite small, compared with the average SR inference time. Thus, we could consider each configuration has a fixed server computational cost, across all the frames. In other words, $t_{\cdot,\cdot}^{SR}$ could be considered only dependent on the selected configuration (if the frame raw size is fixed).

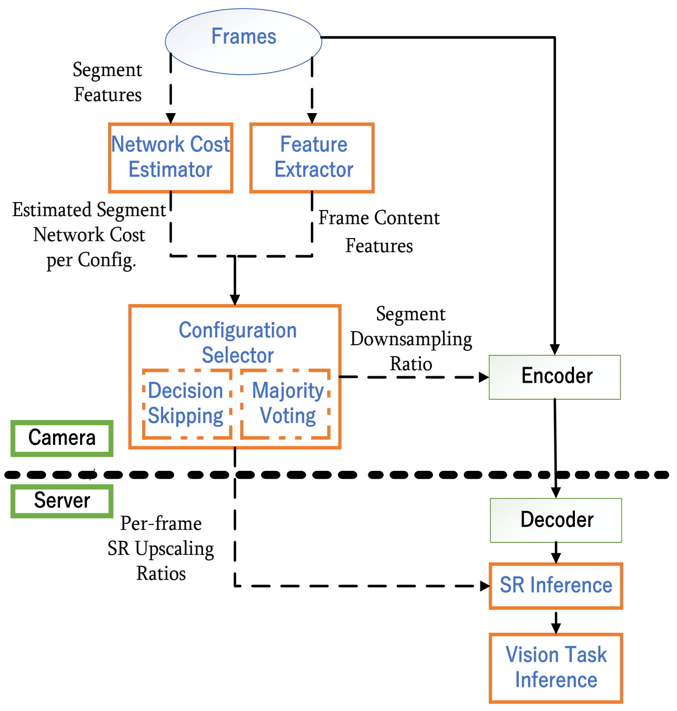


Fig. 3. System overview. Dashed arrow line represents *AdaDSR*'s control pipeline. Solid arrow line represents the video analytics streaming pipeline.

Therefore, the server computational cost could be approximated only by captured frame resolution and the selected configuration.

D. Utility Model

Inference accuracy, network transmission cost, and server-side computational cost are three metrics that users mostly care about for retrospective video analytics. However, there exists a complex trade-off among these metrics. As shown in Sec.III, the inference accuracy is not only affected by the joint configuration selection of both segment downsampling and frame SR upscaling ratio, but it may also have distinct performances for the same configuration across various frame content. We introduce the following utility function Ψ to incorporate all three dimensions.

$$\Psi = \omega_1 \overline{A} + \omega_2 (1 - \overline{N}) + \omega_3 (1 - \overline{C}) \tag{4}$$

where $\omega_1, \omega_2, \omega_3$ are the non-negative normalized weights balancing among the average normalized value of analysis accuracy ($\overline{A}$), network transmission cost ($\overline{N}$), and server-side SR computational cost ($\overline{C}$). For example, some users care most about inference accuracy, with little concern about network or server computational cost, then the ω_1 should be set to be much greater than ω_2 and ω_3. Under another scenario, the user may only have a very limited budget for running tasks on a server, and thus server computational cost becomes their priority concern, then ω_3 should not be set much smaller than ω_1 and ω_2. In other words, the weights could be determined according to users' preferences. The purpose of designing customized weights is to showcase the versatility of our framework in accommodating diverse preferences. In practice, predefined weight combinations can be made available to users. Users can conveniently select the weight combination that meets their needs.

E. Problem Formulation

Our goal is to maximize the user-defined utility function Ψ, by adaptively selecting the configuration knob for each frame in the analyzed period.

We use the decision variable $x_{i,j}^{d,u}$ to represent our configuration selection in the candidate pool. $x_{i,j}^{d,u}$ is set to be 1 when we choose the d^{th} configuration of frame downsampling ratio (from $\mathcal{D}$) along with the u^{th} configuration of SR upscaling ratio (from $\mathcal{U}$), for the i^{th} segment's j^{th} frame.

Formally, our utility maximization problem is formulated as follows:

$$\max_{x_{i,j}^{d,u}} \quad \Psi \tag{5}$$

$$s.t. \quad \sum_{d\in\mathcal{D}}\sum_{u\in\mathcal{U}} x_{i,j}^{d,u} = 1 \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{J} \tag{6}$$

$$\sum_{j}\sum_{u\in\mathcal{U}} x_{i,j}^{d,u} \in \{0, s\} \quad \forall d \in \{1, 2..., |\mathcal{D}|\} \quad \forall i \in \mathcal{I} \tag{7}$$

$$x_{i,j}^{d,u} \in \{0,1\} \quad \forall d \in \{1, 2..., |\mathcal{D}|\} \quad \forall u \in \{1, 2..., |\mathcal{U}|\} \quad \forall i \in \mathcal{I} \quad \forall j \in \mathcal{J} \tag{8}$$

Constraints (6) and (8) ensure that for each frame, we select and only select one configuration in the candidate pool. Constraint (7) ensures that the downsampling ratios are the same for all frames in a segment.

Due to the intricate relationships between SR configurations and our accuracy, network, and computational cost, close-form equations to describe these relationships cannot be derived. This motivates us to use an NN-enhanced method to approximate the optimal solution.

V. Design

In this section, we present the detailed method and procedure for optimizing the utility maximization problem shown in Eq. 5. We first present the overall architecture and workflow of our system, and then state the detailed functions as well as the training processes of the different components respectively.

A. System Workflow

Fig. 3 illustrates the detailed workflow of *AdaDSR*. We first extract content features from the current video segment. The features are then fed into a random-tree-regressor to estimate the segment network cost per each downsampling ratio. The estimated network cost is then fed into the CNN-based configuration selector along with the frame-level content features from the feature extractor, to get the frame-level temporary optimal configuration selection according to users' defined utility. We optimize the selector by using decision skipping to reduce the processed frames running on the per-frame configuration selector, leveraging videos' temporal consistency. Finally, the majority voting scheme is applied to generate a consistent configuration for all frames in a segment and output the final configuration selection.

B. Per-frame Configuration Selector

Fig. 4 depicts the overall architecture of our per-frame configuration selector. Our customized utility-based configuration selector receives both the frame-level content features as well as the segment-level network cost estimation, both of which are obtained from the pretrained machine-learning-based unit. To be specific, a high-level feature map is extracted from compressed frames by a pre-trained feature extractor. After receiving the frame's deep features, a convolution layer is applied to capture the general neighboring characteristics of the extracted frame-level feature map, followed by the maximum pooling to reduce the dimension of feature size. Then the resulting neurons are flattened, concatenated with the estimated network cost from the regressor (normalized for better training performance) for each downsampling ratio, and fed into an FC layer. A softmax layer is applied to generate the classification output finally.

Here we introduce the frame content feature extraction and network cost estimator, serving as the vital building block of our configuration selector, as well as the detailed guideline of the overall training procedure.

Frame Content Feature Extraction. The deep features are extracted with the backbone of a lightweight pre-trained neural network, mobileNetv3-small [38]. The extracted features contain the high-level semantic information of the image, e.g., the class, size, edge quality, etc. Although the camera-side inference overhead is not included in our utility, it also needs to be considered since the camera-side resources are limited, which may prohibit the deployment of large models or cannot support frequent inference operations. In our implementation, we select mobileNetv3-small, a model with a much smaller parameter size and faster processing time, rather than other backbone models for object detection, such as Faster R-CNN [39], as our high-level frame feature extractor. We remove the last max pooling layer as well as the following FC layers to fit into our own task.

Network Cost Estimator. To estimate the data size of one segment with different downsampling ratios, we design a novel network cost estimation regressor based on random forest. It

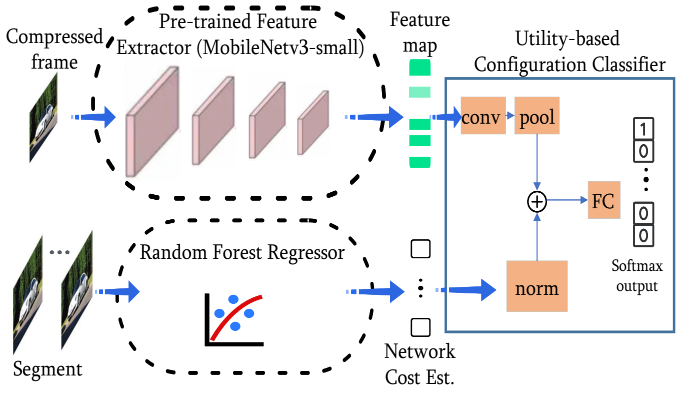


Fig. 4. Classifier architecture.

estimates the network cost for each compression rate under the current codec per segment. The ideal regressor should both 1). be easy to compute to reduce the camera-side processing time, and 2). capture the intra- and inter-frame complexity well, which could provide the information for estimating the segment size. Random forest is selected as the machine-learning-based model for the task, due to its simple network structure [40], as well as its high generalization ability without the effect of over-fitting [41].

Specifically, we choose three frames $s_{j,0}$, $s_{j,1}$, and $s_{j,2}$, representing the first, middle, and last frames of the j^{th} segment, which is assumed to reflect the segment properties well since the segment duration we choose is quite short ($\sim$ 2 seconds). For each downsampling rate d and each raw video segment j, the regressor outputs the network cost estimation $\widetilde{nw}_{j,d}$:

$$\widetilde{N}_{j,d} = \mathcal{F}_i(size(s_{j,1}), std(s_{j,1}), diff(s_{j,0}, s_{j,1}, s_{j,2})) \quad (9)$$

where $size(\cdot)$, $std(\cdot)$ represents the size and spatial standard deviation of an image, capturing the intra-frame complexity; while $diff(\cdot,\cdot,\cdot)$ represents the average pixel-level difference across the frames, capturing the inter-frame complexity. All three operations require computation on low-level handcrafted numerical features only. $\mathcal{F}(\cdot)$ is the pre-trained random forest regressor.

We now introduce the overall training process of *AdaDSR*, which is done after obtaining the pre-trained network cost estimator. We first dig into the accuracy metric specifically, which serves as an important component in the user's utility. We have found that the accuracy is quite unstable for different frames. Some frames could achieve satisfactory detection accuracy with almost any configuration, while some others' detection accuracy is low even when applying the optimal configuration. This is because the detection difficulty fluctuates a lot. For example, some frames only consist of objects of classical sizes or shapes, and/or the background does not affect the foreground detection; while some others may consist of many objects with weird shapes, and/or the objects are not salient enough due to the background color. Therefore, the training process will be unstable when one uses an online bandit [42] using the cumulative value of pure inference accuracy for rewards, which cannot reflect the goodness of the current configuration selection well with regard to the accuracy concern. In other words, the absolute inference accuracy can be shaped by specific content, making it difficult to tell whether lower absolute accuracy values are caused by poor configuration choices or complex content.

Therefore, we normalize the accuracy into the interval [0,1]. "0" would be endowed if the configuration has the lowest accuracy for the current frame, while "1" would be given for the configuration inducing the highest accuracy. To make it uniform, the network cost and server-side computational cost are also normalized in practical implementation.

For the overall training process, we develop one-hot labels to mark the configuration with the maximum utility. For a given frame, its ground-truth label is a one-hot vector of a length C, where C is the number of candidate (after filtering) configurations, which is not greater than $|\mathcal{D}| \cdot |\mathcal{U}|$. The weight of the configuration selector is optimized by minimizing the multi-class cross-entropy loss.

$$L = -\frac{1}{|m_{batch}|} \sum_{i \in m_{batch}} \sum_{c \in C} y_{i,c} \log \hat{y}_{i,c} \quad (10)$$

where m_{batch} denotes the samples in a training batch. $y_{i,c}$ is the ground-truth label, while $\hat{y_{i,c}}$ is our prediction value, for the c^{th} configuration of the i^{th} frame.

C. Online Deployment

When deployed online, algorithm 1 is executed to generate the final configurations for all frames in a segment. First, we apply a selection skipping scheme (Lines 4-17) to optimize the per-frame configuration selector. To be specific, by leveraging the temporal consistency of videos, we did not need to run our selector for all frames. This could greatly save the camera-side computational cost, even though it is not included in our utility. Then, majority voting (Lines 18-22) is employed to get the final configuration selection. The frame-level configuration selector introduced in Sec. V-B provides the optimal configuration generated by our NN-based selector for each frame. However, constrained by the requirement given in (7), all the frames in the same segment should have the same downsampling ratio. In other words, some frames need to make some compromise into a sub-optimal configuration to generate feasible configurations. We now introduce the two components below in detail.

Selection Skipping scheme. Thanks to the content similarity of consecutive frames, there is no need to run the configuration selector on each frame in production. Frames without significant content variations compared to the reference frame will skip the configuration classification procedure and reuse the cached configuration $cache\_config$ to save the camera's computational overhead (Lines 5 to 7). The skipping decision is controlled by a pre-defined threshold ϵ, indicating the average square of pixel difference. Frame j will be fed to the configuration selector only when the pixel-level difference between frame j and the reference frame is above ϵ (Lines 9

Algorithm 1 Online Configuration Selection

```
1: Input: A video segment $i$; frame difference threshold $\epsilon$;
2: Output: $x_{i,j}^{d,u}$.
3: Initialization: $x_{i,j}^{d,u} \leftarrow 0$; $ref\_frame \leftarrow$ the first frame;
   $config\_list$, $softmax\_list \leftarrow$[],[].
4: for each frame $j$ in segment $i$ do
5:   if diff($j, ref\_frame$)$\leq \epsilon$ and j is not the first frame then
6:     $config\_list$.append($cache\_config$)
7:     $softmax\_list$.append($cache\_softmax$)
8:   else
9:     $config \leftarrow \arg\max\ per\_frame\_selector$(frame $j$)
10:    $cache\_config \leftarrow config$
11:    $config\_list$.append($config$)
12:    $curr\_softmax \leftarrow per\_frame\_selector$(frame $j$)
13:    $cache\_softmax \leftarrow curr\_softmax$
14:    $softmax\_list$.append($cache\_softmax$)
15:    $ref\_frame \leftarrow j$
16:  end if
17: end for
18: Find the downsampling ratio $d$ with the highest frequency
    in $config\_list$.
19: for each frame $j$ in segment $i$ do
20:   $prune\_softmax \leftarrow$ entries of $softmax\_list[j]$ with
      downsampling ratio $d$
21:   $u \leftarrow \arg\max\ prune\_softmax$
22:   $x_{i,j}^{d,u} \leftarrow 1$
23: end for
```

to 16). Intuitively, ϵ serves as a trade-off between preserving a high utility and saving camera-side computational overhead. Sec. VI-E further shows that our method is highly robust to a wide range of ϵ.

Majority Voting. We employ majority voting to determine the downsampling ratio for a segment. To be specific, we keep track of the predicted optimal configurations for all frames with $config\_list$ and select the downsampling ratio d with the highest frequency in $config\_list$ as the segment's downsampling ratio (Line 18). Each frame's upscaling ratio is chosen (Lines 19 to 22) to be the one leading to the largest softmax output which has the downsampling ratio d, recorded by the softmax output list $softmax\_list$.

VI. Experimental results

In this section, we evaluate the performance of *AdaDSR* on real-world video clips.

A. Methodology and Setup

Dataset and Experimental Setup. We evaluate our proposed method on the same video clips, and apply the same hardware setup as in Sec. III. All video segments are encoded with H.264. Similarly, we conduct multi-object detection on cars, traffic lights, and pedestrians, which are the three most common objects for traffic surveillance videos. We choose mAP@0.75 as our metric for accuracy. Network cost is

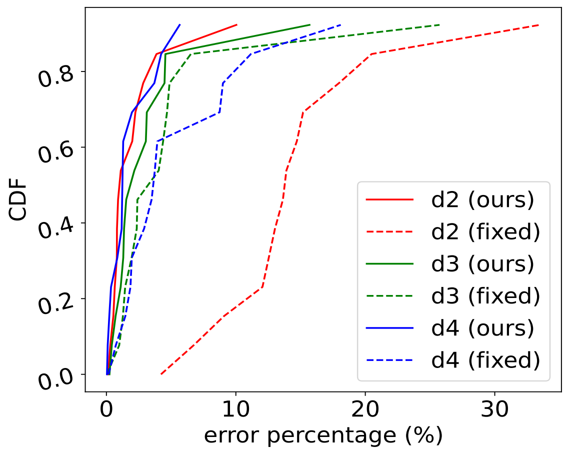


Fig. 5. CDF of error percentage of segment network cost estimation for different methods. d2, d3, and d4 represent downsampling ratio of 2×, 3×, and 4×, respectively.

measured in transmitted data size, while computational cost is computed to be proportional to the SR inference time of the selected configuration, represented by the simplification model illustrated in Sec. V-B. We split each video clip into 6 segments, each of which contains 600 video frames, with 80% for training, and 20% for testing.

The candidate of configuration space is chosen as: frame downsampling ratio in {2×, 3×, 4×}; SR upscaling ratio in {1× (i.e. no SR), 2×, 3×, 4×}. We choose three different combinations[2] of $\omega_1, \omega_2, \omega_3$, namely (0.7, 0.2, 0.1), (0.5, 0.3, 0.2), and (0.5, 0.2, 0.3), to indicate users' preference on higher inference accuracy, less network cost, and less server computational cost, respectively.

Parameter Setting. The number of random trees in the network cost estimator is set to 10. For the configuration selector, we set the kernel size of convolution to be 3 × 3, and the neurons in the FC layer to be 100. The selector is trained offline with a learning rate of 10^{-4}. When deployed online, the skipping threshold ϵ is set to be 10 in Sec. VI-C.

Benchmark. We compare *AdaDSR* with the following benchmarks.

1) Random: Configurations are chosen at random.

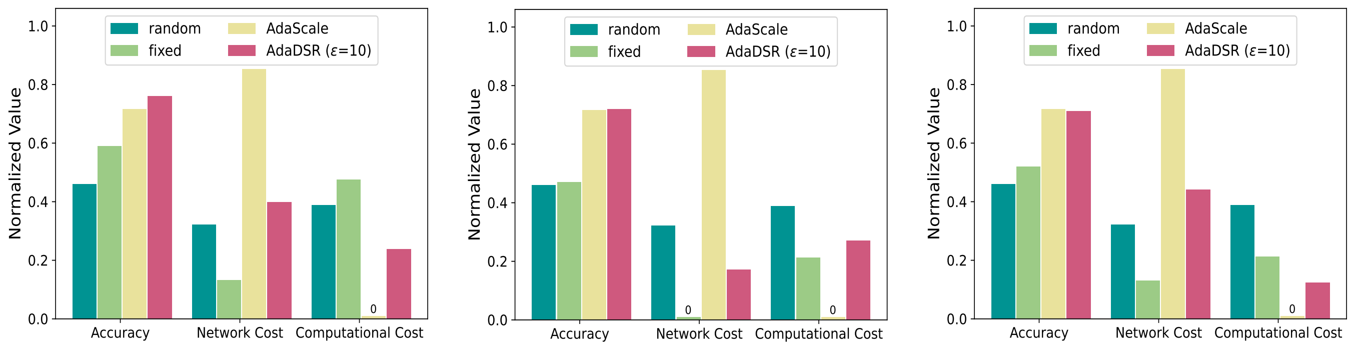
2) Fixed: This is an offline profiling [27] [10] based solution tailored to our problem. It considers the same candidate configurations as *AdaDSR* but always chooses the configuration that achieves the highest mean utility on the training dataset.

3) AdaScale [43]: It uses a pre-trained R-FCN [44] backbone for feature extraction, and trains a CNN regressor to determine the optimal downsampling ratio, in order to achieve high accuracy without using any SR models.

B. Performance for the Network Cost Estimator

Before analyzing the utility performance of our whole system, we first show the performance of our random-forest-

[2]Since the motivation of introducing SR is mainly to enhance inference accuracy, ω_1 is chosen to be larger than ω_2 and ω_3 in all scenarios to ensure satisfactory accuracy.



(a) Accuracy biased (b) Network cost biased (c) Computational cost biased

Fig. 6. Normalized values for accuracy, network cost, and computational cost under various user preferences.

based network cost estimator. Fig. 5 shows the cumulative distribution function curves of the error percentage for the network-cost estimation in the test set. d2, d3, and d4 represent downsampling ratio of 2×, 3×, and 4×, respectively. Solid lines refer to our method for prediction, while dashed lines refer to the benchmark which uses the average segment size for each downsampling ratio for prediction. We use absolute error percentage to reflect the performance of the estimators, which is calculated as $|est - true|/true$, where est and $true$ are the estimated and true values of the segment size, respectively.

We find that our estimation is very close to the true data size for each downsampling ratio, with an average percentage of error of only about 2%, which is greatly smaller than the benchmark. Furthermore, the maximum percentage of error of our method is about 15% only, which is nearly half our the benchmark, showing the tail error is also controllable.

C. Performance Comparison on Utility

Fig. 6 shows the overall normalized accuracy, network, and computational cost of three settings for all videos in the dataset. All measured values are normalized to [0, 1] for fair comparison. High accuracy and low network/computational cost normalized value are expected. The three subfigures correspond to the accuracy-biased, network-cost-biased, and computational-cost-biased settings, respectively. From the subfigures, we find that *AdaDSR* can satisfy users' preferences without substantially compromising the other two metrics. Specifically, *AdaDSR* significantly outperforms the preference-agnostic random scheme in all scenarios. Compared with the accuracy-centric AdaScale scheme, *AdaDSR* achieves higher or comparable accuracy (with up to 5% normalized accuracy boost) at substantially lower network costs (with at least 48% saving of network cost). It means that *AdaDSR* balances a satisfactory trade-off between accuracy and network cost. Although the fixed scheme can also adapt to users' preferences to some extent, it cannot strike the desired trade-offs precisely for unseen video frames with new content features. The selected configuration would either lead to a low level of accuracy (e.g. the accuracy is comparable for the network-cost-biased case) or induce a much higher computational cost (e.g. suffer from the highest computational cost for the accuracy-biased case), according to the performance of the three settings.

Table III further demonstrates the reached utility values of all schemes for each video clip. As one can see, *AdaDSR* achieves the highest utility under all cases, with an average of about 18.4%, 7.2%, and 8.8% gain over Random, Fixed, and AdaScale, respectively. It shows that *AdaDSR* has a robust performance under a variety of video scenes and illumination conditions.

D. Sensitivity analysis of the frame difference threshold ϵ

In Sec. VI-C, we set the frame difference threshold ϵ to be 10 by default. However, different ϵ might affect the utility, since many frames would follow the configuration of the reference frame, which might lead to sub-optimal performance.

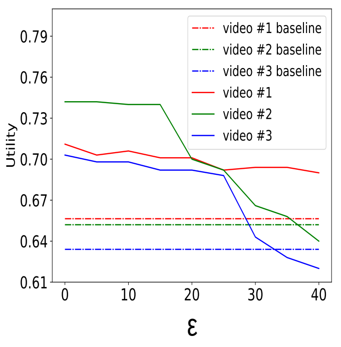
We thus conduct a sensitivity analysis of frame difference threshold ϵ to show the effect of camera-side decision skipping on utility. We provide our findings for the accuracy-biased setting in Fig. 7, and we believe that the findings could be generalized to the other settings. Fig. 7 (a) plots the effect of ϵ on our utility, and the three dashed baselines correspond with the benchmark performing the best. Fig. 7 (b) shows the effect of ϵ on the frame-skipping ratio, which could reflect the characteristic of temporal consistency for the videos purposed for surveillance. We find that when ϵ is small, frames are skipped at a high rate, with only a minor utility drop. For example, when ϵ=10, an average of 70% of frames could be skipped, while the normalized utility only drops by 1% at most. Therefore, *AdaDSR* is highly robust to a wide-range small ϵ, benefiting from a large proportion of frame skipping gain at the same time. This is because frames with little difference may favor a similar configuration, and thus there is no need to run our frame-level configuration selector for all the frames.

E. Camera-side Inference Overhead

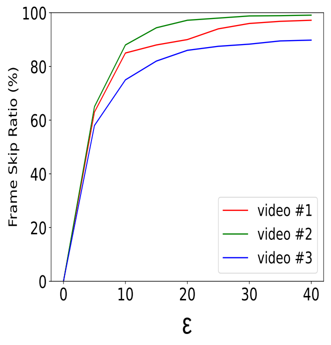
We keep track of the frame content feature extraction time as well as the configuration selection model inference time at the camera side to reveal the inference overhead. The total

TABLE III
NORMALIZED UTILITY COMPARISON (FIRST THREE ROWS' ENTRY CORRESPONDS TO RANDOM / FIXED / ADASCALE / **ADADSR** (ϵ=10), AND THE LAST ROW REFERS TO THE COMPARISON WITH RANDOM / FIXED / ADASCALE).

# clip	Accuracy-biased	Network-cost-biased	Computational-cost-biased
1	0.539 / 0.625 / 0.656 / **0.716**	0.568 / 0.676 / 0.644 / **0.759**	0.550 / 0.665 / 0.685 / **0.720**
2	0.499 / 0.652 / 0.604 / **0.742**	0.544 / 0.708 / 0.561 / **0.748**	0.517 / 0.655 / 0.633 / **0.705**
3	0.554 / 0.626 / 0.634 / **0.703**	0.578 / 0.655 / 0.686 / **0.745**	0.544 / 0.634 / 0.658 / **0.712**
Avg. Gain (%)	19.0 / 8.6 / 8.9	18.7 / 7.1 / 12.0	17.5 / 18.3 / 5.4



(a) Utility



(b) Frame Skip Ratio

Fig. 7. Sensitivity analysis of the frame difference threshold ϵ.

computation time at the camera side only accounts for less than 10% inference time of the SR inference time. Fig. 7 (b) has revealed that a majority of frames would be skipped when ϵ is under a predefined value. We find that when combined with the frame decision-skipping scheme, the average camera-side computational time per frame can be reduced to 13.8 ms, accounting for only 1.8% of the server-side SR time. By comparison, if we use yolov7 to do direct inference, the inference time boosts up to 207 ms on average, and Adascale consumes up to 232 ms inference time. Therefore, *AdaDSR* consumes little camera-side inference overhead compared with direct inference or state-of-the-art methods.

VII. CONCLUSION

Neural-based super-resolution opens up new opportunities for resource-efficient video analytics applications. In this paper, we first conducted thorough data measurements to show that, first, SR-enhanced video frames with the highest accuracy require that the downsampling ratio and the upscaling ratio not necessarily be the same; Second, simple vision features are insufficient to determine the best ratio combinations. We then formulate the SR-enhanced video analytics problem as a utility maximization problem that jointly considers the practical analytic performance, network cost, and communication cost. We design a neural network-based configuration selector to learn the complex relationship between content and the optimal configuration and determine the desirable configurations. The temporal consistency of videos is further utilized to perform decision-skipping so that the resource consumption on the camera side can be greatly reduced. Experiments on real-world video datasets demonstrate that our approach can improve the average utility by 7.2% -18.4% in diverse video scenes over state-of-the-art baselines.

VIII. ACKNOWLEDGEMENT

This work is supported by National Natural Science Foundation of China (62302292) and the Fundamental Research Funds for the Central Universities. Jiangchuan Liu's work is supported by an NSERC Discovery Grant, a CFI JELF/BCKDF grant, and a British Columbia Salmon Recovery and Innovation Fund (No. 2019-045). We thank Qinhang Wu for making contributions to data gathering and pre-processing.

REFERENCES

[1] S. Guo, C. Zhao, G. Wang, J. Yang, and S. Yang, "Ec^{2}detect: Real-time online video object detection in edge-cloud collaborative iot," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20 382–20 392, 2022.
[2] Z. Wang, X. He, Z. Zhang, Y. Zhang, Z. Cao, W. Cheng, W. Wang, and Y. Cui, "Edge-assisted real-time video analytics with spatial–temporal redundancy suppression," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 6324–6335, 2023.
[3] H. Ma, R. Li, X. Zhang, Z. Zhou, and X. Chen, "Reliability-aware online scheduling for dnn inference tasks in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11 453–11 464, 2023.
[4] D. Feng, L. Zeng, L. Pu, and X. Chen, "Adaptive progressive image enhancement for edge-assisted mobile vision," in *IEEE MSN*, 2022, pp. 744–751.
[5] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, 2016.
[6] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *IEEE CVPR*, 2016, pp. 1646–1654.
[7] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *IEEE CVPR*, 2016, pp. 1874–1883.
[8] Y. Zhang, Y. Zhang, Y. Wu, Y. Tao, K. Bian, P. Zhou, L. Song, and H. Tuo, "Improving quality of experience by adaptive video streaming with super-resolution," in *Proc. IEEE INFOCOM*, 2020, pp. 1957–1966.
[9] M. Dasari, A. Bhattacharya, S. Vargas, P. Sahu, A. Balasubramanian, and S. R. Das, "Streaming 360-degree videos using super-resolution," in *Proc. IEEE INFOCOM*, 2020, pp. 1977–1986.
[10] Y. Wang, W. Wang, D. Liu, X. Jin, J. Jiang, and K. Chen, "Enabling edge-cloud video analytics for robotics applications," in *Proc. IEEE INFOCOM*, 2021, pp. 1–10.
[11] M. Haris, G. Shakhnarovich, and N. Ukita, "Task-driven super resolution: Object detection in low-resolution images," in *Proc. Springer Neural Info. Proc. Int. Conf.*, 2021, pp. 387–395.
[12] Y. Wang, W. Wang, J. Zhang, J. Jiang, and K. Chen, "Bridging the edge-cloud barrier for real-time advanced vision analytics," in *Proc. USENIX HotCloud*, 2019.
[13] M. Nguyen, E. Çetinkaya, H. Hellwagner, and C. Timmerer, "Super-resolution based bitrate adaptation for HTTP adaptive streaming for mobile devices," in *Proc. ACM Mile-High Video Conf.*, 2022, pp. 70–76.

[14] J. Filho, M. Coelho, and C. A. V. Melo, "Super-resolution on edge computing for improved adaptive HTTP live streaming delivery," in *Proc. IEEE CloudNet*, 2021, pp. 104–110.
[15] X. Liu, Z. Ke, X. Zhou, T. Qiu, and K. Li, "Qoe-oriented adaptive video streaming with edge-client collaborative super-resolution," in *IEEE GLOBECOM*, 2022, pp. 6158–6163.
[16] A. Zhang, C. Wang, B. Han, and F. Qian, "Efficient volumetric video streaming through super resolution," in *Proc. ACM HotMobile*, 2021, pp. 106–111.
[17] A. Zhang, Q. Li, Y. Chen, X. Ma, L. Zou, Y. Jiang, Z. Xu, and G. Muntean, "Video super-resolution and caching - an edge-assisted adaptive video streaming solution," *IEEE Trans. Broadcast.*, vol. 67, no. 4, pp. 799–812, 2021.
[18] J. Zhang, J. Lei, W. Xie, Z. Fang, Y. Li, and Q. Du, "Superyolo: Super resolution assisted object detection in multimodal remote sensing imagery," *IEEE Trans. Geosci. Remote. Sens.*, vol. 61, pp. 1–15, 2023.
[19] B. Hou, X. Chen, S. Zhou, H. Jiang, and H. Wang, "SR-YOLO: small objects detection based on super resolution," in *ICIS*, vol. 659, 2022, pp. 352–362.
[20] D. Zheng, Y. Fu, H. Zhang, M. Gao, and J. Yu, "Semantic segmentation method based on super-resolution," *Int. J. Perform. Eng.*, vol. 16, no. 5, pp. 711–719, 2020.
[21] Y. Cai, Y. Yang, Y. Shang, Z. Shen, and J. Yin, "Dasrsnet: Multitask domain adaptation for super-resolution-aided semantic segmentation of remote sensing images," *IEEE Trans. Geosci. Remote. Sens.*, vol. 61, pp. 1–18, 2023.
[22] M. Shan, S. Zhang, M. Xiao, and Y. Zhao, "LENS: bandwidth-efficient video analytics with adaptive super resolution," *Comput. Networks*, vol. 218, p. 109392, 2022.
[23] I. G. Aguilar, R. Baena, and E. López-Rubio, "Improved detection of small objects in road network sequences using CNN and super resolution," *Expert Syst. J. Knowl. Eng.*, vol. 39, no. 2, 2022.
[24] J. Shermeyer and A. V. Etten, "The effects of super-resolution on object detection performance in satellite imagery," in *Proc. IEEE CVPR*, 2019, pp. 1432–1441.
[25] S. Mu, Y. Zhang, and Y. Jiang, "Drn-videosr: a deep recursive network for video super-resolution based on a deformable convolution shared-assignment network," *Multim. Tools Appl.*, vol. 82, no. 9, pp. 14 019–14 035, 2023.
[26] M. Liu, S. Jin, C. Yao, C. Lin, and Y. Zhao, "Temporal consistency learning of inter-frames for video super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 4, pp. 1507–1520, 2023.
[27] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. Lee, "Awstream: adaptive wide-area streaming analytics," in *Proc. ACM SIGCOMM*, 2018, pp. 236–252.
[28] Jackson hole wyoming usa town square live cam. https://www.youtube.com/watch?v=1EiC9bvVGnk/.
[29] Live cam: rotary traffic circle derry nh. https://www.youtube.com/watch?v=fuuBpBQElv4/.
[30] Live stream—public square, watertown ny. https://www.youtube.com/watch?v=qP1y7Tdab7Y/.
[31] K. Zhang, W. Zuo, and L. Zhang, "Deep plug-and-play super-resolution for arbitrary blur kernels," in *Proc. IEEE CVPR*, 2019, pp. 1671–1681.
[32] [Online]. Available: https://github.com/WongKinYiu/yolov7
[33] S. Liu, T. Wang, J. Li, D. Sun, M. B. Srivastava, and T. F. Abdelzaher, "Adamask: Enabling machine-centric video streaming with adaptive frame masking for DNN inference offloading," in *Proc. ACM MM*, 2022, pp. 3035–3044.
[34] B. Wang and P. Dudek, "AMBER: adapting multi-resolution background extractor," in *IEEE ICIP*, 2013, pp. 3417–3421.
[35] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.
[36] Z. Gao, G. Lu, P. Yan, and L. Wang, "Retrospective analysis of time series for frame selection in surveillance video summarization," *Signal Image Video Process.*, vol. 11, no. 4, pp. 581–588, 2017.
[37] K. N. Bui, H. Yi, H. Jung, and J. Cho, "Video-based traffic flow analysis for turning volume estimation at signalized intersections," in *ACIIDS*, vol. 12034, 2020, pp. 152–162.
[38] A. Howard, R. Pang, H. Adam, Q. V. Le, M. Sandler, B. Chen, W. Wang, L. Chen, M. Tan, G. Chu, V. Vasudevan, and Y. Zhu, "Searching for mobilenetv3," in *Proc. IEEE/CVF ICCV*, 2019.
[39] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *Proc. IEEE/CVF NeurIPS*, 2015, pp. 91–99.
[40] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[41] [Online]. Available: https://en.wikipedia.org/wiki/Talk%3ARandom_forest
[42] N. Gutowski, O. Camp, T. Amghar, and F. Chhel, "Using individual accuracy to create context for non-contextual multi-armed bandit problems," in *Proc.IEEE-RIVF Int. Conf. Comput. Comm. Tech.*, 2019, pp. 1–6.
[43] T. Chin, R. Ding, and D. Marculescu, "Adascale: Towards real-time video object detection using adaptive scaling," in *Proc. IEEE MLSys*, 2019, pp. 431–441.
[44] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: object detection via region-based fully convolutional networks," in *Proc. IEEE/CVF NeurIPS*, 2016, pp. 379–387.

Sheng Cen received his B.Eng. degree from UM-SJTU Joint Institute, Shanghai Jiao Tong University, China in 2021. He is currently a Ph.D. candidate in UM-SJTU Joint Institute, Shanghai Jiao Tong University. His research interests include edge computing, multimedia systems, and satellite networking.

Miao Zhang received her B.Eng. degree from Sichuan University, Chengdu, China, in 2015 and her M.Eng. degree from Tsinghua University, Beijing, China, in 2018. She is currently a Ph.D. student in the School of Computing Science at Simon Fraser University, Burnaby, BC, Canada. Her research interests include cloud computing and multimedia systems.

Yifei Zhu is currently an Assistant Professor at the University of Michigan-Shanghai Jiao Tong University Joint Institute in Shanghai Jiao Tong University, China. He received his B.E. degree from Xi'an Jiaotong University, China, in 2012, his M.Phil. degree from The Hong Kong University of Science and Technology, China, in 2015, and his Ph.D. degree in Computer Science from Simon Fraser University, Canada, in 2020. His current research interests include edge computing, multimedia networking, and distributed machine learning systems, where he has published in ACM SIGCOMM, IEEE INFOCOM, ACM Multimedia, and many other venues.

Jiangchuan Liu (S'01-M'03-SM'08-F'17) is a Professor at Simon Fraser University, BC, Canada. He is a Fellow of The Canadian Academy of Engineering and an IEEE Fellow. He received B.Eng. (cum laude) from Tsinghua University and Ph.D. from HKUST. He has served on the editorial boards of IEEE/ACM Transactions on Networking, IEEE Transactions on Multimedia, IEEE Communications Surveys and Tutorials, and IEEE Internet of Things Journal. He was a Steering Committee member of IEEE Transactions on Mobile Computing and Steering Committee Chair of IEEE/ACM IWQoS. He was TPC Co-Chair of IEEE INFOCOM'2021.