IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 18, 2020, accepted June 11, 2020, date of publication June 24, 2020, date of current version July 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3004571

Lightweight Online Profiling-Based Configuration Adaptation for Video Analytics System in Edge Computing

WOO-JOONG KIM, (Member, IEEE), AND CHAN-HYUN YOUN, (Senior Member, IEEE)
School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

Corresponding author: Chan-Hyun Youn (chyoun@kaist.ac.kr)

This work was supported in part by the Cross-Ministry Giga KOREA Project grant through the Korean Government (MSIT) (Development of Mobile Edge Computing Platform Technology for URLLC Services) under Grant GK20P0400, and in part by Electronics and Communications Research Institute (ETRI) & National Research Council of Science and Technology (NST) grant funded by the Korea government [20ZS1100, Core Technology Research for Self-Improving Integrated Artificial Intelligence System], and in part by Samsung Electronics, Device Solution (DS).

ABSTRACT Video Analytics System has emerged as a promising technology to realize deep neural network based intelligent applications for video streams. Its objective is to maximize the video analytics performance of video streams, such as accuracy, while utilizing the limited computing resource capacity efficiently. Existing video analytics systems attempt to adapt configurations to optimize resource-accuracy tradeoffs under limited resource capacity. Especially, several works recently propose a configuration adaptation algorithm based on online profiling to overcome the inefficiency of one-time offline profiling caused by the dynamics of the configuration's impact on video analytics accuracy. However, their systems are inefficient or limited to process the video analytics of multiple video streams in a GPU-enabled edge server. Furthermore, their online profiling methods still leads to a high profiling cost. In this paper, we design a video analytics system to adapt configurations for optimizing the resource-accuracy tradeoffs of multiple video streams with respect to frame rate and resolution fully under limited resource capacity of a GPU-enable edge server. In addition, we propose a lightweight online profiling method, utilizing the underlying characteristics of the video objects. Then, based on it, we propose a configuration adaptation algorithm to find the best configuration of each video stream and minimize accuracy degradation of multiple video streams under limited resource capacity of a GPU-enable edge server. To evaluate the proposed algorithm, we use a subset of surveillance videos and annotations from VIRAT 2.0 Ground dataset. The experimental results show that, in a GPU-enabled edge server, our video analytics system achieves the optimal configuration adaptation on resource-accuracy tradeoffs and our algorithm reduce the profiling cost of existing systems significantly while achieving the video analytics performance comparable to them.

INDEX TERMS Video analytics, deep neural networks, object detection.

I. INTRODUCTION

video cameras are pervasively deployed on streets or in buildings [1], [2], [6]. In this trend, many cities and companies demand to utilize the potential of these cameras for various intelligent applications leading to smart city and company. Video analytics is a key to achieve this demands, consisting of a pipeline of computer vision components [15]. Video analytics systems that analyze multiple video streams individually or collectively from large camera deployments at an edge or cloud environment have been used in many application scenarios, (e.g. traffic congestion management such as counting cars, crime prevention such as reading the license plates) [3], [14], [17]. A desired video analytics quality (i.e. accuracy) varies depending on the application to be realized on video streams [11]. Recently, Deep Neural Network (DNN) has emerged as a powerful advanced tool due to their innovative accuracy in visual analyses such as human action recognition, object detection, scene geometry estimation, face recognition, etc. Applying DNN to video streams is prohibitively

The associate editor coordinating the review of this manuscript and approving it for publication was Xu Chen.

expensive [16]. For example, detecting objects in video is a core function for many application scenarios but the running time of the best DNN based object detector [4] is 198ms per image (5fps) on a K40 GPU. Even the running time of a real-time DNN based object detector [5] is 50ms per image (20fps) on a Pascal Titan X GPU. Therefore, video analytics requires a large cost to operate compute hardware (i.e. GPUs) or edge/cloud resources (i.e. GPU machine time) [14].

Meanwhile, Mobile Edge Computing (MEC) has been identified as a promising environments for the video analytics system to support low-latency with reasonable compute power [12]. In this regards, analyzing multiple video streams efficiently with limited computing resources of an edge-server while guaranteeing a desired video analytics quality has been a challenge issue in the video analytics system [11]. Unfortunately, what resources will be available to each video stream is often unknown. Instead, it is assumed that each video stream is typically analyzed in isolation with either infinite resources or a predetermined resource allocation. When the video analytics processes on multiple video streams are run concurrently, resource competition forces the video stream to be analyzed at a lower frame rate as frames are dropped-leading to unsatisfactory video analytics quality [18].

One of the promising solutions to address this issue is adapting configurations optimally in the video analytics system [10], [11]. The video analytics system has several controllable factors on each video stream, called as *knob*, which have an effect on both resource-consumption and accuracy, such as frame resolution and frame sampling rate. A *configuration* is defined as a combination of the knob values. If the expensive configuration such as high frame resolution or high frame rate is chosen, it shows high accuracy but demands high resource-consumption. On the other hand, if the cheap configuration such as low frame resolution or low frame rate is chosen, it shows low accuracy but demands low resource-consumption. Video streams have a variety of resource-accuracy tradeoff for video analytics depending on their context. Some video streams can achieve a desired video analytics quality with only small resources while others can do it with sufficient resources. Thus, based on this characteristics, the video analytics systems attempt to minimize the resource demands of video streams optimally by adapting their configurations to the best which guarantees a desired video analytics quality in minimal resources, considering their resource-accuracy tradeoff. Recently, there have been several previous works on how to design an efficient video analytics system based on configuration adaptation. Most of these previous works search a large space of configurations by empirically profiling the resource-accuracy tradeoffs of video streams for video analytics in off-line (one-time) or on-line (periodic) manner [10], [11].

However, their works have the limitation of existing video analytics systems which are inefficient or limited to process the video analytics of multiple video streams in a GPU-enabled edge server. They cannot benefit from the potential to efficiently enhance the video analytics performance of video streams based on their resource-accuracy tradeoffs with respect to frame rate. In addition, their online profiling methods require a high profiling cost. Especially, even though Chameleon [10] proposes several techniques leveraging the spatial/temporal correlations to reduce the profiling cost, its profiling cost is still a few times higher than its inference cost so that the cost savings gained by its proposed algorithm is meaningless. The high profiling cost still makes it difficult for video analytics systems to guarantee real-time video analytics and scale video analytics to massive camera deployments. It is necessary to avoid the one-time or periodic search approach on a large space of configurations.

In order to resolve these problems of previous works, we propose a video analytics system to achieve the optimization on the resource-accuracy tradeoffs of multiple video streams with respect to frame rate and resolution fully in a GPU-enabled edge server by controlling configurations. We propose a lightweight online profiling method to generalize the accuracy model on configurations efficiently in real-time. Utilizing the underlying characteristics of the video objects (i.e. the velocity on location and size), which can be extracted in low-cost from the analytics results of the short-term profiling process, the proposed method estimates the accuracy model reflecting how the accuracy on configurations is changed depending on the velocity of objects on location and size in a video stream. We also propose the lightweight online profiling based configuration adaptation algorithm to find the best configuration of each video stream and minimize accuracy degradation of each video stream in the constrained computing capacity of a GPU-enabled edge computing.

II. PROBLEM DESCRIPTION FOR EFFICIENT VIDEO ANALYTICS SYSTEM

In this section, we describe the existing video analytics system of previous works which optimizes the resource consumption and accuracy/latency of video analytics by adapting configurations. We also describe the existing profiling methods of previous works on configurations to find an optimal resource-accuracy tradeoff. Especially, we describe the existing online profiling method, in detail, considering the dynamics of the configuration's impact on video analytics accuracy. Then, we address the limitation of the existing video analytics system and the existing profiling method, utilized by previous works, when applied to the limited resource capacity of a edge-server.

A. LIMITATION OF EXISTING VIDEO ANALYTICS SYSTEMS

Many previous works attempted to optimize the resource-accuracy tradeoff by adapting the configuration of video streams for video analytics under limited resource capacity such as MEC or clusters environment. The problem of optimizing the resource-accuracy tradeoff is first introduced in [6]. VideoStorm [11] processes various video analytics queries on live video streams, only considering the

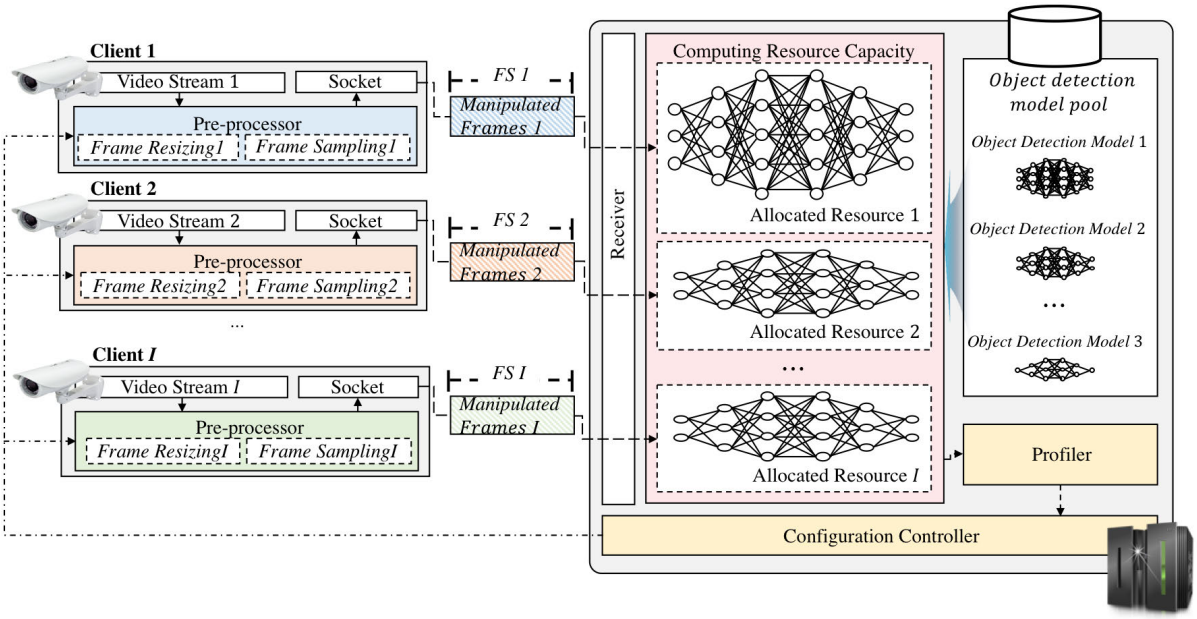


FIGURE 1. An illustration of the existing video analytics systems [20], [21].

CPU resource in a single cluster. VideoEdge [15] considers network and compute resources or dynamic network bandwidths in a hierarchy of clusters for video analytics. DeepDecision [12] focus on client-side analytics scheduling in edge computing environment, considering the complex interaction among model accuracy, video quality, battery constraints, network data usage, and network conditions. Without modeling the accuracy or latency w.r.t configurations in analytic form, they formulate the video analytics optimization. Then, they just propose a heuristic algorithm in which the empirically measured value on the tradeoff is used as it is. Meanwhile, Edge Network Orchestrator [20], DARE [21] and the authors of [22] present and use the analytical models of accuracy and latency w.r.t configurations in order to formulate the video analytics optimization at the network edge. Edge Network Orchestrator only considers resolution as a knob, not considering frame rate (i.e. It use a fixed sampling rate). DARE and the authors of [22] take both frame resolution and rate as knobs.

Figure 1 shows the existing video analytics systems of previous works. There are I multiple clients, each of which generates a video stream by its camera and transmits frames manipulated based on a configuration by its pre-processor. The configuration controller determines the configurations of I multiple video streams fairly based on the resource-accuracy tradeoff provided by the profiler under limited resource capacity. The configuration knobs are usually composed of frame rate and resolution and especially, the frame resolution knob is simply used to decide the object detection model of which input size is matched to it. Then, for each video stream, the configuration controller allocates the resource which its decided object detection model demands and loads its decided object detection model. Finally, the system provides the video analytics results of identifying and classifying objects on video streams in high accuracy/latency under limited resource capacity. Note that it assume the computing resource capacity of an edge server can be divided and allocated to each video stream depending on its resource demand. It also assume the allocated resource of each video stream can independently work for its video analytics in parallel. Then, the resource saved by decreasing the frame resolution of a video stream (i.e. deciding a cheap object detection model) and degrading its performance slightly based on the resource-accuracy tradeoff can be utilized to enhance the video analytics performance of other video streams significantly. That is, with respect to frame resolution, optimizing the resource-accuracy tradeoffs of multiple video streams under limited resource capacity of an edge-server is possible.

However, with respect to frame rate, it is not possible in this system. Structurally, it cannot utilize and share the non-use time of a video stream's resource obtained by decreasing its frame rate based on the resource-accuracy tradeoff. That is, it cannot benefit from the potential to enhance the video analytics performance of other video streams significantly by decreasing the frame rate of a video stream and degrading its performance slightly. Furthermore, this system causes significant inefficiencies within a GPU, which is the dominant resource for DNN processing [10]. A GPU is limited to load multiple video streams's DNN based object detetcion models concurrently, which require high resource demands, due to its insufficient resource capacity on memory and computation. That is, extending GPUs is required as the number of video streams to be processed increases, but is not suitable to a GPU-enabled edge server which can only have

a few GPUs. It is necessary to maximize the efficiency of a GPU for video analytics. Especially, in a GPU, the allocated resources of video streams cannot independently work each other in practice due to interference which degrades each other's performance. In addition, this system requires the reload of DNN-based object detection model to a GPU with configuration adaptation, which causes high overhead and degrades the video analytics performance significantly.

Resolving the limitation of previous works, we propose a video analytics system to divide and allocate the usage time of a GPU, as a resource, to each video stream in each time slot. A GPU is entirely occupied by a video stream at a time [24], so is sequentially used by video streams based on their allocated usage time. In this structure, the non-use time of a GPU, saved by decreasing the frame rate and resolution of a video stream and degrading its performance based on the resource-accuracy tradeoff, can be utilized to enhance the video analytics performance of other video streams significantly. As a result, it makes it possible to achieve the optimization on the resource-accuracy tradeoffs of multiple video streams with respect to frame rate and resolution fully.

B. LIMITATION OF EXISTING PROFILING METHODS

Most of previous works [11], [12], [15], [20]–[22] profile the resource-accuracy tradeoff with respect to each knob for video analytics in off-line. They exhaustively profile all or part of possible configurations on video queries only once at the beginning of the video stream in offline. Then, based on the profiles, they chooses and adjusts the optimal configuration for the whole duration of the video. However, they do not consider and handle the dynamics of the configuration's impact on video analytics accuracy. Even in a identical video, the accuracy characteristics can be changed in real-time depending on how video stream content represent (e.g. objects in a video is moving fast or slow). Its configuration adjusted by offline profiling cannot be optimal in the entire duration.

Meanwhile, several previous works [10], [22] address the dynamics of the configuration's impact on video analytics accuracy. The video analytics system in [22] profiles several accuracy models with respect to frame rate and resolution under three different conditions (i.e. the speed and size of objects at different three levels) in off-line and periodically update the accuracy model at the beginning of each time slot according to the current video content (i.e. the size and the velocities of targets). However, this heuristic approach cannot generalize the accuracy model varying depending on the underlying characteristics of the video objects and may cause an error that cannot be ignored, in many scenarios. The reason is that the accuracy model of a video stream cannot be determined only by the size and the velocities of objects. A lot of factors, such as the type of object within a class, how object look like, the number of objects and so on, also affect the accurcy model besides the size and velocity of objects. For two different times in the same video stream, the accuracy model w.r.t resolution may not be same even if they have the same object size.

Chameleon [10] also propose an efficient real-time re-profiling technique to adapt its configuration optimally, considering the dynamics of the configuration's impact on video analytics accuracy. Figure 2 shows how Chameleon works. Chameleon operates periodically based on w segments (by default, $w = 4$), each of which is a contiguous set of frames spanning a T second interval (by default, $T = 4$). Chameleon observes that the underlying characteristics of the video objects that affect accuracy tend to be consistent temporally, called temporal correlation. Based on it, in each segment, Chameleon profiles the accuracy on possible configurations for the first t_p seconds (by default, $t_p = 1$) and does inference with the best configuration for the remaining $t_a = T - t_p$ seconds. It profiles the accuracy of every possible configurations only in the first segment and then, profiles the accuracy of the first segment's top-k ranked configurations (cheapest k configurations with accuracy above the desired threshold, by default, $k = 5$) over the remained segments. However, this solution is still so expensive and cannot guarantee real-time processing. Since its profiling method is based on direct measurement, it is required to perform a significant number of video analytics executions to profile at the beginning of each segment. The profiling cost of Chameleon is even much higher than its inference cost. We conduct an experiment to measure the profiling and inference cost of Chameleon for video analytics on VIRAT 2.0 ground dataset. We construct four video streams for our experiments using the fully-annotated 20 videos of VIRAT 2.0 Ground dataset, captured by four video cameras deployed in different area, of which frame resolution and rate are 30 fps and 1920 × 1080p (for 3 video streams) or 1280 × 720p (for 1 video stream). Each video stream is composed of 10800 frames and its objects mainly consist of car and person. We use pre-trained object detection models, Yolov3-320, 416, 608 [5] of which input size are 320 × 320, 416 × 416 and 608 × 608, on our workstation with GTX 1080 GPU. We use the sets of selectable values on frame resolution knob and rate knob: $\{608, 416, 320\}p$ and $\{30, 15, 10, 6, 5, 3, 2, 1\}fps$. The cost is measured by GPU processing time. As shown in Figure 3, the results shows that the profiling cost of Chameleon is more than about seven times bigger than its inference cost.

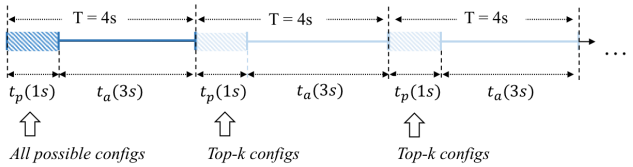


FIGURE 2. Configuration adaptation of Chameleon [10].

In order to resolve these problems of previous works, we propose a lightweight online profiling method to generalize the accuracy model on configurations efficiently in real-time. The proposed method handles profiling process in analytics process concurrently by allocating the short-term

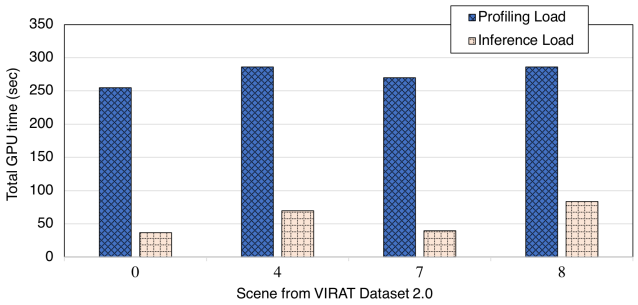


FIGURE 3. Comparison of inference cost and profiling cost in Chameleon.

profiling process to each segment in the analytics process, while Chameleon handles profiling process and analytics process separately. Utilizing the underlying characteristics of the video objects (i.e. the velocity on location and size), which can be extracted in low-cost from the analytics results of the short-term profiling process, the proposed method estimates how the objects change in a segment. Based on it, leveraging the temporal correlations, the proposed method estimates the accuracy model on configurations in subsequent segments, while Chameleon directly measures it. The estimated accuracy model reflects how the accuracy on configurations is changed depending on the velocity of objects on location and size in a video stream.

III. PROPOSED VIDEO ANALYTICS SYSTEM FOR A GPU-ENABLED EDGE SERVER

In this section, we describe our target edge computing environment and video analytics system with several assumptions and conditions. Then, we define and resolve the problem with an objective function to maximize the video analytics performance of multiple video streams in a GPU-enabled edge server by controlling configurations.

A. SYSTEM MODEL AND ENVIRONMENT

Figure 4 shows our target edge computing environment and video analytics system which supports multiple video streams efficiently in a GPU-enabled edge server. There are I multiple clients, each of which generates a video stream by its camera in the default frame rate and resolution. Let dfs, dfr are the default frame rate and resolution of all video streams.[1] In the same way as Chameleon, each video stream is split into smaller segments, each of which is a contiguous set of frames spanning a T-second interval (by default, $T = 4$). Let $S_i = \{S_{i,1}, S_{i,2}, \ldots, S_{i,j}, \ldots, S_{i,J}\}$ denote the segments of i-th video stream of which size is J. The frames of $S_{i,j}$ are denoted as $S_{i,j} = \{f_{i,j}^1, f_{i,j}^2, \ldots, f_{i,j}^{l_s}\}$, where the number of frames in $S_{i,j}$ is $l_s = T * dfs$. In each segment of i-th video stream, we define the sub-segments, each of which is a contiguous set of dfs frames.

In a client, the pre-processor samples the frames of each segment with a certain frame rate and resizes each sampled frame in a certain frame resolution. These frame rate and resolution are considered as main configuration knobs and controlled by the configuration controller. Let $fr_{i,j}$ and $fs_{i,j}$ denote the frame resolution and rate of i-th video stream

[1] We assume that the default frame rate and resolution of all video streams are same

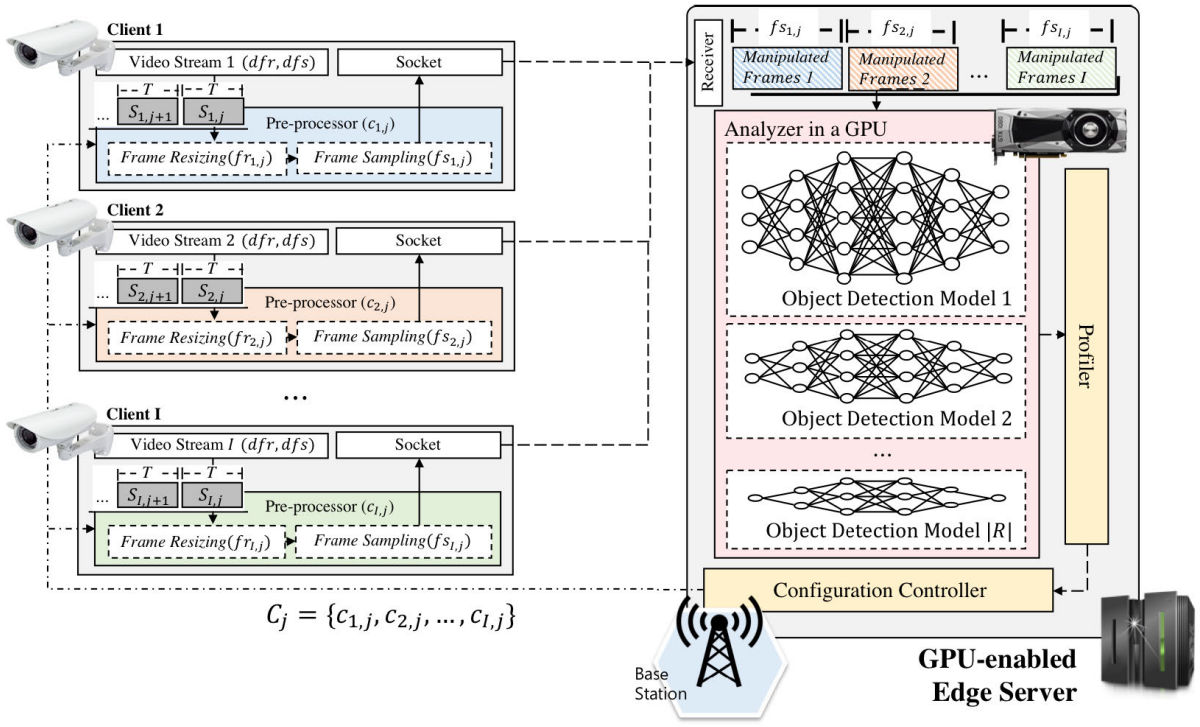


FIGURE 4. An illustration of the proposed video analytics system for a GPU-enabled edge server.

for j-th segment. There are the sets of selectable values on frame rate knob and resolution knob which the configuration controller can choose. They are denoted as R and S, respectively, and are sorted in a descending order (e.g. $R = \{608, 416, 320\}p$ and $S = \{30, 15, 10, 6, 5, 3, 2, 1\}fps$). For simplicity, dfr and dfS are defined as the maximum value of R and S, respectively. We use the divisors of dfs for S. The input sizes of available object detection models in a GPU are set as R, so the frame resolution knob is simply used to decide the object detection model of which input size is matched to it. The manipulated frames of j-th segment in each video stream are transferred to our video analytics system via wireless link.

Our video analytics system consists of an analyzer, a profiler and a configuration controller [10]. It is deployed on an GPU-enabled edge server directly connected to a small cell base station via a high speed link. We assume the edge server has a single GPU, so the scheduling for multi-GPUs is out of the scope of this paper. Let $[n] = 1, \ldots, n$ denote the set of integer numbers which has a cardinality of n.

The manipulated frames of j-th segment in each video stream are fed into the queue and processed for video analytics by the analyzer. The analyzer is a GPU loading all of available object detection models, which provide different accuracy and processing time with different input size and network structure. We utilize the fact that only a few object detection models are used in an edge based video analytics system and are enough to be loaded to a single GPU. The analyzer is shared by multiple video streams in a unit of usage time. In this paper, we do not consider the parallelism of different object detection models loaded in a GPU. Therefore, it is entirely occupied by a video stream at a time, so is sequentially used by video streams based on their allocated usage time. Then, in each time slot, we allocate the usage time of the analyzer as a resource to each video stream depending on its configuration. In each time slot, the total usage time of video streams have not to exceed a time slot, which acts as a limited resource capacity, to guarantee the real-time processing without accumulating lag.

The resolution of each frame waiting in the queue is matched to the input size of one of the object detection models and it is fed into its matched model. Lastly, the model recognizes objects such as car or person and draws bounding boxes around them in the frame. The frames can be processed in a batch but we do not consider the batch processing in this paper (i.e. batch size is 1).[2] Thus, each manipulated frame is processed as a task sequentially [25].

The configuration controller periodically decides the configurations of video streams on each segment, based on the configuration adaptation algorithm. The configuration of i-th video stream for j-th segment is defined as $c_{i,j} = \{fs_{i,j}, fr_{i,j}\}$. Let $C_j = \{c_{1,j}, c_{2,j}, \ldots, c_{I,j}\}$ denote the configurations of I video streams for j-th segment, decided by the configuration controller. The profiler profiles the video analytics accuracy and latency of configurations and provides them to the configuration adaptation algorithm. Let $a_{i,j}(c_{i,j})$ and $l_{i,j}(c_{i,j})$ are the accuracy and latency model of i-th video stream for j-th segment depending on $c_{i,j}$.

[2] In this paper, the goal is to deliver the analytics result of each frame as soon as possible to guarantee real-time requirement. Batching leads to higher processing latency in the inference phase [24]

B. ANALYTICS ACCURACY MODEL

For analytics accuracy model, we measure how frame resolution and rate affect the video analytics performance on analytics accuracy. The points in a line represents the measured accuracy on a knob over its available values while fixing the other knob to its most expensive value. For this measurement, we use the experimental setting of section II-B. We use average F1 score per frame as the metric of accuracy. F1 score is the harmonic mean of precision and recall. We identify true positives for the F1 score by checking if the bounding box has the same label and the spatial overlap above $\eta_o = 0.5$ with ground truth boxes [10]. In order to measure the average F1 score per frame, we measure the average F1 score of each segment depending on a configuration $c = \{fs, fr\}$, which is calculated as follows. To simplify notations, we omit the mention of the video stream index i. Let $B^{'fr}_{j,q} = \{b'^1, b'^2, \ldots, b'^h, \ldots\}$ denote the bounding boxes of objects detected in the q-th frame of j-th segment which is manipulated (i.e. sampled and resized) by frame rate fs and frame resolution fr. b'^h is the h-th object's bounding box. The bounding box b'^h is composed of the 4-dim coordinates, left, top, width and height, denoted as $\{x'_h, y'_h, w'_h, h'_h\}$. Based on the detection results of manipulated frames for j-th segment of a video stream, the detection result of objects for all frames of j-th segment including frames not sampled are predicted and finalized. Let $B_{j,q}$ denote the final bounding boxes of objects on q-th frame of j-th segment in a video stream. To predict the detection results of frames not sampled, we use the bounding boxes of objects detected from the latest manipulated frame as follows:

$$B_{j,q} = B^{'fr}_{j,(d-1)\delta+1} \quad \forall\,(d-1)\delta < q \le d\delta, \quad \forall d \in [T * fs], \quad where \;\; \delta = \left\lceil \frac{dfs}{fs} \right\rceil. \tag{1}$$

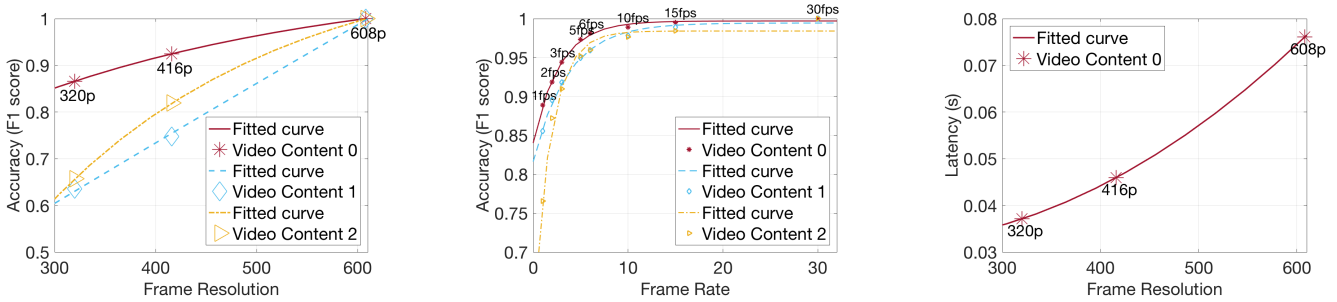
Here, δ is the interval between manipulated frames in j-th segment and d is the index of the interval in j-th segment.

In addition, we use the detection results of the golden configuration $c_{max} = \{dfs, dfr\}$ (i.e. the most expensive configuration) for j-th segment as "ground truth", following prior works [10], [11].

$$G_{j,q} = B'^{dfr}_{j,(d-1)\delta+1} \quad \forall\,(d-1)\delta < q \le d\delta, \quad \forall d \in [T * dfs], \quad where \;\; \delta = 1. \tag{2}$$

We calculate the average F1 score of j-th segment on c in a video stream with $B_{j,q}$ and $G_{j,q}$, defined as follows:

$$a^{\dagger}_j(c) = \frac{1}{l_s} \sum_{q=1}^{l_s} f1(B_{j,q}, G_{j,q}). \tag{3}$$



(a) Frame Resolution (b) Frame Rate (c) Frame Resolution

FIGURE 5. Impact of configuration knobs on the video analytics accuracy and latency according to frame resolution and rate.

Finally, we calculate the average F1 score per frame on c in a video stream, defined as follows:

$$a^{\dagger}(c) = \frac{1}{J}\sum_{j=1}^{J} a_j^{\dagger}(c)\ . \tag{4}$$

Figure 5(a)-(b) shows the accuracy results on frame resolution and rate over R and S for three video streams among four video streams. In Figure 5(a), the points of a line represent $a^{\dagger}(c), \forall fr \in R, fs = dfs$ in a video stream. Fitting the measured accuracy points of i-th video stream on frame resolution, the function $\epsilon_i(fr)$ is empirically formulated as a concave exponential function of three coefficients to represent the accuracy function with a continuous variable fr of frame resolution for i-th video stream [22]. It reflects the fact that a higher frame resolution produces a better analytics accuracy and the analytics accuracy gain decreases at a high frame resolution.

$$\epsilon_i(fr) = \alpha_{i,1} - \alpha_{i,2} * e^{-fr/\alpha_{i,3}}. \tag{5}$$

Here, $\{\alpha_{i,1}, \alpha_{i,2}, \alpha_{i,3}\}$ are known constant coefficients of i-th video stream.

In Figure 5(b), the points of a line represent $a^{\dagger}(c)$, $\forall fs \in S, fr = dfr$ in a video stream. Fitting the measured accuracy points of i-th video stream on frame rate, the function $\phi_i(fs)$ is also empirically formulated as a concave exponential function of three coefficients to represent the accuracy function with a continuous variable fs of frame rate for i-th video stream [22]. It reflects the fact that a higher frame rate provides a better analytics accuracy and the analytics accuracy gain decreases at a high frame rate.

$$\phi_i(fs) = \beta_{i,1} - \beta_{i,2} * e^{-fs/\beta_{i,3}}. \tag{6}$$

Here, $\{\beta_{i,1}, \beta_{i,2}, \beta_{i,3}\}$ are known constant coefficients of i-th video stream.

We obtain the constant coeffcients of four video streams in the experimental setting with less than 0.03 root mean square error (RMSE), as listed in Table 1.

Using $\epsilon_i(fr)$ and $\phi_i(fs)$, we derive the analytics accuracy model of each video stream with the configuration c of continuous variables on frame resolution fr and rate fs. As experimentally observed in [10], [22], we confirm that frame resolution and rate independently impact accuracy. Based on the observation, the analytics accuracy model of i-th video stream can be defined as

$$a_i(c) = \epsilon_i(fr) \cdot \phi_i(fs). \tag{7}$$

TABLE 1. Constant Coeffcients used in Equation 7.

Functions	Constant Coeffcients	RMSE
ϵ_1	$\{\alpha_{1,1}; \alpha_{1,2}; \alpha_{1,3}\} = \{1.091; 0.6166; 0.5218\}$,	0.002812,
ϕ_1	$\{\beta_{1,1}; \beta_{1,2}; \beta_{1,3}\} = \{0.9971; 0.1568; 0.09202\}$	0
ϵ_2	$\{\alpha_{2,1}; \alpha_{2,2}; \alpha_{2,3}\} = \{6.841; 6.564; 8.568\}$,	0.01135,
ϕ_2	$\{\beta_{2,1}; \beta_{2,2}; \beta_{2,3}\} = \{0.9823; 0.6911; 0.06176\}$	0
ϵ_3	$\{\alpha_{3,1}; \alpha_{3,2}; \alpha_{3,3}\} = \{9.236; 9.032; 10.88\}$,	0.004225,
ϕ_3	$\{\beta_{3,1}; \beta_{3,2}; \beta_{3,3}\} = \{0.9945; 0.1782; 0.1214\}$	0
ϵ_4	$\{\alpha_{4,1}; \alpha_{4,2}; \alpha_{4,3}\} = \{1.146; 1.869; 0.3926\}$,	0.01119,
ϕ_4	$\{\beta_{4,1}; \beta_{4,2}; \beta_{4,3}\} = \{0.9842; 0.3627; 0.06262\}$	0
P	$\{p_1; p_2; p_3\} = \{0.08315; -0.04463; 0.03757\}$	0

C. SERVICE LATENCY MODEL

The service latency consists of three parts: transmission latency, computation latency and static latency. On each part in the service latency, we use average latency per frame.

The transmission latency is the average latency per frame to transfer the data generated and manipulated from a video stream to the video analytics system. The transmission latency of i-th video stream with the configuration c of continuous variables on frame resolution fr and rate fs is defined as follows:

$$l_i^T(c) = \frac{S(c)}{bw_i} = \frac{\sigma}{bw_i} * \frac{fr^2 * fs}{dfs}. \tag{8}$$

Here, $S_i(c)$ is the average data size per frame generated and manipulated from i-th video stream. bw_i denotes the bandwidth (Mbits/s) allocated to i-th video stream. σ is the number of bits required to represent the information carried by one pixel (Mb).

The computation latency is the average latency per frame to process the video analytics on the data transferred from a video stream in the analyzer. For the computation latency, we measure how frame resolution affect the video analytics performance on latency in the measurement of III-A. We use GPU processing time per frame as the metric of latency. A GPU tend to be stable and static for processing each

operation of an object detection model and the number of operations to be processed is fixedly determined with the frame resolution and the object detection model. Figure 5(c) shows the latency results on frame resolution over R for a video stream (i.e. The latency results of four video streams are same). In Figure 5(c), the points of a line represent the GPU processing time over R and $fs = dfs$ in a video stream. Fitting the measured latency points of each video stream on frame resolution, the function $P(fr)$ is empirically formulated as a convex polynomial function of three coefficients to represent the latency function with a continuous variable fr of frame resolution. It reflect the fact that the latency increases as the video frame resolution becomes higher.

$$P(fr) = p_1 * fr^2 + p_2 * fr + p_3. \tag{9}$$

Here, $\{p_1, p_2, p_3\}$ are known constant coefficients of a video stream.

We obtain the constant coeffcients of a video streams in Figure 5(c) with about 0 RMSE, as listed in Table 1.

Besides, the frame rate's impact on latency can be easily reflected by multiplying the frame rate and $P(fr)$. The computation latency of i-th video stream with the configuration c of continuous variables on frame resolution fr and rate fs is defined as follows:

$$l_i^P(c) = \frac{fs * P(fr)}{dfs}. \tag{10}$$

Here, dfs means the number of generated frames from i-th video stream in a second and fs means the number of manipulated frames among dfs.

The static latency l^C is the static latency per frame to process the communication link establishment, image preprocessing in a video stream.

Finally, the service latency experienced by i-th video stream with the configuration c of continuous variables on frame resolution fr and rate fs is defined as follows:

$$l_i(c) = l_i^T(c) + l_i^P(c) + l^C. \tag{11}$$

D. SYSTEM OBJECTIVES

In this subsection, we formulate the video analytics system problem to minimize the total service latency and maximize the total analytics accuracy for analyzing video streams generated from cameras under limited resource capacity of a GPU by controlling configurations over the time horizon. We decouple the continuous-time problem to the discrete-time problem with discrete time slots. For convenience, the one time slot length is defined as the segment length T and the j-th time slot is defined as the time at which the j-th segment, which has generated during $(j-1)$-th time slot, have to be processed. Then, in the j-th time slot, the problem is to process the j-th segments of video streams, denoted as $\{S_{i,j} | \forall i \in [I]\}$. The configuration decision vectors to be decided for $\{S_{i,j} | \forall i \in [I]\}$ are defined as $C_j = \{c_{i,j} | \forall i \in [I]\}$. The configuration decision vector $c_{i,j}$ decides the frame rate knob $fs_{i,j}$ to sample $S_{i,j}$ and the frame resolution knob $fr_{i,j}$ to resize the sampled frames of $S_{i,j}$, denoted as $c_{i,j} = \{fs_{i,j}, fr_{i,j}\}$. $fr_{i,j}$ and $fs_{i,j}$ are determined among selectable discrete values of R and S. respectively, as mentioned in section III-A. In addition, the set of configuration candidates for each video stream, which is comprised of all possible combination of the knob values in R and S, is defined as $C_v = \{(fr, fs) | \forall fr \in R, fs \in S\}$. We relax the discrete variables, $c_{i,j} = \{fs_{i,j}, fr_{i,j}\}$ and C_j, into the continuous variables, $\bar{c}_{i,j} = \{\bar{fs}_{i,j}, \bar{fr}_{i,j}\}$ and $\bar{C}_j$, to efficiently solve the video analytics system problem. $\bar{fs}_{i,j}$ is constrained by the minimum and maximum value of S, denoted as $[s_{min} \ s_{max}]$. $\bar{fr}_{i,j}$ is constrained by the minimum and maximum value of R, denoted as $[r_{min} \ r_{max}]$. Based on Equation 7, the analytics accuracy and of i-th video stream with $\bar{c}_{i,j}$ for $S_{i,j}$ is defined as $a_{i,j}(\bar{c}_{i,j}) = a_i(\bar{c}_{i,j})$, since the accuracy of j-th segment is same with the average accuracy per frame in i-th video stream. Based on Equation 11 and Equation 10, the service latency of i-th video stream with $\bar{c}_{i,j}$ for $S_{i,j}$ is defined as $l_{i,j}(\bar{c}_{i,j}) = l_i(\bar{c}_{i,j}) * l_s$ with $bw_{i,j}$ which is the bandwidth allocated to i-th video stream for j-th segment. The computation latency of i-th video stream with $\bar{c}_{i,j}$ for $S_{i,j}$ is the resource allocated to i-th video stream (i.e. the usage time of the analyzer), defined as $l_{i,j}^P(\bar{c}_{i,j}) = l_i^P(\bar{c}_{i,j}) * l_s$.

In order to characterize the tradeoff between the service latency and analytics accuracy, we define the utility of i-th video stream on j-th segment with a positive weight parameter γ which reflects the preference between the latency and the accuracy for analyzing j-th segment of i-th video stream. It is denoted as $f_{i,j}(\bar{c}_{i,j}) = l_{i,j}(\bar{c}_{i,j}) - \gamma a_{i,j}(\bar{c}_{i,j})$. Then, the total utility $\mathcal{F}_j$ is defined as $\mathcal{F}_j(\bar{C}_j) = \frac{1}{I}\sum_{i\in[I]} f_{i,j}(\bar{c}_{i,j})$. Our video analytics system problem $\mathcal{P}_0$ aims to make an configuration decision $\bar{C}_j$ to optimize the utility $\mathcal{F}_j$.

$$\mathcal{P}_0 : \underset{\bar{C}_j}{\text{minimize}} \ \mathcal{F}_j(\bar{C}_j) = \frac{1}{I}\sum_{i\in[I]} f_{i,j}(\bar{c}_{i,j}) \tag{12}$$

$$= \frac{1}{I}\left(\sum_{i\in[I]} l_{i,j}(\bar{c}_{i,j}) - \gamma \sum_{i\in[I]} a_{i,j}(\bar{c}_{i,j})\right) \tag{13}$$

$$\text{subject to } \mathcal{C}_1 : l_{i,j}(\bar{c}_{i,j}) \le l_i^{max}, \quad i \in I; \tag{14}$$

$$\mathcal{C}_2 : \sum_{i\in[I]} l_{i,j}^P(\bar{c}_{i,j}) \le T; \tag{15}$$

$$\mathcal{C}_3 : s_{min} \le \bar{fs}_{i,j} \le s_{max}, \quad i \in I; \tag{16}$$

$$\mathcal{C}_4 : r_{min} \le \bar{fr}_{i,j} \le r_{max}, \quad i \in I; \tag{17}$$

where l_i^{max} is the maximum tolerable latency of the i-th video stream, and T is the one time slot length, in which the total processing time should be finished on the edge server. The constraints C1 means that the service latency of users does not exceed their maximum tolerable latency; the constraint C2 means that the allocated computation resources, the usage times of video streams on the analyzer, do not exceed the total computation resource, a time slot, on the edge server, as mentioned in section III-A. The constraints C3 and C4 are the constraints of the frame rate and resolution.

We utilize the Lagrange duality method to solve the above problem. We define the dual variables $\lambda = \{\lambda_i | \forall i \in [I]\}$ and

μ associated with the constraints in (14)–(15), respectively. We also define the dual variables $\zeta = \{\zeta_i | \forall i \in [I]\}$, $\eta = \{\eta_i | \forall i \in [I]\}$, $n = \{n_i | \forall i \in [I]\}$ and $m = \{m_i | \forall i \in [I]\}$ associated with the constraints in (16)–(17), respectively. The Lagrange function of $\mathcal{P}_1$ is follows:

$$\begin{aligned}\mathcal{P}_1 : \underset{\bar{C}_j}{\text{minimize}}\ & \mathcal{L}(\bar{C}_j, \lambda, \mu, \zeta, \eta, n, m) \\ = & F_j(\bar{C}_j) - \sum_{i\in[I]} \lambda_i * (l_{i,j}(\bar{c}_{i,j}) - l_i^{max}) \\ & -\mu(\sum_{i\in[I]} l_i^P(\bar{c}_{i,j}) - T) \\ & - \sum_{i\in[I]} \zeta_i(s_{min} - \bar{fs}_{i,j}) - \sum_{i\in[I]} \eta_i(\bar{fs}_{i,j} - s_{max}) \\ & - \sum_{i\in[I]} n_i(r_{min} - \bar{fr}_{i,j}) - \sum_{i\in[I]} m_i(\bar{fr}_{i,j} - r_{max}), \quad (18)\end{aligned}$$

where λ_i, ζ_i, η_i, n_i, m_i and μ are non-negative Lagrange multipliers. Through $\mathcal{P}_2$, the dual problem $\mathcal{P}_3$ is defined as follows:

$$\begin{aligned}\mathcal{P}_2 : \ & \psi(\lambda, \mu, \zeta, \eta, n, m) \\ & = \underset{\bar{C}_j}{\text{minimize}}\ \mathcal{L}(\bar{C}_j, \lambda, \mu, \zeta, \eta, n, m) \\ \mathcal{P}_3 : \ & \underset{\lambda,\mu,\zeta,\eta,n,m}{\text{maximize}}\ \psi(\lambda, \mu, \zeta, \eta, n, m) \quad (19)\end{aligned}$$

It can be readily proved that $\mathcal{P}_0$ is convex and satisfies Slater's condition. Thus, strong duality holds between the primal and dual problems, which means solving $\mathcal{P}_0$ is equivalent to solving the dual problem $\mathcal{P}_3$.

$$\begin{aligned}& \psi(\lambda^*, \mu^*, \zeta^*, \eta^*, n^*, m^*) \\ & = \mathcal{L}(\bar{C}_j^*, \lambda^*, \mu^*, \zeta^*, \eta^*, n^*, m^*) \\ & = \mathcal{F}(\bar{C}_j^*) \quad (20)\end{aligned}$$

Here, $\bar{C}_j^*$ and $\lambda^*, \mu^*, \zeta^*, \eta^*, n^*, m^*$ are the primal optimal solution of $\mathcal{P}_0$ and the dual optimal solution of $\mathcal{P}_3$, which satisfy the Karush-Kuhn-Tucker(KKT) condition. To obtain $\bar{C}_j^*$ and $\lambda^*, \mu^*, \zeta^*, \eta^*, n^*, m^*$, we resolve $\mathcal{P}_2$ and $\mathcal{P}_3$ based on the inexact block coordinate descent (BCD) method (a variant of BCD method), described in section IV-A in detail. Finally, the solution of C_j is decided to the configuration candidate close to $\bar{C}_j^*$ among C_v.

E. LIGHTWEIGHT ONLINE PROFILING BASED CONFIGURATION ADAPTATION ALGORITHM IN A GPU-ENABLED EDGE SERVER

In practice, the configuration's impact on accuracy is significantly dynamic, as mentioned in II-B. To find the best configuration which varies over time, the accuracy model on configurations have to be profiled periodically in online. The online profiling, which profiles possible configurations periodically by the direct measurement, requires a high profiling cost. Furthermore, practically, the accuracy model cannot be expressed completely in analytic form like Equation 7 in III-B, with the lack of analytic understanding on theoretical properties of the object detection models. The accuracy model defined in III-B is not applicable in some cases which violate a concave function and causes a significant error. In the extension of the experiment in II-B, we measure the accuracies on the different values of each knob over the whole duration of a video stream as shown in Figure 6. We use the frame resolution of 320p and 608p and the frame rate of 1fps, 30fps, respectively. The results show that the accuracy on each value is so fluctuated, as expected. Especially, the best configuration is not consistent. For example, in Figure 6(a), in the early part of a video stream, 608p is the best configuration to guarantee a desired accuracy threshold, but in the latter part of a video stream, the best configuration is changed to 320p. In Figure 6(b), in the early part of a video stream, the best configuration is 1fps enough to guarantee a desired accuracy threshold, but in the latter part of a video stream, the best configuration is changed to 30fps.

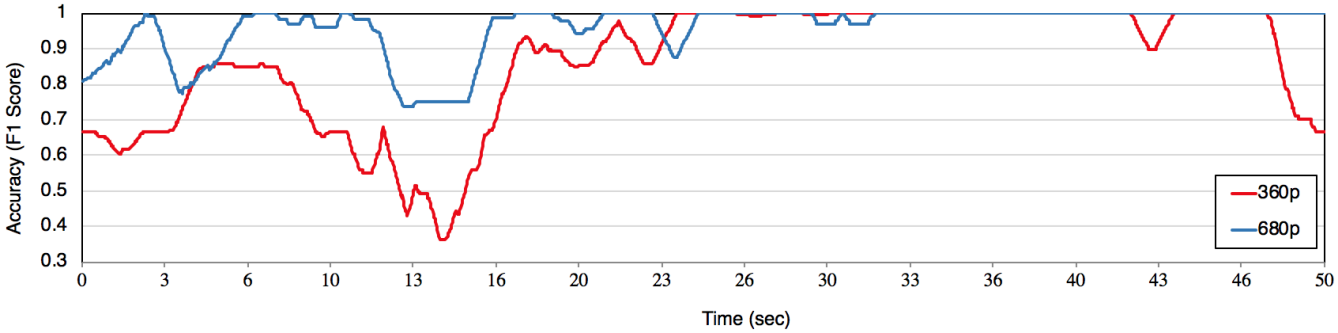
Since the accuracy model $a_{i,j}(c)$ only can be formulated as a discrete function with a configuration c of C_v in this case, we reformulate the video analytics system problem to find the configuration decision vector $C_j = \{c_{1,j}, c_{2,j}, \ldots, c_{i,j}, \ldots c_{I,j}\}$ optimizing the utility $\mathcal{F}_j$ for j-th segment, as follows:

$$\begin{aligned}\mathcal{P}_4 : \underset{C_j}{\text{minimize}}\ & \mathcal{F}_j(C_j) = \frac{1}{I}\sum_{i\in[I]} f_{i,j}(c_{i,j}) \quad (21) \\ & = \frac{1}{I}\Big(\sum_{i\in[I]} l_{i,j}(c_{i,j}) - \gamma \sum_{i\in[I]} a_{i,j}(c_{i,j})\Big) \quad (22) \\ \text{subject to } & \mathcal{C}_1, \mathcal{C}_2, \quad (23) \\ & \mathcal{C}_3 : c_{i,j} \in C_v, \forall i \in [I]; \quad (24) \\ \text{where } & C_v = \{fr, fs | \forall fr \in R, fs \in S\}; \quad (25)\end{aligned}$$

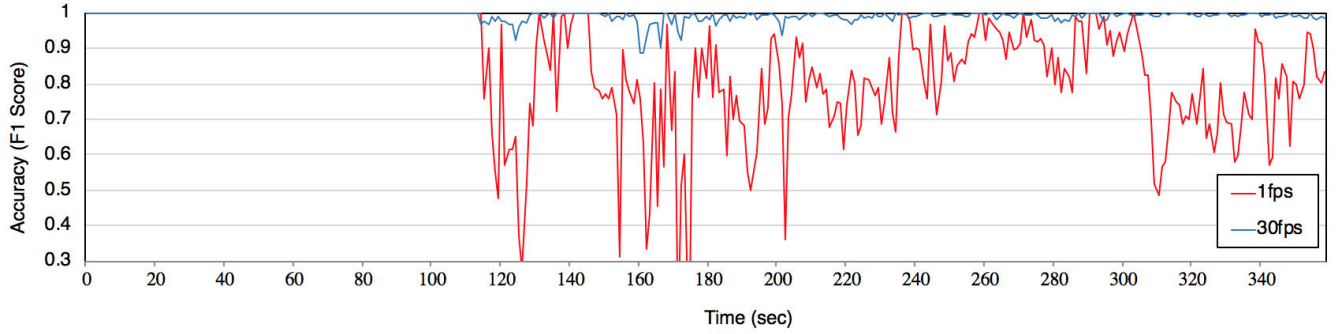
Here, C_v is the set of configuration candidates for each video stream which is comprised of all possible combination of the knob values in R and S. The constraints $\mathcal{C}_3$ mean that each configuration is determined among the items of C_v.

The video analytics system problem is a multiple-choice knapsack problem (MCKP), where the items of each class are the configuration candidates of each video stream. The item's profit and weight in each class are the utility and processing time of its configuration in each video stream. In a form of MCKP, the problem is to decide exactly one configuration (item) of each video stream (class) in order to maximize the utility sum (profit sum) without exceeding the processing capability (capacity) in the processing time sum (weight sum). However the problem is proven to be NP-complete, so no polynomial time algorithm exists for it. A brute-force solution to the video analytics system problem would take a running time of $O\big(I \cdot |S| \cdot |R|\big)$. Obviously, it is impractical to search an optimal configuration decision in fine granularity for j-th segment and proceed the new decisions periodically over the subsequent segments.

In order to resolve these problems, we propose the lightweight online profiling method to obtain the accuracy model $a_{i,j}$ for $\mathcal{P}_4$. The proposed method overcomes the high profiling cost problem of the existing online profiling systems



(a) Frame Resolution



(b) Frame Rate

FIGURE 6. Dynamics of video analytics accuracy in a video according to frame resolution and rate.

such as Chameleon [10] and AWStream [13], as mentioned in II-B. Finally, we propose the new configuration adaptation algorithm based on the lightweight online profiling method to find the best configuration of each video stream and minimize accuracy degradation of each video stream in the constrained computing capacity of a GPU-enabled edge computing. Figure 7 shows the entire structure and procedure of our video analytics system with the components based on the proposed methods, described in subsection III-E1 and III-E2: Lightweight Online Profiler and Lightweight Online Profiling based Configuration Controller.

1) LIGHTWEIGHT ONLINE PROFILING METHOD

The existing video analytics systems based on online profiling, such as Chameleon [10] and AWStream [13], execute a profiling process and an analyzing process in parallel separately with additional computing resources, which cause high profiling cost. Avoiding this high profiling cost, we allocate the short-term profiling process to be executed concurrently in the analyzing process. Since the short-term profiling process is not enough to measure the accuracy model on configurations like previous works, we try to estimate it with some error. Before starting to analyze j-th segment, we profile the accuracy model on configurations, $a_{i,j}(c), \forall c \in C_v$, to determine a configuration $c_{i,j}$ for j-th segment in i-th video stream. In this procedure, we consider $(l_p + 1)$ frames as a short-term profiling period consisting of the last l_p frames of $j-1$-th segment and the first frame of j-th segment, denoted as $\{f_{i,j-1}^{(l_s-l_p+1)}, \ldots, f_{i,j-1}^{(l_s-1)}, f_{i,j-1}^{l_s}, f_{i,j}^{1}\}$. To simplify the notations, hereinafter we will omit the mention of the index i, j, as $pf = \{f_{(l_s-l_p+1)}, \ldots, f_{(l_s-1)}, f_{l_s}, f_1\}$. We profile the short-term profiling period in the most expensive configuration, called golden configuration and utilize its results to estimate the accuracy model for j-th segment.

The first step is to find all of objects that are expected to exist in j-th segment and extract the velocity and size characteristics of them in the short-term profiling period. We assume that the objects and their characteristics are maintained over j-th segment, based on the temporal correlations [10]. Firstly, we profile the profiling period in the golden configuration. As a result, all frames in the short-term profiling period are sampled and the bounding boxes of objects are detected in each frame of $pf = \{f_{(l_s-l_p+1)}, \ldots, f_{(l_s-1)}, f_{l_s}, f_1\}$ with the highest resolution. Let $\{B'^{dfr}_{(-l_p+1)}, \ldots, B'^{dfr}_{-1}, B'^{dfr}_{0}, B'^{dfr}_{1}\}$ denote the bounding boxes of objects detected in pf, respectively. They are finalized as the detection result $\{B_{(-l_p+1)}, \ldots, B_{-1}, B_0, B_1\}$, without the additional manipulation, $B_q = B'^{dfr}_{q}, q = (-l_p + 1), \ldots, -1, 0, 1$. Secondly, we extract K tracks of objects on f_1 over the short-term profiling period, denoted as $\{tr_k | \forall k \in [K]\}$, using a simple online and real-time tracking method on $\{B_{(-l_p+1)}, \ldots, B_{-1}, B_0, B_1\}$ [19]. Each track represents the linear velocity model (i.e. the representation and motion model used to propagate a target's identity into the next frame), solved optimally via a Kalman filter framework [9], [19]. The state of object k is modelled as:

$$tr_k = [x_k, y_k, l_k, h_k, v_{x_k}, v_{y_k}, v_{l_k}, v_{h_k}], \tag{26}$$

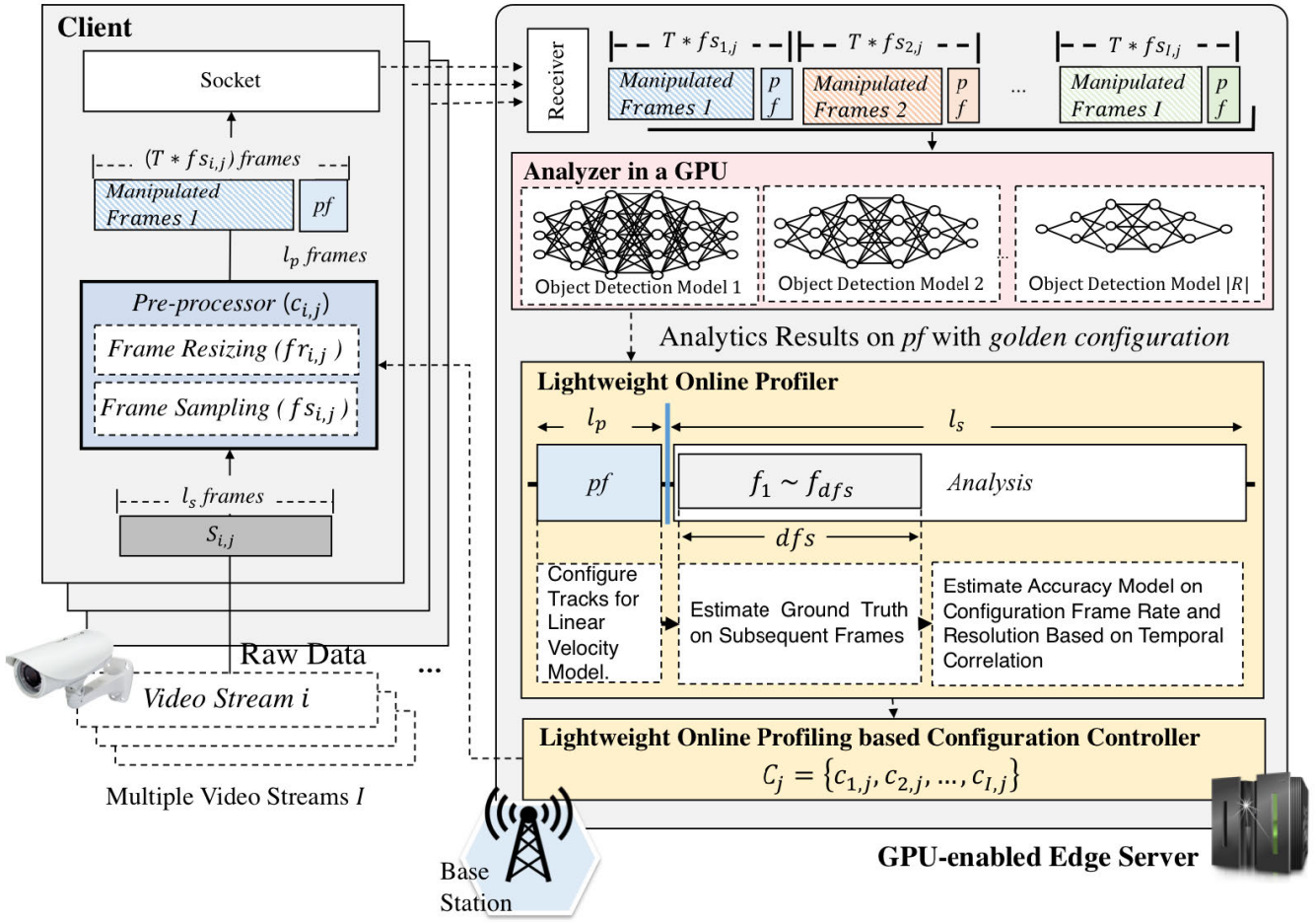


FIGURE 7. The proposed video analytics system with lightweight online profiling-based configuration adaptation algorithm in a GPU-enabled edge server.

where position, size, and velocity are represented by (x, y), (l, h) and (v_x, v_y, v_l, v_h).

The second step is to estimate the ground truth boxes of frames on the first sub-segment, $\{f_1, f_2, \ldots, f_q, \ldots, f_{dfs}\}$, in j-th segment, denoted as $\{\hat{G}_1, \hat{G}_2, \ldots, \hat{G}_q, \ldots, \hat{G}_{dfs}\}$, leveraging the tracks $\{tr_1, tr_2, \ldots, tr_K\}$ extracted in the first step. Firstly, we extract the bounding boxes and velocities of objects in f_1 from $\{tr_1, tr_2, \ldots, tr_K\}$. Let $TB_1 = \{tb_1^1, tb_1^2, \ldots, tb_1^K\}$ and $V_1 = (v_1^1, v_1^2, \ldots, v_1^K)$ denote the bounding boxes and velocities of objects in f_1. Let tb_1^k and v_1^k denote the bounding box and velocity of k-th track in f_1. Since derived from the detection results of the golden configuration on the short-term profiling period, the tracks can provide the approximated ground truth boxes and velocities for the first frame f_1 of j-th segment.

$$\begin{aligned} TB_1 &= (tb_1^1, tb_1^2, \ldots, tb_1^K) \\ &= (\{x_1, y_1, l_1, h_1\}, \{x_2, y_2, l_2, h_2\}, \ldots, \{x_K, y_K, l_K, h_K\}) \\ &= \hat{G}_1 \approx G_1 \\ V_1 &= (v_1^1, v_1^2, \ldots, v_1^K) \\ &= (\{v_{x_1}, v_{y_1}, v_{l_1}, v_{h_1}\}, \{v_{x_2}, v_{y_2}, v_{l_2}, v_{h_2}\}, \\ &\quad \ldots, \{v_{x_K}, v_{y_K}, v_{l_K}, v_{h_K}\}) \end{aligned} \tag{27}$$

Here, $\hat{G}_1$ denote the predicted ground truth boxes of f_1. The bounding boxes TB_1 can approximate the ground truth boxes G_1 so that $TB_1 = \hat{G}_1$. Secondly, we predict the ground truth boxes of the remained frames $\{f_2, \ldots, f_q, \ldots, f_{dfs}\}$ by shifting TB_1 based on their velocity of V_1. The tracks can be also utilized to estimate how the approximated ground truth boxes of objects in f_1 change with respect to location and size over j-th segment, leveraging V_1. In such a short-term time period like a segment, the simple linear velocity models of the tracks can be effectively used with a reasonable error, based on temporal correlation. Let $TB_{1\to q}$ denote the shifted boxes of TB_1 to f_q. Let $tb_{1\to q}^k$ denote the shifted bounding box of tb_1^k to f_q.

$$\begin{aligned} TB_{1\to q} &= (tb_{1\to q}^1, tb_{1\to q}^2, \ldots, tb_{1\to q}^K) \\ &= \hat{G}_q = (\hat{g}_q^1, \hat{g}_q^2, \ldots, \hat{g}_q^K) \\ &\approx G_q \end{aligned} \tag{28}$$

Here, $\hat{G}_q$ denote the predicted ground truth boxes of f_q and $\hat{g}_q^k$ denote the predicted ground truth box of k-th track for f_q. In the same way with (27), the bounding boxes TB_q can approximate the ground truth boxes G_q so that $TB_q = \hat{G}_q$. It is performed by shifting the coordinates of tb_1^k based on v_q^k and

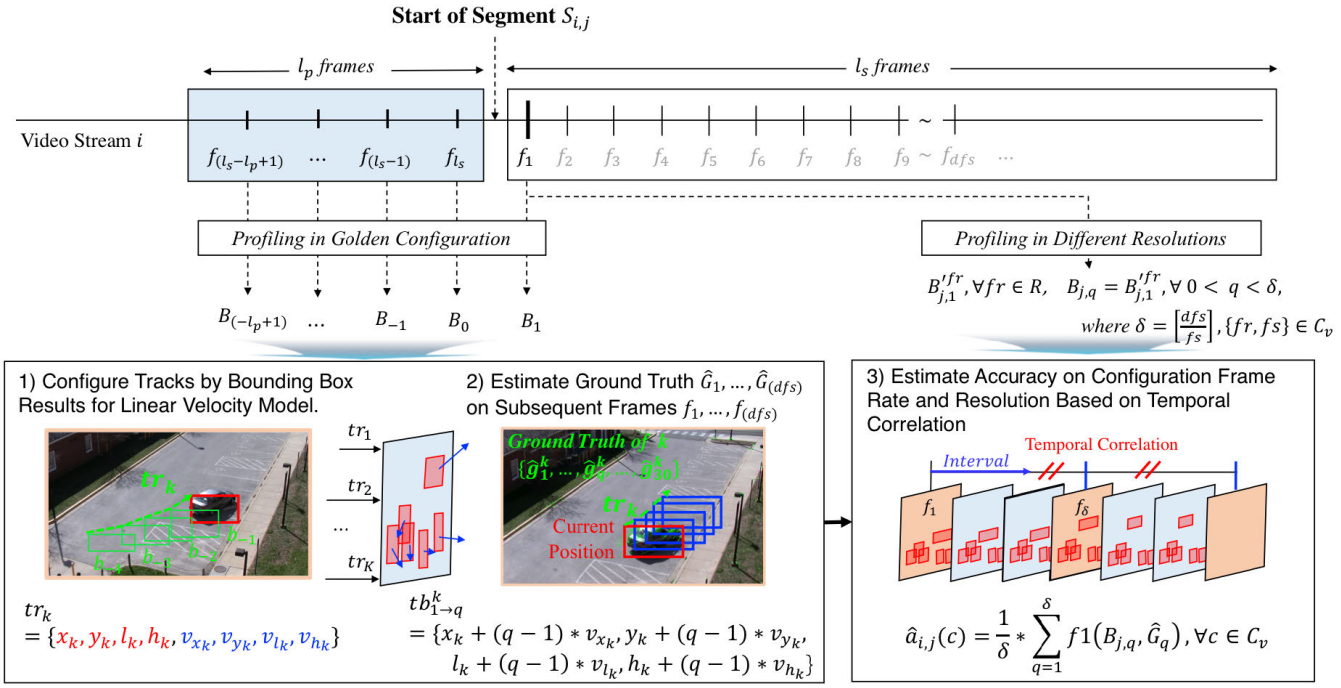


FIGURE 8. Procedure of the proposed lightweight online profiling method.

predicting those of $\hat{g}^k_q$, as defined as follows:

$$\hat{g}^k_q = tb^k_{1\to q} = \big(x_k + (q-1) * v_{x_k}, y_k + (q-1) * v_{y_k}, \\ l_k + (q-1) * v_{l_k},\ h_k + (q-1) * v_{h_k}\big), \\ \forall k \in [K]. \tag{29}$$

The third step is to estimate the accuracy model on configurations, $a_{i,j}(c), \forall c \in C_v$, for j-th segment. Using the predicted ground truth boxes of the first sub-segment, $\{\hat{G}_1, \ldots, \hat{G}_q, \ldots, \hat{G}_{dfs}\}$, and the detected bounding boxes of the first frame for all available frame resolutions, $\{B'^{fr}_{j,1} | fr \in R\}$, we can measure the accuracy on a configuration c of C_v for the small part of j-th segment based on Equation (30). The small part of j-th segment is defined as the 1-th interval of j-th segment and its finalized bounding boxes are defined based on Equation 1, as follows:

$$B_{j,q} = B'^{fr}_{j,1}, \quad \forall\, 0 < q \le \delta, \text{where } \delta = \left\lceil \frac{dfs}{fs} \right\rceil, \{fr, fs\} \in C_v. \tag{30}$$

The measured accuracies are used to estimate the accuracy model $a_{i,j}$, denoted as follows:

$$\hat{a}_{i,j}(c) = \frac{1}{\delta} * \sum_{q=1}^{\delta} f1(B_{j,q}, \hat{G}_q), \quad \forall c \in C_v \tag{31}$$

We assume that $\hat{a}_{i,j}$ can approximate the accuracy model $a_{i,j}$ based on temporal correlation. Accuracy characteristics tend to be relatively stable over time in short-term time period like a segment.

$$\hat{a}_{i,j}(c) \approx a_{i,j}(c) = \frac{1}{l_s} * \sum_{q=1}^{l_s} f1(B_{j,q}, G_{j,q}), \quad \forall\, c \in C_v. \tag{32}$$

Avoiding that the short-term profiling period fails to reflect the whole context of a segment fully and cause some error, the weighted moving average method is applied to the estimated accuracy models of several consecutive segments. The estimated accuracy of j-th segment is finalized as follows:

$$\hat{a}^*_{i,j}(c) = w_j * \hat{a}_{i,j}(c) + w_{j-1} * \hat{a}_{i,j-1}(c) \\ + \ldots + w_{j-l_m} * \hat{a}_{i,j-l_m}(c), \forall\, c \in C_v. \tag{33}$$

Here, l_m is the weighted moving average length and w_x is the weight of $\hat{a}_{i,x}$. Finally, we can determine the best configuration $c^*_{i,j}$ of j-th segment for i-th video stream, defined as follows:

Definition 1 (Best Configuration): the one with the lowest latency of which accuracy is over a desired threshold η_a.

$$c^*_{i,j} = \text{argmin}(l^P_{i,j}(c_{i,j}) | \hat{a}^*_{i,j}(c_{i,j}) > \eta_a), \quad c_{i,j} \in C_v \tag{34}$$

Figure 8 shows the entire procedure of the proposed lightweight online profiling method.

2) LIGHTWEIGHT ONLINE PROFILING BASED CONFIGURATION ADAPTATION ALGORITHM

Based on the estimated accuracy model $\hat{a}^*_{i,j}(c_{i,j})$, we propose an algorithm to resolve the problem $\mathcal{P}_4$. It determines the configurations of video streams for j-th segment,

$C_j = \{c_{1,j}, c_{2,j}, \ldots, c_{i,j}, \ldots c_{I,j}\}$, in order to maximize the utility for analyzing the j-th segments of video streams under limited resource capacity. C_j is adapted by reducing the total latency in a greedy way while minimizing the total utility degradation.

Each video stream has a set of configuration candidates which can be chosen, C_v. The configuration candidates C_v of i-th video stream is filtered based on Definition 2 and the constraint $\mathcal{C}_1$ of (23) with the given $bw_{i,j}$.

Definition 2: If two configurations j and k for i-th video stream satisfy

$$l^P_{i,j}(j) \leq l^P_{i,j}(k) \text{ and } f_{i,j}(j) \leq f_{i,j}(k), \quad j,k \in C_v, \tag{35}$$

then we say that configuration j is dominated by configuration k.

The set of the filtered configuration candidates for i-th video stream is sorted in decreasing order on the estimated accuracy of each item and is denoted as N_i.

For $\forall i$, $c_{i,j}$ is initially set with the first item of N_i (i.e. a configuration for the highest accuracy). The initial configurations of video streams are denoted as $C^0_j = \{c^0_{1,j}, c^0_{2,j}, \ldots, c^0_{i,j}, \ldots, c^0_{I,j}\}$. However, if $\sum_{i\in[I]} l^P_i(c^0_{i,j})$ exceeds T, we have to additionally adapt and decide the new configuration for video streams among $\{N_1, N_2, \ldots, N_i, \ldots, N_I\}$ to finish the total latency of C_j within T and keep the input queue stable. That is, in order to remove the total exceeded time $ET = \sum_{i\in[I]} l^P_i(c^0_{i,j}) - T$, the configuration C^0_j is adapted to be cheaper.

The adaptation is conducted for C^0_j iteratively in a greedy way in order to maximize the total latency reduction while minimizing the total utility degradation until $ET \leq 0$. In each iteration, one of configurations is adapted to be cheaper. If an item of N_i is chosen for adapting $c_{i,j}$ in an iteration, the item is removed from N_i. Let $c^d_{i,j}$ denote the configuration for j-th segment of i-th video stream, updated via d iterations. Each adaptation derives the total latency reduction from reducing the data size to be processed, although causing the accuracy degradation. Then, in $(d-1)$-iteration for i-th video stream, the efficiency of the next adaptation candidate $\hat{c}^d_{i,j}$ to realize the latency reduction and minimize the utility degradation is measured with the estimated latency $l^P_{i,j}(\hat{c}^d_{i,j})$ and accuracy $\hat{a}^*_{i,j}(\hat{c}^d_{i,j})$, defined as follows:

$$e^d_{i,j} = \frac{l^P_{i,j}(c^{d-1}_{i,j}) - l^P_{i,j}(\hat{c}^d_{i,j})}{f_{i,j}(c^{d-1}_{i,j}) - f_{i,j}(\hat{c}^d_{i,j})}, \tag{36}$$

where

$$\hat{c}^d_{i,j} = \text{argmax}(\hat{a}^*_{i,j}(x)), \forall x \in N_i \tag{37}$$

Here, in $(d-1)$-th iteration, the available adaptation candidates and their efficiencies are denoted as $\hat{c}^d_{i,j}, \forall i$ and $e^d_{i,j}, \forall i$ for d-th iteration.

The highest efficiency adaptation candidate is determined among $e^d_{i,j}, \forall i$ for updating $c^d_{i,j}$.

$$c^d_{i,j} = \hat{c}^d_{i,j} \tag{38}$$

where

$$i = \text{argmax}(e^d_{x,j}), \quad \forall x \in [I] \tag{39}$$

In the same way, updating the configurations are proceeded until d^*-th iteration satisfying $\sum_{i\in[I]} l^P_i(c^{d^*}_{i,j}) \leq T$. As a result, for $\forall i$, $c^{d^*}_{i,j}$ is decided as the best configuration of i-th video stream for j-th segment, $c_{i,j} = c^{d^*}_{i,j}$.

Algorithm 1: Lightweight Online Profiling based Configuration Adaptation Algorithm in a GPU-enabled Edge Computing

Input: For $\forall i$, bandwidth $bw_{i,j}$, estimated accuracy model $\hat{a}^*_{i,j}$.
Result: Decided configurations $C_j = \{c_{i,j} | i \in [I]\}$.
For $\forall i$, find N_i by filtering C_v based on Definition 2 and the constraint $\mathcal{C}_1$ of (23) with the given $bw_{i,j}$;
For $\forall i$, sort N_i in decreasing order on accuracy;
For $\forall i$, pull the first item of N_i and set $c^0_{i,j}$ with it;
Calculate the total exceeded time
$ET \Leftarrow \sum_{i\in[I]} l^P_i(c^0_{i,j}) - T$;
while $ET > 0$ **do**
  For $\forall i$, extract the configuration candidate of d-th iteration from N_i, $\hat{c}^d_{i,j}$;
  $\hat{c}^d_{i,j} = \text{argmax}(a_{i,j}(x)), \forall x \in N_i$
  For $\forall i$, calculate the efficiency of the configuration candidate, $e^d_{i,j}$;
  The highest efficiency adaptation candidate is determined and updated as follows: $c^d_{i,j} \Leftarrow \hat{c}^d_{i^m,j}$,
  where $i^m = \text{argmax}(e^d_{x,j}), \forall x \in [I]$;
  Pull the decided item of N_{i^m};
  $ET \Leftarrow \sum_{i\in[I]} l^P_i(c^d_{i,j}) - T$;
  $d^* \Leftarrow d$;
end
For $\forall i$, $c_{i,j} \Leftarrow c^{d^*}_{i,j}$;

IV. EVALUATION

In our experiment, we deployed the video analytics system on the physical machine equipped with CPU, Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, GPU, GeForce GTX 1080(8GB), memory 16GB in order to process all analysis tasks on video streams. The analysis task is an DNN based object detection. We use CUDA 9.0 and cuDNN 7.0 [26] to accelerate the DNN inference speed based on GPU. We use Keras 2.2.4 [27] APIs with Tensorflow framework 1.12 [28] to execute an DNN model for object detection. We use a pre-trained object-detection model, Yolo [5],[3] and use three types of Yolo, Yolov3-320, 416, 608, of which input size are 320×320, 416×416 and 608×608 respectively. Its non maximum suppression IoU and score threshold are 0.45 and 0.3. All models are pre-trained on a COCO image dataset and can detect 80 object classes [30].

[3]Our algorithm can also be adapted to other popular DNN models such as faster R-CNN

To evaluate the performance of the proposed algorithms in our video analytics system, we use a subset of surveillance videos and annotations from VIRAT 2.0 Ground dataset [29], captured by real-world static cameras from various scenes of roads and parking lots. Its objects are small compared to frame size, tightly packed together with occlusions and viewed from a distance. We construct four video streams for our experiments using the fully-annotated 20 videos of VIRAT 2.0 Ground dataset, captured by four video cameras deployed in different area, of which frame resolution and rate are 30 fps and 1920 × 1080p (for 3 video streams) or 1280 × 720p (for 1 video stream). Each video stream is composed of 10800 frames and its objects mainly consist of car and person.

With this experimental environment, the default frame rate and resolution of the video streams, *dfr* and *dfs*, are set as 30 and 608 which are the original frame rate of VIRAT 2.0 Ground dataset and the highest input size of available object-detection models, respectively. Then, the set of selectable values on frame rate knob and resolution knob, R and S, are set as $\{608, 416, 320\}p$ and $\{30, 15, 10, 6, 5, 3, 2, 1\}fps$. Table 2 presents the associated parameters of these experiments.

TABLE 2. Experiment Setting of Our Video Analytics System.

Notation (Unit)	Values
Number of video streams (#)	4
Frames of a video stream (frames)	10800
Object detection models	Yolo, Yolov3-320, 416, 608
Object detection non maximum suppression IoU and score threshold	0.45 and 0.3
Frame resolution values R (p)	320, 416, 608
Frame rate values S (fps)	30, 15, 10, 6, 5, 3, 2, 1
Default frame rate dfs (fps)	30
Default frame resolution dfr (p)	608

A. EXPERIMENTAL RESULTS AND ANALYSIS OF VIDEO ANALYTICS OPTIMIZATION ALGORITHM IN A GPU-ENABLED EDGE COMPUTING

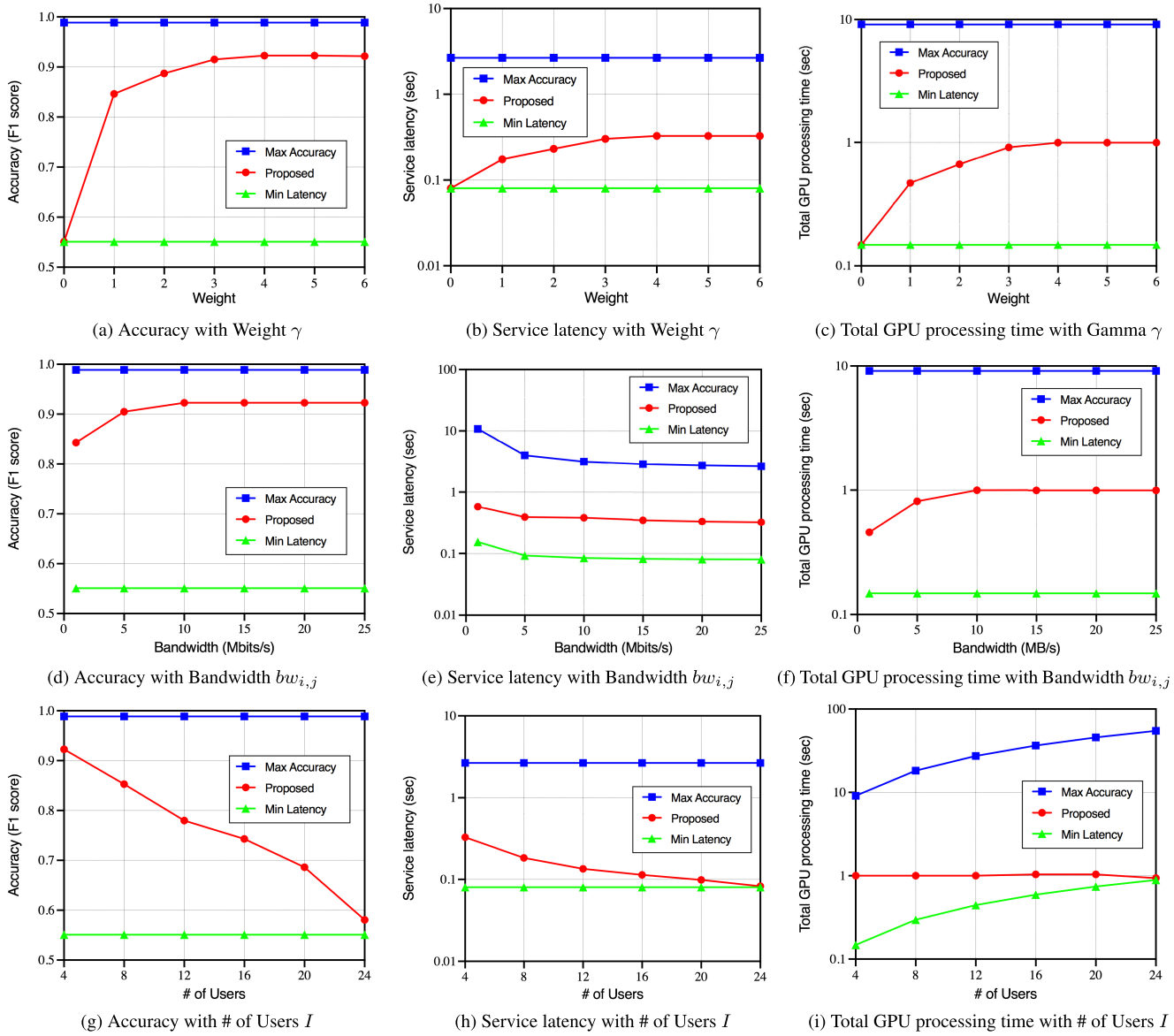
In this experiment, we evaluated the performance of the proposed video analytics optimization algorithm to resolve the problem $\mathcal{P}_0$. We assume that $bw_{i,j}$ is known by prediction. For convenience, T is set to 1, but it can be longer as long as bandwidth is predictable in each time slot. Since a segment of each video stream is basically given a time slot, which is the time until the next segment is created, l_i^{max} is also set to 1. l^C is set to 0.04.

We compared the performance of the proposed algorithm with two baseline policies: *Max Accuracy* policy, which uses the most expensive configuration to maximize the video analytics accuracy of video streams, and *Min Latency* policy, which uses the most cheap configuration to minimize the service latency experienced by video streams.

Figure 9 shows the results of video analytics performance for *Max Accuracy*, *Min Latency* and the proposed algorithm. Figure 9(a)-(c) shows the analytics accuracy, service latency and total GPU processing time of the proposed algorithm under different weight γ. The values of γ are set as [0, 1, 2, 3, 4, 5, 6]. In this experiment, the bandwidth $bw_{i,j}$ and the number of video streams I are set to 25(Mbits/s) and 4, respectively. As γ increases, the proposed algorithm prefers the accuracy performance more than the latency performance. In other words, γ trades the latency for the accuracy. As a result, as shown in Figure 9(a)-(b), the accuracy of the proposed algorithm increases over $\gamma = 0, 1, 2, 3$ while increasing the latency. The proposed algorithm optimizes the configurations and provides the optimized resource-accuracy tradeoffs over the values of γ. On the other hand, the accuracy of the proposed algorithm does not increase anymore over $\gamma = 4, 5, 6$. The reason is that the total GPU processing time of video streams is limited by $T = 1$, as shown in Figure 9(c). Obviously, *Max Accuracy* and *Min Latency* cannot optimize the resource-accuracy tradeoffs, regardless of β. Especially, *Min Latency* violates a certain desired accuracy and *Max Accuracy* violates the maximum tolerable latency $l_i^{max} = 1$ and the one time slot length $T = 1$, as shown in Figure 9(a)-(c). These results also show that the proposed algorithm reduce the service latency about 2.359s compared to *Max Accuracy* at $\gamma = 3$ while the accuracy only drops about 7% and enhance the accuracy about 30% compared to *Min Latency* at $\gamma = 1$ while the service latency only increases about 0.0947s.

Figure 9(d)-(f) shows the analytics accuracy, service latency and total GPU processing time of the proposed algorithm under different bandwidth $bw_{i,j}$. The values of bandwidth $bw_{i,j}$ are set as [0.5, 1, 5, 10, 15, 20, 25] (Mbits/s), according to the experimental measurements in [21]. In this experiment, the weight γ and the number of video streams are set to 4 and 4, respectively. As $bw_{i,j}$ decreases, the proposed algorithm adapts the configuration cheaper and trades the accuracy for the service latency in order to maintain the service latency less than $l_i^{max} = 1$. As shown in Figure 9(d)-(f), over $bw_{i,j} = 5, 1, 0.5$, the proposed algorithm decreases the accuracy and the service latency and the total GPU processing time is not fully utilized. Since a higher bandwidth, $bw_{i,j} = 10, 15, 20, 25$, can support the more expensive configuration, the total GPU processing time is fully utilized to enhance the accuracy. In this case, *Max Accuracy* and *Min Latency* are sensitively affected by insufficient bandwidth and increase the service latency while the proposed algorithm try to relax it.

Figure 9(g)-(i) shows the analytics accuracy, service latency and total GPU processing time of the proposed algorithm under different number of video streams I. The values of I are set as [4, 8, 12, 16, 20, 24]. We duplicate the four video streams to realize $I = 8, 12, 16, 20, 24$. In this experiment, the weight γ and the bandwidth $bw_{i,j}$ are set to 4 and 25 (Mbits/s), respectively. As I increases, the allocated usage time of GPU is insufficient to each video stream so, the proposed algorithm adapts the configuration cheaper to reduce the processing latency. As shown in Figure 9(g)-(i),



(a) Accuracy with Weight γ (b) Service latency with Weight γ (c) Total GPU processing time with Gamma γ

(d) Accuracy with Bandwidth $bw_{i,j}$ (e) Service latency with Bandwidth $bw_{i,j}$ (f) Total GPU processing time with Bandwidth $bw_{i,j}$

(g) Accuracy with # of Users I (h) Service latency with # of Users I (i) Total GPU processing time with # of Users I

FIGURE 9. Comparision results of the proposed video analytics system in terms of accuracy (%), service latency (sec) and total GPU processing time (sec) with weight γ, bandwidth $bw_{i,j}$ and number of users I.

the proposed algorithm decreases the accuracy and the service latency continuously and the total GPU processing time is fully utilized over $I = 4, 8, 12, 16, 20, 24$. In this case, *Max Accuracy* and *Min Latency* are sensitively affected by many video streams with high workload and increase the total GPU processing time while the proposed algorithm keep it within the one time-slot length T.

B. EXPERIMENTAL RESULTS AND ANALYSIS OF LIGHTWEIGHT ONLINE PROFILING BASED CONFIGURATION ADAPTATION ALGORITHM IN A GPU-ENABLED EDGE COMPUTING

In this experiment, we evaluated the performance of the lightweight online profiling based configuration adaptation algorithm to resolve the problem $\mathcal{P}_9$. The proposed algorithm is compared with the aforementioned existing well-known algorithms of Chameleon and VideoStorm, which are based on on-line profiling and off-line profiling respectively. For these existing algorithms, we use the original parameters and configurations described in their paper. For the proposed algorithm, its segment interval T, the number of frames in its profiling period, $l_p + 1$, and its weighted moving average length l_m are set as 4(*sec*), 5(#) and 4. For Chameleon, its segment interval T, the number of segments in its profiling window, w,and its profiling interval t are 4(*sec*), 4(#) and 1(*sec*). For VideoStorm, its profiling interval x is 10.

Before doing this, we evaluated the prediction performance of lightweight online profiling based accuracy model first. In this experiment, the lightweight online profiling based

accuracy model predicts the accuracy of possible configurations on each segment and filters the configurations expected to be above a desired threshold η_a. Then, we check whether the real accuracy of each filtered configuration is more than a desired threshold or not and calculate its F1 score. In this way, we measure how accurately the model predicts. Actually, rather than predicting the exact value of accuracy on a configuration, it predicts the approximate value enough to distinguish the relative accuracy difference between configurations. We conduct the experiment based on the four video streams from VIRAT Dataset 2.0. Figure 10 shows the results of lightweight online profiling based accuracy model on prediction performance for three different desired accuracy threshold (i.e. 0.7, 0.8, 0.9) under the four video streams. It achieves more than about 0.8 on F1 score entirely. Then, the model is applicable to predict the accuracy of possible configurations in on-line.

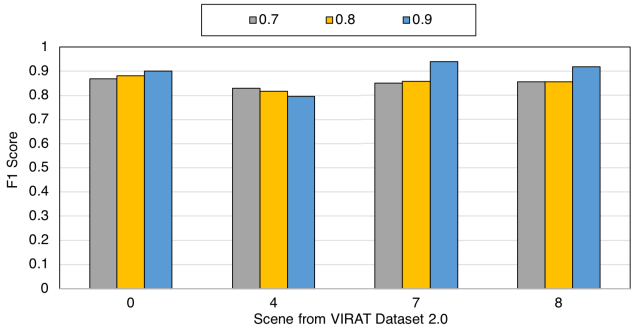


FIGURE 10. Prediction accuracy of the proposed lightweight online profiling method with desired threshold $\eta_a = 0.7, 0.8, 0.9$.

With these results, we evaluated the video analytics performance of the lightweight online profiling based configuration adaptation algorithm on a single video stream in terms of accuracy, total GPU processing time (for inference and profiling) compared to VideoStorm and Chameleon. Figure 11 shows the results of video analytics performance on a single video stream for VideoStorm, Chameleon and the proposed algorithm. We use each of the four video streams as a single video stream. The desired accuracy threshold η_a is set to 0.8. The proposed algorithm runs based on Definition 1 with the lightweight online profiling based accuracy model. In this experiment, without considering an edge based video analytics system, we measure how long it takes to process a single video stream for inference and profiling to ensure the desired accuracy based on each algorithm.

In Figure 11(a), the proposed algorithm guarantees the desired accuracy threshold and achieves almost similar or slightly lower performance on accuracy entirely in comparison of Chameleon, as follows: $\{-0.0078, -0.027, -0.014, -0.039\}$. It means that the prediction of the proposed accuracy model is comparable with the direct measurement of Chameleon. On the other hand, VideoStorm cannot achieve even the desired accuracy threshold performance due to the dynamics of video analytics accuracy which makes its one-time offline profiles invalid. In Figure 11(b), the proposed algorithm achieve better performance on total GPU processing time for inference entirely in comparison of Chameleon, as follows: $\{-0.657457352, -3.233877659, -1.281781197, -13.2078526\}$. As the accuracy of the proposed algorithm is slightly lower than those of Chameleon, its total GPU processing time for inference is lower than Chameleon's. On the other hand, VideoStorm does not use the sufficient GPU processing time for inference to ensure the desired accuracy threshold. In Figure 11(c), the proposed algorithm reduces the total GPU processing time for profiling more than seven times entirely in comparison of Chameleon. The total GPU processing time of the proposed algorithm for profiling is three times smaller than even those of VideoStorm. As a result, the proposed algorithm outperforms Chameleon and VideoStorm on the total GPU processing time (i.e. Inference + Profiling) while ensuring the accuracy similar to Chameleon. Furthermore, the performance of the

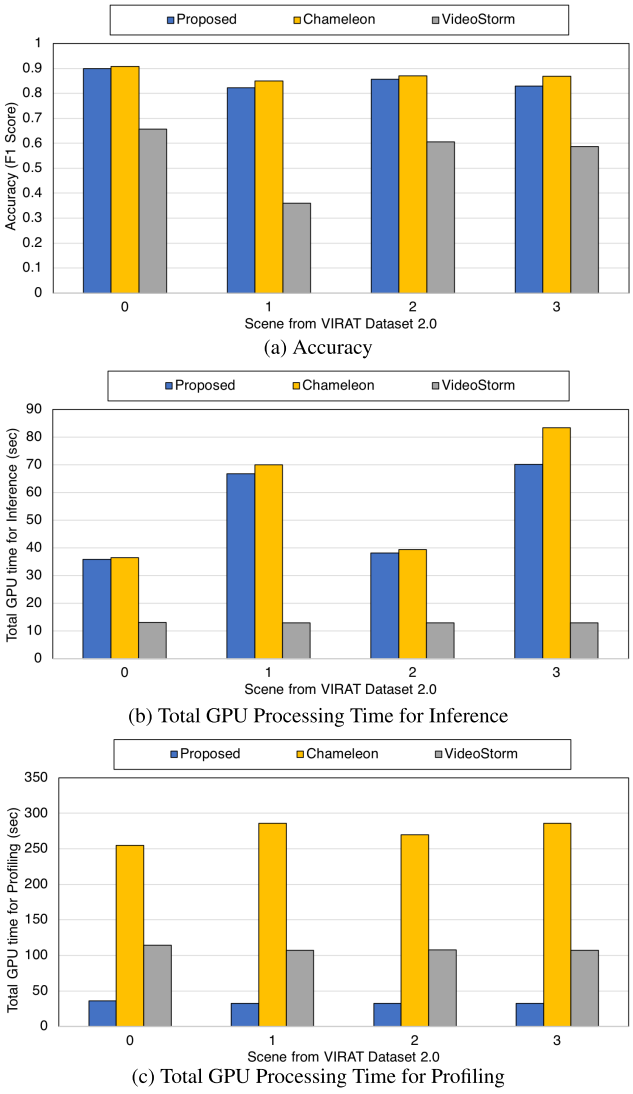
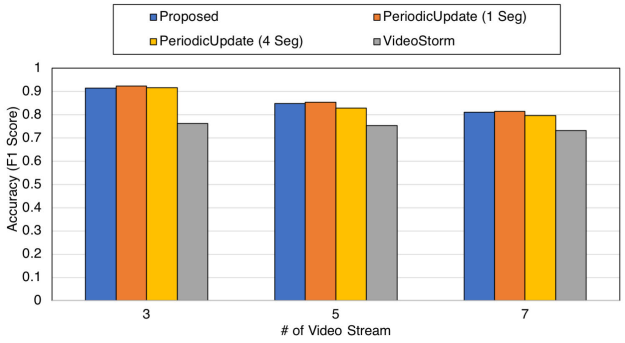


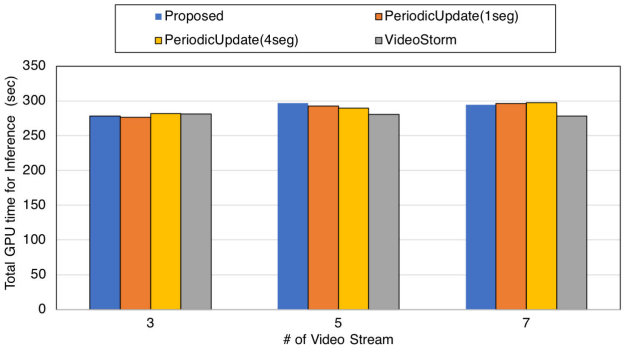
FIGURE 11. Comparision results of the proposed lightweight online profiling based configuraiton adaptation algorithm on single video stream in terms of accuracy (%), total GPU processing time for inference and profiling (sec) with several scenes from VIRAT Dataset 2.0.

proposed algorithm can achieve real-time processing in video analytics, since a profiling process is executed concurrently in an inferencing process, as mentioned in III-E.

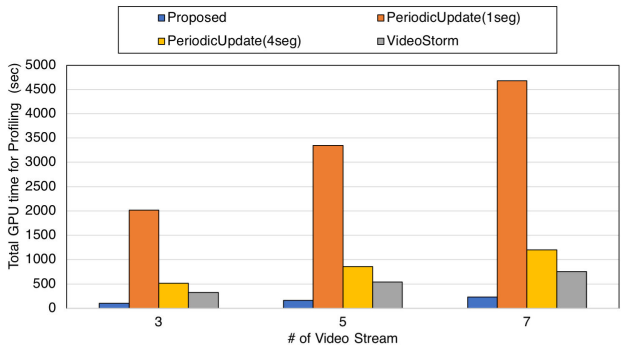
Figure 12 shows the results of video analytics performance on multiple video streams for VideoStorm, Chameleon and the proposed algorithm. The values of I are set as [3, 5, 7]. We pick and duplicate several video streams among the four video streams to realize $I = 3, 5, 7$. The desired accuracy threshold η_a is set to 0.8. The proposed algorithm runs based on the lightweight online profiling based configuration adaptation algorithm, described in Algorithm 1. In this experiment, we consider an edge based video analytics system. Especially, we focus on its limited resource capacity



(a) Accuracy



(b) Total GPU Processing Time for Inference



(c) Total GPU Processing Time for Profiling

FIGURE 12. Comparision results of the proposed lightweight online profiling based configuraiton adaptation algorithm on multiple video streams in terms of accuracy (%), total GPU processing time for inference and profiling (sec) with several scenes from VIRAT Dataset 2.0.

of GPU. The variation of bandwidth $bw_{i,j}$ is not considered. We assume that the radio access capacity are sufficient to handle all video streams with high bandwidth. Since a time slot, which is the time until the next segment is created, is given to process the current segment in each video stream, the maximum tolerable latency of each video stream l_i^{max} is basically set to the one time-slot length T. In addition, the variation of weight γ is not considered. The weight γ is set to a sufficiently large value in order to simplify the objective of $\mathcal{P}_9$ to maximize the average accuracy of video streams. With these setup, we measure how much accuracy degradation can be minimized under the limited resource capacity of GPU, based on each algorithm. Since Chameleon only consider a single video stream without considering the limited resource capacity, we simplify Chameleon to *PeriodicUpdate* for comparison, which profiles all possible configurations periodically by the direct measurement. The configuration decision of *PeriodicUpdate* is made based on Algorithm 1. *PeriodicUpdate*(4 Seg) and *PeriodicUpdate*(1 Seg) mean that the profiles are updated every 4 segments and 1 segment, respectively. Note that *PeriodicUpdate*(4 Seg) is the version without the top-k operation in the chameleon. For the configuration decision of VideoStorm, its scheduling heuristics to maximize sum of utilities is used.

In Figure 12(a)-(b), the proposed algorithm achieves almost similar performance on accuracy and total GPU processing time for inference entirely in comparison of *PeriodicUpdate (1 Seg)* and *PeriodicUpdate (4 Seg)*. The results show that the prediction approach of the proposed accuracy model is comparable with the direct measurement approach even for multiple video streams under the limited resource capacity. On the other hand, VideoStorm shows considerable accuracy degradation relatively, but it consumes the GPU processing time similar to Chameleon for inference. That is, its GPU processing time is wasted and is not contributed to enhance the accuracy enough. In Figure 12(c), the proposed algorithm reduces considerable GPU processing time for profiling entirely in comparison of *PeriodicUpdate (1 Seg)* and *PeriodicUpdate (4 Seg)*. Obviously, *PeriodicUpdate (1 Seg)*, which update the profiles on every segment, consume the highest profiling cost and *PeriodicUpdate (4 Seg)* consume the quarter of *PeriodicUpdate (1 Seg)*'s profiling cost due to the high profiling cost problem of the direct measurement. The proposed algorithm reduces *PeriodicUpdate (4 Seg)*'s profiling cost more than about five times. The profiling cost of the proposed algorithm is about three times better than even those of VideoStorm which updates the profiles in one-time offline. As a result, the proposed method shows the low profiling cost enough to realize real-time processing while achieving the performance comparable to Chameleon for minimizing the accuracy degradation in multiple video streams scenario.

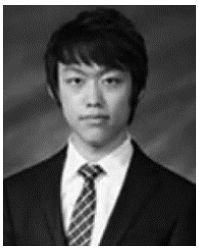
V. CONCLUSION

In this paper, we address the limitation of existing video analytics systems which are inefficient or limited to process

the video analytics of multiple video streams in a GPU-enabled edge server. In addition, their online profiling methods require a high profiling cost. In order to resolve these problems, we design a video analytics system to adapt configurations for optimizing the resource-accuracy tradeoffs of multiple video streams with respect to frame rate and resolution fully under limited resource capacity of a GPU-enable edge server. Utilizing the underlying characteristics of the video objects (i.e. the velocity on location and size), we propose a lightweight online profiling based configuration adaptation algorithm to find the best configuration of each video stream and minimize accuracy degradation of multiple video streams under limited resource capacity of a GPU-enable edge server. We implemented our video analytics system and lightweight online profiling based configuration adaptation algorithm and conducted real video analytics experiments with a subset of surveillance videos and annotations from VIRAT 2.0 Ground dataset. In these experiments, the results showed that our proposed system optimizes configurations and provides the optimized resource-accuracy tradeoffs with respect to frame resolution and rate over the values of weight γ, bandwidth $bw_{i,j}$ and number of users I. Our proposed algorithm reduces the profiling cost of existing video analytics systems more than 3∼7 times while achieving the video analytics performance comparable to them in a single video stream scenario. It also reduces the profiling cost of existing video analytics systems more than 3∼5 times while achieving the video analytics performance comparable to them for minimizing the accuracy degradation in multiple video streams scenario. Our video analtycis system shows several directions for future works. First, the spatio-temporal GPU resource management can be considered to maximize the throughput of real-time DNN tasks, regarding system-level optimizations. Second, other types of hardware accelerators, such as FPGAs, can be applicable to CPUs and GPUs for accelerating DNN tasks.

REFERENCES

[1] *China's 100 Million Surveillance Cameras*. Accessed: 2014. [Online]. Available: https://goo.gl/UK3Obl
[2] *One Surveillance Camera for Every 11 People in Britain, Says CCTV Survey*. Accessed: 2013. [Online]. Available: https://goo.gl/cHLqiK
[3] U.S. Department of Justice. *AMBER Alert*. [Online]. Available: http://www.amberalert.gov/faqs.htm
[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1–9.
[5] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767
[6] G. Ananthanarayanan, P. Bahl, P. Bodik, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.
[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key tech-nology towards 5G," ETSI, Sophia Antipolis, France, ETSI White Paper 11.11, 2015, pp. 1–16.
[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
[9] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, Mar. 1960.
[10] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. Conf. ACM Special Interest Group Data Commun.* New York, NY, USA: ACM, Aug. 2018, pp. 253–266.
[11] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *Proc. 14th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 377–392.
[12] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 1421–1429.
[13] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "AWStream: Adaptive wide-area streaming analytics," in *Proc. Conf. ACM Special Interest Group Data Commun.* New York, NY, USA: ACM, Aug. 2018, pp. 236–252.
[14] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez, "Scaling video analytics systems to large camera deployments," in *Proc. 20th Int. Workshop Mobile Comput. Syst. Appl.* New York, NY, USA: ACM, Feb. 2019, pp. 9–14.
[15] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "VideoEdge: Processing camera streams using hierarchical clusters," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 115–131.
[16] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "NoScope: Optimizing neural network queries over video at scale," *Proc. VLDB Endowment*, vol. 10, no. 11, pp. 1586–1597, Aug. 2017.
[17] Z. Fang, D. Hong, and R. K. Gupta, "Serving deep neural networks at the cloud edge for vision applications on mobile platforms," in *Proc. 10th ACM Multimedia Syst. Conf.* New York, NY, USA: ACM, Jun. 2019, pp. 36–47.
[18] A. H. Jiang, D. L.-K. Wong, C. Canel, L. Tang, I. Misra, M. Kaminsky, M. A. Kozuch, P. Pillai, D. G. Andersen, and G. R. Ganger, "Mainstream: Dynamic stem-sharing for multi-tenant video processing," *USENIX Annu. Tech. Conf. (USENIX ATC)*, 2018, pp. 29–42.
[19] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3464–3468.
[20] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 756–764.
[21] Q. Liu and T. Han, "DARE: Dynamic adaptive mobile augmented reality with edge computing," in *Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP)*, Sep. 2018, pp. 1–11.
[22] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. 39th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Aug. 2020, pp. 1–10.
[23] Y. Suzuki, S. Kato, H. Yamada, and K. Kono, "GPUvm: Why not virtualizing GPUs at the hypervisor?" in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2014, pp. 109–120.
[24] Z. Fang, T. Yu, O. J. Mengshoel, and R. K. Gupta, "QoS-aware scheduling of heterogeneous servers for inference in deep neural networks," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 2067–2070.
[25] M. Ali, A. Anjum, M. U. Yaseen, A. R. Zamani, D. Balouek-Thomert, O. Rana, and M. Parashar, "Edge enhanced deep learning system for large-scale video stream analytics," in *Proc. IEEE 2nd Int. Conf. Fog Edge Comput. (ICFEC)*, May 2018, pp. 1–10.
[26] *Nvidia Developer*. Accessed: 2019. [Online]. Available: https://developer.nvidia.com/
[27] *Keras: The Python Deep Learning Library*. Accessed: 2015. [Online]. Available: https://keras.io/
[28] *Tensorflow: An Open Source Machine Learning Library for Research and Production*. Accessed: 2015. [Online]. Available: https://www.tensorflow.org/
[29] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, and E. Swears, "A large-scale benchmark dataset for event recognition in surveillance video," in *Proc. CVPR*, Jun. 2011, pp. 3153–3160.
[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 740–755.



WOO-JOONG KIM (Member, IEEE) received the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2014, where he is currently pursuing the Ph.D. degree. His research interests include mobile cloud computing, mobile edge computing, and networks.

CHAN-HYUN YOUN (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronics engineering from Kyungpook National University, Daegu, South Korea, in 1981 and 1985, respectively, and the Ph.D. degree in electrical and communications engineering from Tohoku University, Japan, in 1994. From 1986 to 1997, he was the Leader of the High-Speed Networking Team with the KT Telecommunications Network Research Laboratories, where he was involved in the research and developments of centralized switching maintenance systems, high-speed networking, and ATM networks. Since 2009, he has been a Professor with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, where he was an Associate Vice-President of office of planning and budgets, from 2013 to 2017. He is currently the Director of the Grid Middleware Research Center and the XAI Acceleration Technology Research Center, KAIST, where he is developing core technologies. He wrote a book on *Cloud Broker and Cloudlet for Workflow Scheduling* (Springer), in 2017. His research interests include high-performance computing, edge-cloud computing, AI acceleration systems, and so on. He has served many international conferences as a TPC Member. He was a General Chair of the sixth EAI International Conference on Cloud Computing (Cloud Comp 2015), KAIST, in 2015. He has served as a Guest Editor for the IEEE WIRELESS COMMUNICATIONS, in 2016.