DAO: Dynamic Adaptive Offloading for Video Analytics

Taslim Murad
Georgia State University

Anh Nguyen
George Mason University

Zhisheng Yan
George Mason University

ABSTRACT

Offloading videos from end devices to edge or cloud servers is the key to enabling computation-intensive video analytics. To ensure the analytics accuracy at the server, the video quality for offloading must be configured based on the specific content and the available network bandwidth. While adaptive video streaming for user viewing has been widely studied, none of the existing works can guarantee the analytics accuracy at the server in a bandwidth- and content-adaptive way. To fill in this gap, this paper presents DAO, a dynamic adaptive offloading framework for video analytics that jointly considers the dynamics of network bandwidth and video content. DAO is able to maximize the analytics accuracy at the server by adapting the video bitrate and resolution dynamically. In essence, we shift the context of adaptive video transport from traditional DASH systems to a new dynamic adaptive offloading framework tailored for video analytics. DAO is empowered by some new discoveries about the inherent relationship among analytics accuracy, video content, bitrate, and resolution, as well as by an optimization formulation to adapt the bitrate and resolution dynamically. Results from real-world implementation of object detection tasks show that DAO's performance is close to the theoretical bound, achieving 20% bandwidth saving and 59% category-wise mAP improvement compared to conventional DASH schemes.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**.

KEYWORDS

Video analytics, adaptive video offloading, neural networks

ACM Reference Format:
Taslim Murad, Anh Nguyen, and Zhisheng Yan. 2022. DAO: Dynamic Adaptive Offloading for Video Analytics. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548249

1 INTRODUCTION

Video analytics plays a pivotal role in science and engineering monitoring. The advancement of Deep Neural Networks (DNNs) has enabled complicated analytics tasks such as security surveillance [48] and victim search in disaster response [40]. In selected applications [41], even superhuman performance can be achieved. However, advanced DNNs demand excessive computation, placing a barrier to deploying them on constrained end devices that capture the video. Therefore, camera-captured videos are typically offloaded from the end device to a computationally capable edge/cloud server for complex DNN-based video analytics [9, 19, 24, 35, 47].

For desired video analytics, video frames must be continuously offloaded to the server and then accurately analyzed by the DNN. To this end, videos must be offloaded in an adaptive way to accommodate both the *network dynamics* and *content dynamics*. First, the video bitrate must be adapted to dynamic network conditions. The mismatch between bitrate and bandwidth would cause laggy or failed video analytics with large time gaps between frames [5]. The interrupted analytics could diminish the benefits of video analytics since it loses significant insight in the temporal domain. Second, the video resolution and bitrate must be configured based on the given content to ensure satisfactory analytics accuracy at the server. A slight change of the content may severely affect the analytics accuracy even with the same bitrate and resolution [12, 13]. For example, a higher video resolution generally results in a higher video analytics accuracy [25, 35], but this might not hold on to videos with detailed background or occluded objects.

Despite the rich history of research in adaptive video streaming, none of the existing works supports both *bandwidth-adaptive* and *content-adaptive* offloading for video analytics. Most traditional adaptive streaming systems [4, 31] follow the Dynamic Adaptive Streaming over HTTP (DASH) standard. They adapt the video bitrate or resolution dynamically to the current bandwidth in order to maximize video quality. This bandwidth-adaptive approach works well for video viewing systems as a higher bitrate or resolution generally enhances users' viewing experience. However, it does not address the dynamic impacts of video content on analytics accuracy and thus cannot be used for offloading in video analytics. Recently, machine-centered video processing and compression has been studied to preserve visual features of the content and maximize the analytics accuracy of reconstructed videos [8, 10, 11, 17]. These approaches utilize pixel-level processing or DNN to remove analytics-redundant video information. Nevertheless, these are all highly-handcrafted static algorithms without a knob for dynamic bitrate adaptation, preventing them from being used in uplink networks with dynamic and limited offloading bandwidth.

In this paper, we bridge the aforementioned gaps by proposing **DAO**, **D**ynamic **A**daptive **O**ffloading for video analytics. DAO is the first offloading framework for video analytics that supports *both* bandwidth and content adaptation. At the end device, each video chunk is dynamically encoded with optimal bitrate and resolution and then offloaded to the server. This encoding configuration will maximize the analytics accuracy of DNNs deployed at the server while addressing the dynamics of network bandwidth and video content. By shifting the context of adaptive video transport from

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548249

the widely-studied DASH for video viewing to a new scenario of offloading for video analytics, DAO will enable large-scale distributed video analytics applications that are otherwise unattainable.

Realizing DAO requires us to overcome two unique challenges of adaptive video offloading. First, while we are aware of the impacts of bitrate, resolution, and content features on the analytics accuracy, the interplay among these factors and the way to quantify this relationship is unknown. We begin by exploring the DNN perception of video content by learning a novel mapping between the video content and analytics accuracy with respect to various encoding configurations. We learn directly from pixel values and obtain a lightweight convolutional neural network (CNN) feasible for deployment on constrained end devices. As a result, this model can efficiently estimate the analytics accuracy of an offloaded video under a specific bitrate and resolution.

Another challenge is that since adaptive video offloading for analytics is fundamentally different from adaptive video streaming for viewing, the video adaptation for maximal server analytics accuracy (rather than user viewing experience) has not been explored. To address this challenge, we propose an optimization algorithm that integrates the proposed content-accuracy mapping model in order to select the optimal video bitrate and resolution. The optimization is able to handle the complex dynamics of bandwidth and visual features. This algorithm is also effective and efficient to run on camera-equipped end devices.

We prototype DAO using an NVIDIA Jetson module and Ubuntu servers. We validate the designs and algorithms of DAO by focusing on object detection, one of the most popular analytics tasks and a key primitive for many high-level computer vision applications. We evaluate the bandwidth consumption and analytics accuracy of DAO through a dataset of 21 videos and various practical experiments under WiFi and 4G bandwidth. Our results show that the performance of DAO is close to the theoretical bound, achieving 20% bandwidth saving and 59% category-wise mAP improvement with negligible overhead compared to conventional DASH schemes.

To summarize, the contributions of this paper include,

- A dynamic adaptive offloading framework for video analytics that replaces the traditional DASH framework for video viewing (Section 3-4.1).
- A lightweight CNN model characterizing the relationship among video content, analytics accuracy, video bitrate and resolution (Section 4.2).
- An optimization formulation for video adaptation to efficiently maximize the analytics accuracy given dynamic content and networks (Section 4.3).
- A practical demonstration of the satisfactory performance achieved by the dynamic adaptive offloading (Section 5).

2 RELATED WORK

UDP and DASH are two common protocols that support live streaming and offloading of videos. We focus on DASH in this paper because of its recent popularity and its support of distributing live videos over the Internet [46].

Adaptive Video Streaming for Human Viewing. Adaptive video streaming has been used to address the network dynamics in the context of video viewing [21, 28, 34], where video bitrate or resolution is dynamically changed to match the network bandwidth so that user-perceived video quality can be maximized. The formats and structures of adaptive video streaming were defined by the DASH standard [20]. Building on top of DASH, multi-tier [15, 38] and layered video streaming [30] were proposed to improve the tradeoff between bandwidth and video quality. In addition, systems were also designed to optimize the receiving energy of mobile devices in adaptive video viewing [43]. The successful deployment of DASH-based systems in commercial applications demonstrates the necessity of video adaptation for video delivery. However, this line of work is designed for video viewing and they only adapt video quality to the bandwidth dynamics. They cannot be used in adaptive offloading for video analytics because they do not address how content dynamics would affect the analytics accuracy at the server and how the video should be adapted for such content dynamics. Instead, DAO will support this content adaptation.

Video Processing and Compression for Networked Machine Analytics. Despite the development of several lightweight DNNs [18, 44], their computation requirements still prevent the direct deployment of advanced video analytics on everyday end devices, e.g., widely distributed camera devices. Therefore, modern video analytics tasks are often performed at edge/cloud servers through video offloading [19, 24, 35, 47].

Offloading works attempted to first identify salient video data for DNN analytics, such as video frames containing important objects [23] and regions of interest within a frame [11]. Such analytics-salient data could be detected by either local on-device image processing [10, 48] or remote server-initiated feedback [14, 32]. Then the non-salient video data was discarded during the offloading, which was expected to save network bandwidth while retaining analytics performance. However, both types of salient data identification have limitations. On-device methods rely on traditional image processing and detection of pixel-level patterns, which are known to cause false negatives and false positives. On the other hand, the server-initiated methods cause extra delay when sending salient cues from the server to the device. This can make the offloaded data stale and hamper the performance of server analytics.

Recently, machine-centered compression has been studied to represent visual data in a compact way before being offloaded to DNN machine analytics. Low-level visual features were employed to enhance standard codecs such as JPEG and H.265 [17, 42]. These feature-assisted approaches improve the accuracy of video analytics, but they are built on top of standard codecs, where all components are hard-coded. There still remains significant room for improving compression efficiency in these heavily engineered codecs. In addition, autoencoders were utilized for encoding an image into a vector that can be reconstructed later. These approaches [7–9, 27] learned a smaller feature representation of the data and achieved more compact compression than traditional codecs. However, their encoders require sophisticated CNN models to extract latent features from the image, which is challenging to scale to everyday end devices with computation constraints.

More importantly, all aforementioned video processing and compression algorithms for server DNN analytics prepare the data to offload solely based on the input content. The video to offload is fixed in place irrespective of the available bandwidth at the time of

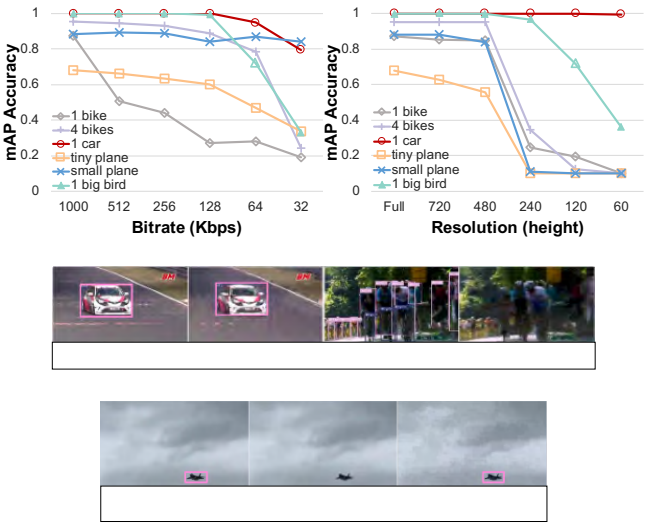

Figure 1: Results show that (1) different videos achieve distinct accuracy when adapting the bitrate/resolution and (2) the effect of adapting bitrate versus resolution on a given video is different.

offloading because it is difficult to dynamically configure these pre-trained models and highly-optimized algorithms. This can cause serious issues in a time-varying uplink offloading network with significantly lower bandwidth than a normal download network. In this paper, we explore the inherent relationship among encoding configuration of videos, network bandwidth, and analytics accuracy to enable both content and bandwidth adaptation in offloading.

3 MOTIVATION

The fundamental reason we cannot use DASH-based approaches in adaptive offloading for video analytics is that the analytics accuracy of the server DNN is not simply determined by either bitrate or resolution in a linear manner. The accuracy is closely related to the nature of video content and is jointly affected by both bitrate and resolution. In this section, we elaborate this point by conducting a motivational study to illustrate this complex relationship.

We examined analytics accuracy under various resolutions and bitrates. We focused on object detection, where the class of objects and the coordinates of object bounding boxes in the input videos are detected. The accuracy of object detection was measured by mean Average Precision (mAP) [16]. We selected 6 annotated videos with distinct content features (single/multiple objects, simple/complex object, small/big object, and simple/complex background) from the ILSVRC dataset [37]. Each video was transcoded via *FFmpeg* [6] and had 30 frames per second.

Figure 1 (top left) shows the accuracy under different bitrates by using the SSD model, a widely used object detection algorithm [26]. It can be seen that the accuracy-bitrate tradeoff varies across different video content types. The accuracy of some videos drops slowly as bitrate decreases, allowing more bandwidth saving in video offloading by degrading the video, whereas other content types incur a rapidly dropping accuracy, making the bitrate down-scaling undesirable. For example, the detector successfully identifies the car even when the bitrate is reduced by seven times to 64 Kbps (Figure 1 middle row left). This is because the monolithic structure of the single car can be shown clearly in the low-bitrate video. However, the performance of "4 bikes" is decreased in 64 Kbps (Figure 1 middle row right) because the structural textures of bicycles in the images are more complicated than the car. Therefore, if we do not treat different videos differently during offloading, the analytics at the server could become ineffective or even fail.

We also show the accuracy under different resolutions in Figure 1 (top right) to make a comparison. We observe that the accuracy-resolution tradeoff is also significantly different across content. More importantly, for a given video, adapting resolution versus adapting bitrate results in highly distinct performance. For example, the accuracy of "small plane" quickly drops with decreasing resolution because the plane becomes highly blurred (Figure 1 bottom). However, this is not true when decreasing the bitrate since the encoder allocates far more bits to the delicate video background than the plane. To meet the overall bitrate reduction, the bits allocated to the background must reduce faster while the plane can keep a relatively stable bit budget, preventing the fast drop of the detection accuracy. Hence, if we fail to adapt the bitrate and resolution jointly, the offloading and analytics performance will be suboptimal.

From these results, we conclude that it is *necessary* to differentiate visual features of videos and pick the optimal offloading bitrate and resolution for each video respectively in order to enhance the analytics accuracy at the server.

4 THE PROPOSED DAO FRAMEWORK

4.1 System Architecture

We first introduce the architecture of DAO as shown in Figure 2, which empowers the system to accomplish bandwidth-adaptive and content-adaptive offloading for video analytics.

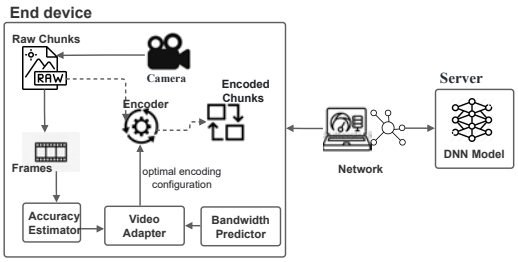

Figure 2: DAO system architecture.

4.1.1 End device. The offloading process begins at the end device when the camera captures a pre-defined number of video chunks. This threshold number depends on the time sensitiveness of the analytics application. Each video chunk is encoded and offloaded using a specific bitrate and resolution based on the visual content features of this chunk. The encoding decision also relates to the predicted upcoming bandwidth from the Bandwidth Predictor that leverages state-of-the-art estimation algorithms.

The principle intelligence of the end device in DAO lies in two modules that optimize the encoding decision, i.e., Accuracy Estimator (Section 4.2) and Video Adapter (Section 4.3). First, the Accuracy

Estimator (AE) models the relationship among video bitrate, resolution, content, and analytics accuracy. Given the key frame of a video chunk and the potential encoding bitrate and resolution, the AE estimates the analytics accuracy of this chunk when it is analyzed by the DNN at the server. Second, along with the predicted bandwidth, the Video Adapter receives the output of the AE to dynamically select the optimal video bitrate and resolution to encode the chunk. Its goal is to maximize the server analytics accuracy. The design of the AE and Video Adapter supports the bandwidth and content adaptation in DAO.

4.1.2 Server. The server of DAO can be an edge or cloud server connected to the end device through a dynamic network. The server is deployed with a DNN model for analytics tasks that cannot be executed in constrained end devices. Once a video chunk is received, the DNN model will perform the analytics task and render the results. Note that we focus on video offloading for popular DNN models already proven fast when deployed on servers, such as object detection and target tracking, i.e., the analytics itself is not the bottleneck, but the network is. We do not aim to expedite the DNN computation which involve other orthogonal research areas.

4.2 Accuracy Estimator

Given the complex relationship presented in Section 3, it is crucial to characterize the relationship between the encoding configuration and the accuracy of a specific video in order to ensure effective offloading. To this end, the proposed Accuracy Estimator (AE) will learn this content-dependent relationship through a CNN due to its advantages of distilling visual content features.

Specifically, the AE receives one raw video chunk as input and estimates a set of accuracy scores for the DNN model deployed at the server. Each estimated accuracy score corresponds to one possible pair of bitrate and resolution considered in the offloading. We fix the video frame rate for all videos because a video retains similar information at different frame rates, resulting in similar DNN model performance.

While providing the entire video chunk as input to the AE is straightforward, doing so increases the computation cost significantly because more video frames have to be processed. This is inappropriate for end devices that have limited computation capabilities. More importantly, video information is temporal redundant across frames. The information in a set of frames can be generalized to a key frame, provided that the frame set is small enough. DAO employs the first frame of each video chunk to be the key frame given the small chunk duration in modern video systems (1 second in our prototype implementation).

4.2.1 Model Design. The architecture of AE is described in Figure 3. The AE has two main parts, a base network and a decision layer. The base network directly explores the pixel intensities of the input frame. The goal is to extract and represent visual features for the input video so that the AE can provide the best accuracy estimate in a content-adaptive way. The output of the base network is feature maps that are used as input to the decision layer. The decision layer estimates the analytics accuracy of the server DNN.

There are many CNN-based architectures with well-designed base networks for feature extraction and representation. We employ

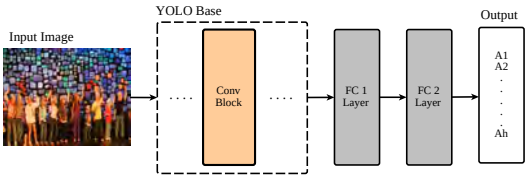

Figure 3: The model architecture of the AE.

the base structure of YOLOv3 [36] that consists of a sequence of Conv Blocks. Each Conv Block includes three convolutional layers with different kernel size and stride. Such a base network is known for its accuracy, robustness, and speed in object detection [39]. This choice ensures the performance of our network as a whole while meeting constraints of computation power at the end devices. We will show the time overhead of AE in Section 5. Furthermore, since object detection is a primitive for many computer vision problems, the proposed base network can extract features effectively for a wide range of DNN analytics tasks.

The decision layer is a stack of two fully connected (FC) layers. It receives the feature maps generated from the base network. The output of the last FC layer includes h neurons, each of which predicts the accuracy of the input video when it is analyzed by the server DNN at one specific encoding configuration. Unlike convolutional layers, which return values corresponding to specific spatial locations from the input frame, FC layers aggregate all input values associated with all input pixels. This helps the model make decisions based on the information from every part of the input frame. Using two FC layers increases the non-linearity of the model, allowing it to learn more complicated decisions.

4.2.2 Model Training. Predicting the accuracy score of an input video corresponding to different encoding configurations is a regression problem. We propose using a loss function that can distill knowledge from the server DNN model and force the AE to behave in a similar way as the server DNN. As a result, the accuracy predicted by the AE would be similar to the actual accuracy provided by the server DNN. Specifically, we use the following loss function. It measures the mean square error (MSE) between the predicted and actual accuracy scores of the DNN model, i.e.,

$$L(A, A') = 1/M \sum_{m=0}^{M} (A(f, b, r) - A'(f, b, r))^2 \quad (1)$$

where M is total number of data samples, $A(f, b, r)$ and $A'(f, b, r)$ are the predicted and ground-truth accuracy for a given frame f under a specific bitrate b and resolution r. The accuracy function $A(\cdot)$ is determined by the DNN model deployed on the server. For example, if the analytics task is object detection, we can select mAP scores to supervise the training. Note that even though the server analytics does not use a DNN model, we can still leverage the most common metric for an algorithm and apply this accuracy loss.

We then utilize transfer learning on the base network of the AE. First, the YOLOV3 model is trained on COCO [22], a large dataset for standard object detection tasks such that the weights of the base network of YOLOV3 can be transferred into the AE. Then the whole network will be trained on our accuracy dataset that will be

presented in Section 5.1. ADAM optimizer is used to update the model parameters.

4.3 Video Adapter

In this section, we introduce the Video Adapter (VA) in DAO, a module that finds the optimal bitrate and resolution that adapt to both content and bandwidth dynamics. We start with formally describing the optimization problem and then propose a practical algorithm to solve the problem.

4.3.1 Problem Formulation. Videos captured by the camera will be chopped into chunks. Each chunk lasts T seconds and is indexed by the variable i. Once chunk i is ready, it will be encoded and offloaded using a specific bitrate $b(i)$ and resolution $r(i)$. The values of the $b(i)$ and $r(i)$ are selected from the set of bitrates $\{b_1, \ldots, b_k, \ldots, b_K\}$ and the set of resolutions $\{r_1, \ldots, r_l, \ldots, r_L\}$, where K and L are the numbers of pre-defined bitrates and resolutions, respectively.

Given the visual content features of chunk i, the analytics accuracy of this chunk under a particular encoding setting, $b(i)$ and $r(i)$, can be estimated by the Accuracy Estimator, i.e., $A(f(i), b(i), r(i))$, where $f(i)$ is the pixel data of the key frame of chunk i. In order to provide high-performance video analytics services at the server, the *goal* of our optimization is to maximize the accuracy scores of the DNN model for the entire video across every video chunk.

Since video chunks are captured by the camera continuously, these chunks must be transported to the server continuously once the offloading starts. The limited storage capacity of the end device makes it desirable to offload video chunks immediately because such data, without timely offloading, may be dropped eventually due to lack of storage. This data loss could lead to zero or near-zero performance at the server DNN for some chunks since the incomplete or missing video chunks may mislead the analytics task, which ultimately causes a decrease in overall accuracy. To prevent this negative effect, a critical *constraint* of the AE optimization is that the chunk bitrate $b(i)$ must be less than the predicted network bandwidth $BW(i)$ when chunk i is offloaded. This would prevent the chunks from piling up and being dropped at the end device.

To derive $BW(i)$, we assume the end device finishes the offloading of chunk $i-1$ at time $t(i-1)$. Once chunk $i-1$ is completely delivered or chunk i becomes available from the camera, whichever comes late, the end device starts to offload chunk i at time $t(i)$. Hence the bandwidth for chunk i can be predicted by

$$BW(i) = \frac{\int_{t(i-1)}^{t(i)} BW_t dt}{t(i) - t(i-1)} \tag{2}$$

In sum, the proposed optimization problem at the VA is formally described as follows.

$$\begin{aligned} \text{maximize:} \quad & \sum_{i=1}^{N} A(f(i), b(i), r(i)) \\ \text{subject to:} \quad & 0 \le b(i) \le BW(i), \qquad \forall i \\ & b(i) \in \{b_1, \ldots, b_k, \ldots, b_K\}, \forall i \\ & r(i) \in \{r_1, \ldots, r_l, \ldots, r_L\}, \quad \forall i \end{aligned} \tag{3}$$

The intuition of (3) is to optimize the encoding configurations across N different video chunks such that maximal analytics accuracy can be achieved at the server while still adapting to the dynamic bandwidth.

4.3.2 Solution Algorithm for VA. Ideally, if the future bandwidths for offloading each chunks $BW(1), \ldots, BW(N)$ are known, one can solve the problem in (3) by a one-time offline computation and obtain the optimal encoding configurations $(\vec{b}, \vec{r})$ for chunk 1 to N. However, it is impossible in practice to acquire such perfect knowledge. Instead, an online decision-making strategy needs to be developed to approximate the theoretically optimal result.

In essence, the combinational optimization problem in (3) can be recognized as a dynamic stochastic control process. There are widely accepted theoretical treatments for this non-trivial problem. Markov Decision Process [33] assumes that the system states, e.g., bandwidth, evolve as a Markov process, and it then probabilistically derives the future bandwidths and makes the decision. However, the Markov properties of bandwidth dynamics have not been proved, and how to obtain a general model for offloading bandwidth is unknown. Receding horizon control [29] assumes a stable state, e.g., bandwidth, within a short period and optimizes the decision within a finite horizon. It then iterates this process to approximate the performance bound. Nevertheless, the search space for each chunk can be ample, degrading the computation efficiency at the end device. In DAO, we propose a simple yet practical greedy algorithm to approximate the optimal solution. We will show in Section 5 that our performance closely approaches the theoretical bound.

Our solution is summarized in the Algorithm 1. The goal is to find and return the encoding configuration $b(i), r(i)$ and estimated accuracy $a(i)$ for chunk i. These variables are initialized to zero. For each chunk i, the algorithm obtains the predicted bandwidth $BW(i)$ and utilizes the AE to estimate the accuracy scores $A(f(i), b_k, r_l)$ by processing the key frame $f(i)$ under various bitrates and resolutions. The algorithm iterates over each combination of bitrates and resolutions and filters out configurations conflicting with the bandwidth limit. Among the acceptable configurations, the algorithm selects the one with the highest accuracy and returns the configuration and predicted accuracy for chunk i.

Algorithm 1: Video Adapter Algorithm

```
for i ∈ {1, ..., N} do
    b(i) ← 0, r(i) ← 0, a(i) ← 0
    Obtain the predicted bandwidth BW(i)
    Access the key frame f(i)
    for k = 1, ..., K  &  l = 1, ..., L do
        if b_k < BW(i)  &  A(f(i), b_k, r_l) > a(i) then
            a(i) ← A(f(i), b_k, r_l);
            b(i) ← b_k;
            r(i) ← r_l;
        else
            continue;
        end
    end
    return (b(i), r(i), a(i))
end
```

Given the algorithm, the VA selects the configuration for each chunk and each round has a complexity of $O(KL)$, where L and K are the numbers of possible bitrate levels and resolution values.

Since K and L are typically small (< 10 in standardized adaptive video transport systems such as DASH), they can be considered constants. Therefore, the VA can run fast with negligible overhead for every chunk decision.

5 EVALUATION

We now evaluate DAO through extensive real-world implementations and experiments. We focus on the server analytics task of object detection because it is one of the most popular video analytics services that is fundamental to many high-level vision applications.

5.1 Experiment Setup

5.1.1 Dataset and AE Training. Our dataset for training the Accuracy Estimator (AE) was derived from Youtube-VOS [2], a large-scale dataset of more than 4,000+ annotated videos for object detection. It covers 90+ semantic categories, and 7800+ unique objects. To ensure the quality of the input video, we selected 320 videos with a 1080p resolution or higher. For each video, we split it into chunks of $T = 1$ second. Each chunk was transcoded into 32 different encoding configurations, with the bitrate ranging from 32 Kbps to 2048 Kbps and the resolution ranging from 240p to 1080p. For each configuration, the DNN-based object detector (SSD-500 [26]) performed object detection on all chunks. This process created a dataset of 18,778 samples. Each sample is a key frame representing a video chunk and the label for the sample is an array of 32 real values that indicate the mAP scores when the video chunk is analyzed at the server under 32 combinations of bitrate and resolution.

After annotation, we divide the dataset into a training set and a test set. We follow the method in Section 4.2 to train the AE with a learning rate of 0.0003 and a weight decay of 1e-5. The number of output neurons of the decision layers is 32 ($h = 32$), corresponding to the number of encoding configurations. The test set consists of 1079 chunks from 21 videos across 20 object categories. The training took less than 5 hours on a single machine with an Intel Xeon Gold 6130 CPU and a 64 GB RAM.

5.1.2 System Implementation. We implemented the end device of DAO on an NVIDIA Jetson AGX Xavier device, which has been widely used as the end device in distributed systems and the Internet of Things. This computing board has a Tensor Cores GPU and an 8-core ARM v8.2 64-bit CPU. The board has an Ubuntu 18.04 system with a JetPack SDK 4.4 to support the development of deep learning. We employed FFmpeg for the video encoding and processing. The AE was trained in Pytorch and ported to the Jetson device. The server and SSD-500 object detector were developed in Python. The server program was deployed on an Ubuntu 18.04 machine with an Intel Xeon Gold 6130 CPU and a 64GB RAM.

We used Wondershaper [3], a bandwidth control tool, to throttle the uplink offloading bandwidth between the server and the end device by using real-world bandwidth traces [23]. These traces were collected when visual data was uploaded from phones to servers through 4G and WiFi. They characterize the unstable and limited offloading bandwidth that is significantly lower than the typical downloading bandwidth in offices and homes. The bandwidth ranges from 144 Kbps to 744 Kbps, with an average of 418 Kbps. By default, we run each experiment using 10 traces and we report the average result.

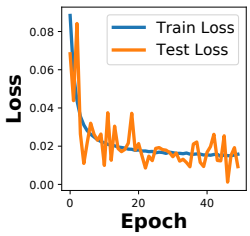

Figure 4: Train and test loss are small for the AE.

	Failure	Success
Precision	0.94	0.80
Recall	0.94	0.82
F1-score	0.93	0.80

Table 1: AE estimation of DNN failure and success.

5.1.3 Baselines. We compare DAO with three baselines.

- Perfect Model: This is the ideal scenario where the Video Adapter (VA) knows the future network bandwidth and always chooses the theoretically optimal bitrate and resolution. It should yield the upper bound of object detection accuracy.
- Constant Baseline: The video has a constant resolution (1080p) and bitrate during an experiment. For each experiment, the constant bitrate is set to be the highest level no greater than 80% of the minimum bandwidth in the trace.
- DASH Baseline: To simulate traditional adaptive video delivery for viewing, the encoding configuration is dynamically selected based on the current bandwidth from the available encoding setting list in standard DASH [1].

5.2 Evaluation Results

5.2.1 Performance of Accuracy Estimator. We first illustrate the convergence of the loss function in the training and testing of the AE in Figure 4. The figure shows that the MSE between the predicted accuracy and the ground truth converges to a close-to-zero value, indicating that the model is well trained.

To further evaluate the performance of the AE, we categorize the accuracy estimation results into two groups. The first group includes the cases when the server DNN fails to detect any objects on the input frame (mAP=0), whereas the second group presents the cases when the server DNN successfully detects all desired objects on the input frame (mAP=100%). We show the precision, recall, and F1 score of the test videos for both groups in Table 1. We observe that the AE correctly predicts the failure case of the DNN on a video 94% of the time (precision), and discovers 94% cases where the DNN would fail (recall). In the second group (success), the precision and recall are around 80%. These results demonstrate that the AE can effectively predict the DNN performance of the offloaded video at the server, which would benefit the adaptive offloading.

5.2.2 Video-wise Analytics Performance. In this section, we investigate the performance of DAO across different videos. Each video is offloaded to and analyzed at the server. The mAP scores achieved at the server for all videos are reported in Table 2. We observe that DAO achieves higher mAP than DASH and Constant on all test videos. This is because DAO can select the best encoding configuration that adapts to both the complexity of the content and the dynamics of the network bandwidth. On the contrary, DASH only pays attention to the bandwidth and Constant is agnostic to both bandwidth and content, both of which results in lower mAP scores. In addition, DAO's performance is close to the theoretical bound achieved by Perfect in all videos. The small gaps mainly stem from

Table 2: Comparison of per-video mAP scores (%) achieved at the server after offloading.

Video Name	Perfect	DAO	Constant	DASH
DogInDryLeaves	68	65	39	13
BigCarAndPeople	69	68	68	48
CarEngineParts	77	75	69	59
TrainTrailView	82	80	71	59
CarsBikesOnRoad	84	84	79	57
PersonRidingHorse	88	84	75	54
KidPlayingCat	86	86	79	81
PersonCatDanceClip	89	87	60	36
CowsGrass	93	89	74	45
CowsRoadCars	91	90	86	35
RoadTrafficBuildings	92	92	88	75
RoadTraffic	96	92	84	57
BigBus	94	93	76	36
BigBusesEngineView	95	94	89	51
PersonKayaking	98	95	94	37
PeopleCyclingRoad	99	96	92	47
DucksLake	97	96	90	75
PersonDogCourt	98	98	95	60
PersonSingInstruments	98	98	98	95
KidsBoxing	99	99	95	69
HousesDogPersons	100	100	95	85

the bandwidth prediction errors, which sometimes causes DAO to choose a sub-optimal encoding. We conclude that DAO effectively supports the content and bandwidth adaptation required in adaptive offloading for video analytics.

5.2.3 Category-wise Analytics Performance. Since each test video has multiple object categories, the video-wise result in Table 2 averages the mAP scores across object categories. To have a different perspective on how DAO adapts to object categories, Table 3 reports the per-category mAP scores. We observe that DAO consistently outperforms DASH and Constant. The average mAP across across object categories for DAO is 59% and 16% higher than DASH and Constant, respectively. DAO also stays close to Perfect in all categories. By contrast, the performance of DASH tends to fluctuate widely. For example, it achieves nearly-perfect scores for "Aeroplane" and "Cat" but poor performance for "Potted plant". This is because DASH lacks awareness of the target object category and cannot adjust the encoding configuration accordingly to ensure the performance on some challenging content.

5.2.4 Bandwidth Usage. This section investigates the bandwidth consumption in the offloading process. We record the size of video data transferred over the network to indicate the offloading bandwidth usage. Figure 5 illustrates the bandwidth usage by each video. We can see that DAO consumes the least amount of bandwidth and remains close to the upper bound. On average, the bandwidth consumption for DAO across the videos is 297.28 Kbps, which is 20% lower than DASH and 33% lower than Constant. The reason why DAO can consume less bandwidth while reaching higher analytics accuracy is that the adaptation of bitrate and resolution could have a distinct effect on the mAP at the server. By understanding how the server DNN would perceive a video chunk, DAO can sometimes reduce the bitrate and increase the resolution to simultaneously increase the analytics accuracy and reduce bandwidth consumption. However, this is not supported in DASH and Constant which do not adapt to the content dynamics for server analytics.

5.2.5 Impact of Different Bandwidth Conditions. We now evaluate the analytics performance achieved at the server under various

Table 3: Comparison of per-category mAP scores (%) achieved at the server after offloading.

Category Name	Perfect	DAO	Constant	DASH
Sofa	56	50	50	50
Bottle	76	73.4	70	41.2
Dinning table	79	75	50	25
Dog	87	81.2	76.7	63.5
Bus	85	83.8	77.8	67.2
TV monitor	88	86	86	28
Car	89	86.5	84.8	69.7
Person	94	91.5	90	57
Bird	93	91.5	81	41.25
Train	94	91.5	78	67.8
Chair	93	92	90	25
Cow	95.6	95.6	89.8	76.8
Boat	96	96	46	87.5
Bicycle	96.5	96.5	96.5	40.5
Motor bike	99.4	99.4	94.6	50
Sheep	100	100	72.3	54.3
Potted plant	100	100	50	0
Horse	100	100	77.2	89.2
Cat	100	100	89	90
Aeroplane	100	100	100	100

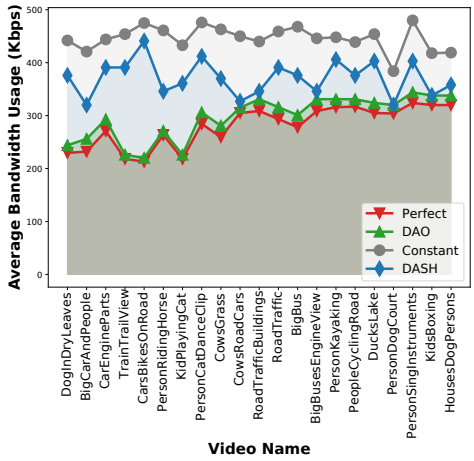

Figure 5: Offloading bandwidth usage by each video.

bandwidth levels. We configured the offloading bandwidth as stable bandwidth at 9 different levels, ranging from 160 Kbps to 800 Kbps by following suggestions in the upload bandwidth study [23]. For each bandwidth level, we perform the offloading for all videos and report the average mAP scores. Figure 6 shows that the mAP scores steadily move up for all systems as the available network bandwidth increases. This is because more available bandwidth could allow a higher-quality video to offload regardless of the adaptation method. The higher-quality video then results in a higher mAP. It can also be seen that DAO and Perfect stay close to each other and achieve the highest mAP scores at all levels. This indicates that DAO performs well across a wide range of offloading bandwidth.

5.2.6 Network Fluctuations. In this section, we study the impact of fluctuating bandwidth. We show the variation of mAP over time when offloading in a 4G and a WiFi network in Figure 7. The mAP results are averaged across all videos. As expected, all systems reach a higher mAP when the bandwidth is abundant. In addition, it is interesting to observe that DAO performs better in lower bandwidth than in higher bandwidth. In a low-bandwidth case, the benefits of joint bitrate and resolution adaptation become more evident

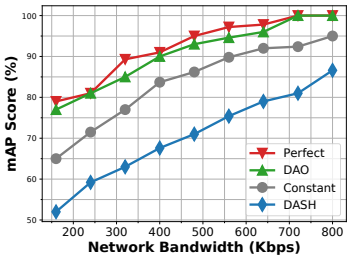

Figure 6: mAP increases as offloading bandwidth increases.

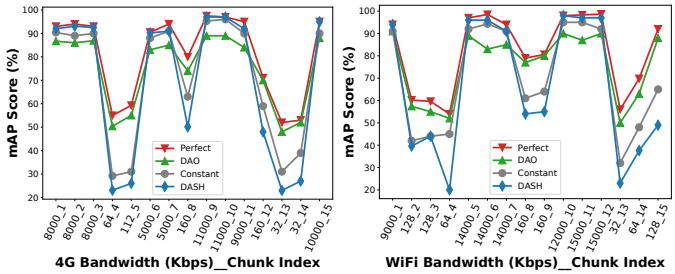

Figure 7: mAP scores over fluctuating 4G and WiFi networks.

because DAO could potentially use a low bitrate and a reasonable resolution to achieve high accuracy while reducing the bandwidth. However, this benefit is diminished in high bandwidth because all systems can opt for a high-bitrate version. Moreover, DAO is designed to optimize the analytics over all chunks, and thus it tends not to be too aggressive when bandwidth is high because it wants to avoid low performance when bandwidth reduces. This result demonstrates the balanced and stable performance of DAO.

5.2.7 Computation Overhead. Since we utilize a CNN-based AE to understand video content and estimate server DNN accuracy, it is vital to examine its computation overhead. We recorded the execution time of the AE throughout experiments and found that the average value is only 26 ms. This is due to our simple CNN architecture with only convolutional layers and FC layers. While it is unlikely to conduct complex DNN analytics on end devices, our result proves the benefits of leveraging simple CNN on end devices and is consistent with the YOLO performance [39]. In addition, we measured the time to run the adaptation algorithm in the VA, and the average value is 0.2 ms. Given that DAO makes an adaptation decision per chunk (rather than per frame), we conclude that the computation overhead of the AE and VA is negligible.

6 DISCUSSION

Advanced Network and System Support for Video Analytics. We observe a small gap between DAO and the theoretical bound in Section 5. This is because our bandwidth predictor and video adapter are fast but sometimes susceptible to prediction errors and sub-optimal performance during transient bandwidth fluctuation. To enhance the system, a learning-based predictor and adapter can be employed to explore a longer history of previous transmissions in order to be both stable against traffic spikes and adaptive to long-term network changes. At the same time, the computation efficiency must be ensured for fast execution on end devices.

Another design choice is the scheduling of video offloading. We currently consider immediate video offloading upon camera capturing. However, if a buffer is added at the end device, video chunks may be selectively held in the buffer based on the predicted bandwidth so that the overall accuracy of the whole video is enhanced.

Moreover, DAO focuses on continuously streaming frames from end devices to servers. As long as frames are delivered in the source video frame rate without data loss, seconds of end-to-end latency are acceptable [45]. Nevertheless, real-time applications may require the video frames to be delivered to the server and analyzed by the DNN in tens or hundreds of milliseconds after camera capturing. In this case, a tradeoff between computation efficiency and system performance must be struck throughout the encoding, offloading, and DNN analytics to meet the real-time requirement.

It is important to note that, as the first adaptive offloading framework that supports both content and bandwidth adaptation, DAO focuses on the most fundamental design. The above directions are open research topics that can be built on top of DAO, each of which would require a careful full-scale study.

Multiple Analytics Tasks and 4K Videos. DAO currently supports a single analytics task at the server. To allow multiple server DNN models simultaneously, we can include multiple columns at the AE's decision layer, each of which is similar to the current AE's last layer and estimates the accuracy of a DNN under various encoding settings. A new AE parameter will be added to index the specific DNN to which the input video is sent. We can then train the multi-task AE in the same way except for using the ground truth of all tasks and the task indexes.

Similarly, DAO considers videos up to 1080p resolution, but 4K resolutions are becoming popular, e.g., for the emerging 360 videos. Offloading 4K videos is more difficult because many small objects on the giant video frames are less evident and encoding them in lower bitrate and resolution as in DAO could destroy the details. To reduce bandwidth while still ensuring the accuracy on 4K videos, a CNN model to crop irrelevant background content can be designed to decrease video resolution. This is possible since background content typically does not affect analytics tasks. The cropping model can be integrated into the AE as a unified model for efficient model execution on the end device.

7 CONCLUSION

This paper takes an important step in dynamic adaptive offloading for video analytics. We propose DAO, the first framework that supports both content and bandwidth adaptation. Thanks to the efficient Accuracy Estimator and Video Adapter, DAO can estimate the accuracy of a video when it is analyzed by a server DNN and then select the optimal bitrate and resolution for the video to offload. Real-world implementation with diverse videos and bandwidth conditions demonstrates the significant bandwidth saving and accuracy improvement of DAO over traditional adaptive video streaming approaches. The core design of joint content and bandwidth adaptation in DAO creates a new context for adaptive video delivery, which can enable a suite of future works studying content perception models and video adaptation algorithms.

8 ACKNOWLEDGEMENT

This work is supported by NSF OAC grants 2151463 and 2144764.

REFERENCES

[1] [n. d.]. DASH. ([n. d.]). https://us.hikvision.com/sites/default/files/tb/tb_bit_rate_chart_120115us_0.pdf

[2] [n. d.]. YouTube-VOS A Large-Scale Benchmark for Video Object Segmentation. https://youtube-vos.org/. Accessed: 2022-04-01.

[3] 2017. The Wonder Shaper. (2017). http://lartc.org/wondershaper

[4] M. Abomhara, O.O. Khalifa, O. Zakaria, A.A. Zaidan, B.B. Zaidan, and A. Rame. 2010. Video Compression Techniques: An Overview. *Journal of Applied Sciences, 10: 1834-1840* 1 (2010).

[5] Shivang Aggarwal, Sibendu Paul, Pranab Dash, Nuka Saranya Illa, Y. Charlie Hu, Dimitrios Koutsonikolas, and Zhisheng Yan. 2020. How to Evaluate Mobile 360 Video Streaming Systems?. In *ACM International Workshop on Mobile Computing Systems and Applications (HotMobile)*.

[6] Fabrice Bellard. https://www.ffmpeg.org. (FFmpeg https://www.ffmpeg.org).

[7] Lahiru D Chamain, Sen-ching Samson Cheung, and Zhi Ding. 2019. Quannet: Joint image compression and classification over channels with limited bandwidth. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*. 338–343.

[8] Lahiru D Chamain, Fabien Racapé, Jean Bégaint, Akshay Pushparaja, and Simon Feltman. 2021. End-to-end optimized image compression for machines, a study. In *2021 Data Compression Conference (DCC)*. 163–172.

[9] Bo Chen, Zhisheng Yan, Hongpeng Guo, Zhe Yang, Ahmed Ali-Eldin, Prashant Shenoy, and Klara Nahrstedt. 2021. Deep Contextualized Compressive Offloading for Images. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*.

[10] Bo Chen, Zhisheng Yan, and Klara Nahrstedt. 2022. Context-aware Image Compression Optimization for Visual Analytics Offloading. In *ACM Multimedia Systems Conference (MMSys)*.

[11] 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 (SenSys '15). 155–168.* 6 (2015).

[12] Samuel F. Dodge and Lina Karam. 2017. A Study and Comparison of Human and Deep Learning Recognition Performance under Visual Distortions. *2017 26th International Conference on Computer Communication and Networks (ICCCN)* (2017), 1–7.

[13] Samuel F. Dodge and Lina Karam. 2019. Human and DNN Classification Performance on Images With Quality Distortions. *ACM Transactions on Applied Perception (TAP)* 16 (2019), 1 – 17.

[14] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. 2020. Server-driven video streaming for deep learning inference. 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 (SIGCOMM)*. 557–570.

[15] Fanyi Duanmu, Eymen Kurdoglu, S Amir Hosseini, Yong Liu, and Yao Wang. 2017. Prioritized Buffer Control in Two-tier 360 Video Streaming. In *ACM Workshop on Virtual Reality and Augmented Reality Network*.

[16] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88, 2 (2010), 303–338.

[17] Leonardo Galteri, Marco Bertini, Lorenzo Seidenari, and Alberto Del Bimbo. 2018. Video compression for object detection algorithms. In *2018 24th International Conference on Pattern Recognition (ICPR)*. 3007–3012.

[18] Song Han, Huizi Mao, and William J Dally. 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In *International Conference on Learning Representations (ICLR)*.

[19] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2016. Low Bandwidth Offload for Mobile AR. In *ACM International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*.

[20] Jean Le Feuvre and Cyril Concolato. 2016. Tiled-based Adaptive Streaming Using MPEG-DASH. In *ACM Multimedia Systems Conference (MMSys)*. 41:1–41:3.

[21] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran. 2014. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications* 32, 4 (2014), 719–733.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. 740–755.

[23] Fang Liu, Yeting Guo, Zhiping Cai, Nong Xiao, and Ziming Zhao. 2019. Edge-enabled disaster rescue: a case study of searching for missing people. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 6 (2019), 1–21.

[24] 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.

[25] Qiang Liu, Siqi Huang, Johnson Opadere, and Tao Han. 2018. An Edge Network Orchestrator for Mobile Augmented Reality. In *IEEE International Conference on Computer Communications (INFOCOM)*.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*.

[27] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. 2019. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11006–11015.

[28] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 197–210.

[29] Jacob Mattingley, Yang Wang, and Stephen Boyd. 2011. Receding horizon control. *IEEE Control Systems Magazine* 31, 3 (2011), 52–65.

[30] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D. Beshay, and Ravi Prakash. 2017. Adaptive 360-Degree Video Streaming using Scalable Video Coding. In *ACM International Conference on Multimedia (MM)*.

[31] Viet-Anh Nguyen, Yap-Peng Tan, and Weisi Lin. 2008. Adaptive downsampling/upsampling for better video compression at low bit rate. In *2008 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1624–1627.

[32] Chrisma Pakha, Aakanksha Chowdhery, , and Junchen Jiang. 2018. Reinventing video streaming for distributed vision analytics. *In Proceedings of the 10th USENIX Conference on Hot Topics in Cloud Computing (HotCloud'18)* 5 (2018).

[33] Martin L Puterman. 1990. Markov decision processes. *Handbooks in operations research and management science* 2 (1990), 331–434.

[34] Yanyuan Qin, Shuai Hao, Krishna R Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue. 2019. Quality-aware strategies for optimizing ABR video streaming QoE and reducing data usage. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 189–200.

[35] Xukan Ran, Haoliang Chen, Xiaodan Zhu, Zhenming Liu, and Jiasi Chen. 2018. DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics. In *IEEE International Conference on Computer Communications (INFOCOM)*.

[36] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *ArXiv* abs/1804.02767 (2018).

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

[38] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2018. Multi-path multi-tier 360-degree video streaming in 5G networks. In *ACM Multimedia Systems Conference (MMSys)*. 162–173.

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[40] H. To, S. H. Kim, and C. Shahabi. 2015. Effectively crowdsourcing the acquisition and analysis of visual data for disaster response. *IEEE International Conference on Big Data (Big Data), pp. 697-706,* 11 (2015).

[41] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. 2016. Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond. *IEEE/CAA Journal of Automatica Sinica* 3, 2 (2016), 113–120.

[42] Xiufeng Xie and Kyu-Han Kim. 2019. Source compression with bounded dnn perception loss for iot edge computer vision. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.

[43] Zhisheng Yan and Chang Wen Chen. 2016. RnB: Rate and Brightness Adaptation for Rate-Distortion-Energy Tradeoff in HTTP Adaptive Streaming over Mobile Devices. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*.

[44] Shuochao Yao, Yiran Zhao, Huajie Shao, ShengZhong Liu, Dongxin Liu, Lu Su, and Tarek Abdelzaher. 2018. Fastdeepiot: Towards understanding and optimizing neural network execution time on mobile and embedded devices. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. 278–291.

[45] Jun Yi, Md Reazul Islam, Shivang Aggarwal, Dimitrios Koutsonikolas, Y Charlie Hu, and Zhisheng Yan. 2020. An Analysis of Delay in Live 360 Video Streaming Systems. In *ACM International Conference on Multimedia (MM)*.

[46] Jun Yi, Shiqing Luo, and Zhisheng Yan. 2019. A Measurement Study of YouTube 360° Live Video Streaming. In *ACM Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*.

[47] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. 2017. Lavea: Latency-aware video analytics on edge computing platform. In *ACM/IEEE Symposium on Edge Computing (SEC)*.

[48] Tan Zhang, Aakanksha Chowdhery, Paramvir (Victor) Bahl, Kyle Jamieson, and Suman Banerjee. 2015. The Design and Implementation of a Wireless Video Surveillance System. *In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15). 426–438.* 7 (2015).