Gemini+: Enhancing Real-Time Video Analytics with Dual-Image FPGAs

Jingrou Wu, Chuang Hu*, Jin Zhang*, *Member, IEEE,* Dan Wang, *Senior Member, IEEE,* and Jing Jiang

***Abstract*—Real-time video analytics demand intensive computing resources, often exceeding device capabilities. Heterogeneous computing resources like CPU and GPU, usually work collaboratively to ensure real-time performance. GPU manage data-intensive computing, while CPU handle instruction-intensive tasks. However, analytics accuracy can degrade because of the imbalance between CPU-GPU workloads and resources, and static resources in most systems limit adaptability to dynamic workloads. In addition to CPU-GPU resource control, accuracy is highly dependent on video analytics configuration including resolution, frame rate, and model selection. In this paper, we propose Gemini+, a hardware-accelerated video analytics pipeline empowered by dual-image FPGAs. It enables flexible CPU-GPU resource adaptation by providing near-instantaneous switching between two pre-configured images. We investigate CPU-GPU resource control and video analytics configuration adaptation in a dual-image FPGA-based video analytics pipeline. We study and formulate two problems of practical importance, a single-camera and multi-camera dual computing resource control problem, i.e., SC-DCRC and MC-DCRC. We analyze the problem complexity and develop optimal and sub-optimal algorithms. We evaluate our algorithms through simulation and build a prototype for verification. The results show that Gemini+ can improve analytic accuracy by 25% compared to fixed CPU-GPU resource systems and 88% compared to GPU-dominant systems.**

***Index Terms*—Video analytics, Real time, CPU-GPU resource control, Dual-image FPGA**

I. Introduction

With the increasing amount of smart cameras [1] and the development of computer vision, video analytics has been widely used in daily life like auto-driving and Metaverse. Though it is technically possible to rely solely on DNN/CNN model inference with GPU resources for video analytics, recent systems [2]–[4] tend to use a combination of CPU and GPU resources to reduce processing latency. CPU is good at instruction-intensive computing and hence used to filter out unimportant frames and/or track detected objects. GPU supports data-intensive computing such as DNN/CNN model inference to detect objects.

However, existing video analytics systems suffer from fixed computing resources, leading to imbalanced CPU-GPU workloads that degrade performance. For example, a $10\% - 90\%$ CPU-GPU system (details in section IV) performs well on a 10 frames per second (fps) video but is ineffective on a 30 fps video. This is because the 30 fps video demands more CPU resources and fewer GPU resources than the 10 fps video.

Inspired by our previous work, Gemini [5], we leverage the newly developed hardware, dual-image FPGAs, to enhance video analytics by providing flexible CPU-GPU resources. Dual-image FPGAs support runtime switching between two pre-configured FPGA images. This allows a flexible CPU-GPU framework by pre-configuring the FPGA with a CPU-image for instruction-intensive tasks and a GPU-image for data-intensive tasks. These two images are multiplexed in the time domain, ensuring efficient utilization of computing resources by dynamically switching between different processing patterns as needed.

Limitations of Status Quo. Computing resource control and video analytics configuration selection are two essential components in video analytics systems with flexible CPU-GPU resources. Computing resource control involves the dynamic allocation of resources between the CPU and GPU, while video analytics configuration selection includes selecting configurations, such as video solution, sampled frame rate, and the specific DNN/CNN model. Most existing works focus only on video analytics configuration selection in fixed CPU-GPU resource systems, as listed in Table I. GPU-dominant systems like DeepDecision [6] and FastVA [7] optimize resolution and model selection, whereas fixed CPU-GPU systems leverage CPU computing resources. For instance, MARLIN [3] and AdAVP [4] consider object detection (GPU-based) and tracking (CPU-based) by selecting the most suitable DNN/CNN model and tracking parameters. Authors in the work [2] propose a threshold control algorithm for the CPU filtering function using a fixed frame resolution and DNN/CNN model. These approaches fail to fully exploit the heterogeneous computing resources and adapt CPU-GPU resources to dynamic video contents due to fixed CPU-GPU resources.

Though Gemini [5] addresses some above issues by offering flexible CPU-GPU computing resources, it still struggles with various video scenarios due to an ineffective CPU-GPU workload partition. This inefficiency arises for two main reasons.

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant T2495254. *(Corresponding authors: Chuang Hu; Jin Zhang)*

Jingrou Wu is with Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, NSW 2007, Australia, and also with the Research Institute of Trustworthy Autonomous Systems and the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: jingrou.wu@student.uts.edu.au).

Chuang Hu is with the School of Computer Science, Wuhan University, Wuhan 430072, China. (e-mail: handc@whu.edu.cn).

Jin Zhang is with the Department of Computer Science and Engineering and the Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen 518055, China, and also with Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: zhangj4@sustech.edu.cn).

Dan Wang is with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: csdwang@comp.polyu.edu.hk).

Jing Jiang is with Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, NSW 2007, Australia. (e-mail: jing.jiang@uts.edu.au).

TABLE I
COMPARISON WITH THE EXISTING WORK

	Objective	Multiple Users	CPU Functions	GPU Functions	Resource Adjustment
Deepdecision [6]	Composite utility of frame rate and accuracy	×	Resizing	Multiple DNN/CNN models	GPU-dominant
FastVA [7]	Accuracy; Composite utility of delay and accuracy	×	Resizing	Multiple DNN/CNN models	GPU-dominant
JCAB [8]	Composite utility of energy consuming and accuracy	✓	Resizing	Multiple DNN/CNN models	GPU-dominant
O^3 [2]	Accuracy	×	Filtering, tracking	Single DNN/CNN model	Fixed CPU-GPU
MARLIN [3]	Balance between energy and accuracy	×	Change detector, tracking	Single DNN/CNN model	Fixed CPU-GPU
AdaVP [4]	Accuracy	×	Resizing, tracking	Single DNN/CNN model	Fixed CPU-GPU
Gemini [5]	Accuracy	×	Filtering, background subtraction, cropping	Multiple DNN/CNN models	Flexible CPU-GPU
Ours	Accuracy	✓	Resizing, tracking	Multiple DNN/CNN models	Flexible CPU-GPU

First, the frame filtering and background subtraction module implemented on the CPU-image do not always effectively reduce the GPU workloads. For example, in a busy traffic scenario, the CPU fails to filter frames, leading to excessive GPU load. Second, Gemini uses a bandit-based approach to address computing resource control and video analytics configuration selection. While bandit algorithms explore different video analytics configurations, they suffer from performance instability in dynamic environments and slow adaptation to sudden video content changes. These limitations hinder Gemini's effectiveness in diverse and changing video scenarios. Moreover, Gemini can not support multiple users.

Challenges and Contributions. In this paper, we address these limitations by proposing Gemini plus (Gemini+), a dual-image FPGA-based video analytics pipeline with a more efficient and robust CPU-GPU workload partition strategy and support for multiple users. Specifically, we reorganize the FPGA implementation by deploying resizing and tracking modules on the CPU-image to leverage its capability for instruction-driven tasks. In contrast, the GPU-image is configured to execute data-intensive tasks such as DNN/CNN model inference. The tracking module allows us to more efficiently allocate CPU-GPU workloads by adjusting tracking frame proportions. Moreover, we provide a comprehensive model of the video analytics pipeline with dual computing resources and propose novel algorithms for resource allocation and video analytics configuration selection. These algorithms dynamically adjust to the varying workloads and video content, hence improving analytics performance. We face two main challenges in fully utilizing flexible computing resources.

First, the optimal CPU-GPU workload partition changes with video characteristics, requiring the system to adapt quickly to dynamic resource demands. Frame rates influence the best CPU-GPU workload partition by altering the total workload, while video content impacts the ideal partition due to the varying performance of the CPU-image and GPU-image. For example, videos with slow-moving objects benefit more from higher CPU workload due to the tracking module's superior accuracy for such targets, whereas fast-moving videos demand greater GPU processing. Therefore, the optimal CPU-GPU partition strategy and resource control scheme should adapt to diverse video types.

Second, it is challenging to jointly decide the CPU-GPU resource control and video analytics configuration selections in the flexible computing resource video analytics system, because they have mutual influences on each other. For example, if we allocate limited computing resources to the GPU-image, it may not be possible to run a complex DNN/CNN model. Conversely, if we increase the number of frames fed to the CPU-image, we need to assign more resources to it. As a result, it is necessary to consider CPU-GPU resource division and video analytics configuration selection in a joint manner.

To overcome the above challenges, we propose a new problem formulation of CPU-GPU resource control and video analytics configuration selection in the video analytics pipeline. We formulate the dual computing resource control problem under two scenarios. In the single-camera scenario, the pipeline runs on FPGA-powered smart cameras, each with flexible CPU-GPU resources. We propose an optimal solution that effectively utilizes the available flexible resources. In the multi-camera scenario, the dual-image FPGA acts as an edge server that provides video analytics services for multiple cameras. We propose a suboptimal yet highly effective solution.

II. System Model and Problem Formulation

A. System Architecture Overview

The architecture of our proposed dual-image FPGA-based system, Gemini+, is designed to provide flexible heterogeneous computing resources for real-time video analytics. As shown in Fig. 1, the Gemini+ system consists of multiple cameras for video capture, an edge device[1] responsible for video analytics optimization and FPGA control, and a dual-image FPGA that performs specific video analytics tasks.

Considering that model performance is influenced by video content, the **content analyzer** in the edge device periodically selects frames to extract critical information, such as object size and motion speed, enabling the system to adapt dynamically to diverse video scenarios. We select object size and

[1]The edge device can either be a standalone processing unit or a microprocessor embedded within a smart camera.

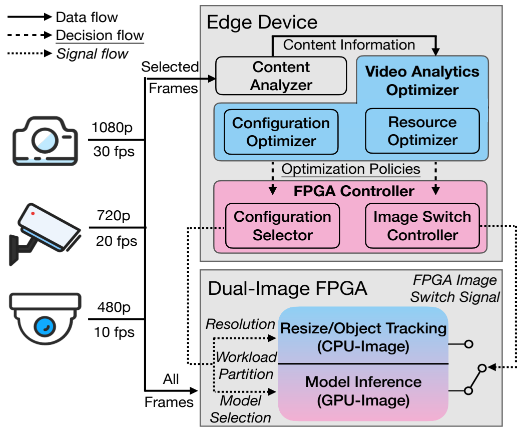


Fig. 1. The system architecture of Gemini+.

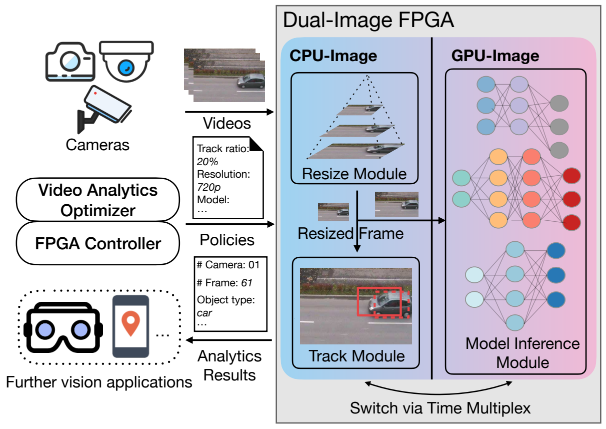


Fig. 2. The video analytics pipeline of Gemini+.

speed as video features because they significantly influence DNN/CNN model performance. Additionally, these features can be efficiently extracted with lightweight algorithms, making them well-suited for real-time video analytics. The framework remains flexible and can incorporate additional features if needed. The information is then utilized by the **video analytics optimizer** to generate video analytics optimization policies. These policies guide the **FPGA controller** in computing resource control and video analytics configuration selection, including workload partition, resolution adjustment, and DNN/CNN model selection.

The dual-image FPGA executes video analytics tasks based on signals from the **FPGA controller**. It contains two pre-configured images: the CPU-image and the GPU-image. The CPU-image is optimized for instruction-intensive tasks including resizing and object tracking, while the GPU-image is dedicated to data-intensive tasks like DNN/CNN model inference. The dual-image FPGA runs each image with video analytics configuration and switches the two images according to the image switch signal. An alternative approach is space-division multiplexing, where CPU and GPU processing units are implemented within a single FPGA image using separate regions of logic cells. However, since dual-image FPGAs typically support only two images, space-division multiplexing allows for only two CPU-GPU resource allocations. To support additional allocations beyond the pre-defined ones, it requires reconfiguring the entire FPGA, which takes several minutes. Additionally, video analytics does not always require CPU and GPU operations to run simultaneously, as video frames arrive sequentially. In such cases, space-division multiplexing may lead to resource idleness because one type of processing unit remains underutilized while waiting for the other to complete. In contrast, the time-division design ensures optimal utilization of computing resources by dynamically switching between the CPU-image and GPU-image within 1 ms, making it more suitable for diverse and dynamic video analytics workloads.

B. Modeling the Video Analytics Process

The video analytics pipeline in Gemini+ is composed of three key stages: resizing, object tracking, and DNN/CNN model inference, as illustrated in Fig. 2. At the start of each epoch (duration T), video clips captured by cameras are uploaded. During each epoch, the **FPGA controller** transmits optimization policies generated by the **video analytics optimizer**, including video analytics configuration selection and the FPGA image switch scheme. Based on these policies, the video clips are resized to the desired resolution and directed to either the tracking module or the model inference module.

Gemini+ supports video inputs with varying frame rates and resolutions. At each epoch, a video clip v_k captured by the k-th camera has features of a frame rate f_k and a resolution r_k^o. Therefore, the k-th video clip has $n_k = Tf_k$ frames that need to be processed in an epoch. The resizing step is crucial because high-resolution videos can lead to high accuracy but also result in long processing delays. Therefore, to meet real-time requirements, the input video is resized to a lower resolution. The resizing module adjusts the frame size to the desired resolution r from the resizing resolutions set $\mathcal{R} = \{r_1, r_2, ..., r_R\}$. After resizing, the system should select a DNN/CNN model m from the models set $\mathcal{M} = \{m_1, m_2, ..., m_M\}$ to detect objects in videos. However, it is time-consuming to process all frames via DNN/CNN models, especially when dealing with large volumes of video data. To address this challenge, Gemini+ takes advantage of the spatiotemporal correlations within video frames. Object tracking is used to estimate the position of objects across multiple frames instead of model inference. Consequently, a proportion of frames γ^t are sampled to be processed by the tracking module in the CPU-image, and a proportion of frames γ^d are handled by DNN/CNN model inference in the GPU-image, while other frames $(1 - \gamma^t - \gamma^d)$ are dropped. As for FPGA image switch scheme, we adopt a time-division multiplexing image switch approach, where the CPU-image and GPU-image run at separate times. Therefore, the CPU-GPU resource control can be considered equal to the time division. We need to decide on resource division for each video clips τ_k^c, τ_k^g and the total resource division T^c, T^g.

In summary, the **video analytics optimizer** need to determine the appropriate resolution r_k, DNN/CNN model m_k, tracking frames proportion γ_k^t and detection frames proportion γ_k^d, collectively known as the video analytics configuration $c_k = (r_k, m_k, \gamma_k^t, \gamma_k^d)$. These decisions are critical to ensuring optimal resource division and system performance.

C. Modeling the Video Analytics Accuracy

The Gemini+ system aims to maximize accuracy within time constraints. The analytics accuracy a is measured by the average accuracy of K video clips,

$$a = \frac{1}{K}\sum_{k=1}^{K} a_k. \tag{1}$$

The accuracy a_k for the video clip k is jointly determined by the detection accuracy a_k^d and sampling discount rate a_k^s. The detection accuracy a_k^d is based on the detection results of DNN/CNN models, while the sampling discount rate a_k^s is determined by the proportion of frames not processed by DNN/CNN models. According to the previous work [8], we assume that the impact of detection accuracy a_k^d and sampling discount rate a_k^s are independent, and the accuracy is the multiplication of detection accuracy and sampling discount rate,

$$a_k = a_k^d a_k^s. \tag{2}$$

Detection accuracy. The detection accuracy a_k^d is jointly determined by the DNN/CNN model $m_{i,k}$ and the video resolution r_k. A binary indicator $x_{i,k}$ is used to denote DNN/CNN model selection,

$$x_{i,k} = \begin{cases} 1, & \text{Model } i \text{ is selected for the video clip } k, \\ 0, & \text{Otherwise.} \end{cases}$$

Given the model accuracy function $a_{i,k}^m(r_k)$ of each DNN/CNN model with resolution r_k, the detection accuracy a_k^d is the selected model accuracy,

$$a_k^d = \sum_{i=1}^{M} x_{i,k} a_{i,k}^m(r_k). \tag{3}$$

According to the previous work [8] and our experiments results in section IV, the model accuracy function $a_{i,k}^m$ is a concave function with the resolution r_k, $a_{i,k}^m(r_k) = c_{i,k,1}^d - c_{i,k,2}^d e^{-\frac{r_k}{c_{i,k,3}^d}}$, where $c_{i,k,1}^d, c_{i,k,2}^d, c_{i,k,3}^d$ are three fitting coefficients for the concave function, which can be obtained by experiments. The model accuracy functions vary for different DNN/CNN models and video content. For instance, the resolution has a greater impact on complex DNN/CNN models, such as Yolov3, compared to simpler ones, like Yolov2-tiny. Additionally, the detection accuracy is influenced by the characteristics of the objects in the video. For example, a small object benefits more from a higher resolution than a larger object. Thus, we require different accuracy functions profiling for different video clips and DNN/CNN models.

Sampling discount rate. Due to the limited time, only a proportion of frames γ^d can be processed by DNN/CNN models. Then a proportion of the rest γ^t is processed by the tracking module, while others $1 - \gamma^t - \gamma^d$ are skipped. Both tracking and skipped frames can have a negative effect on detection accuracy. In tracking frames, the position of the detection box obtained from DNN/CNN model inference results, is adjusted based on tracking results. As for the skipped frame, we use the same result as the nearest unskipped frame. Although both methods may result in a reduction of accuracy, tracking frames typically offer superior performance compared to skipped frames, as they utilize more information between frames. We use sampling discount rate a_k^s to denote such a negative effect. The discount rate is jointly determined by the proportion of detection frames γ^d and tracking frames γ^t. Given the fixed proportion of detection frames γ^d, the sampling discount rate increases with tracking frames γ^t. Conversely, when the proportion of skipped frames $1-\gamma^t-\gamma^d$ is fixed, the sampling discount rate decreases with tracking frames γ^t. We introduce the sampling discount rate function $a_k^s(\gamma_k^d, \gamma_k^t)$, to quantify the impact of detection and tracking frame proportions. However, it lacks a specific mathematical formula due to its complexity and variability. Instead, the function values are derived from experiments.

D. Modeling the Video Analytics Delay

The system delay is the sum of processing time for each video clip,

$$D = \sum_{k=1}^{K} D_k. \tag{4}$$

The k-th video clip delay D_k consists of the CPU processing delay D_k^{cpu} and GPU processing delay D_k^{gpu},

$$D_k = D_k^{cpu} + D_k^{gpu}. \tag{5}$$

CPU processing delay. In the CPU-image, both resizing and tracking modules take time. We use an interpolation resizing method and a feature-based tracking method in the CPU-image. The running time of the interpolation resizing method $t^r(r_k)$ is proportional to the square of resolution of the output figure r_k,

$$t^r(r_k) = T_r r_k^2, \tag{6}$$

where T_r is the resizing time for each pixel. Exceptionally, if the output resolution is equal to the original resolution, i.e., $r_k = r_k^o$, we regard that the frame skips the resizing method and hence the resizing time is 0 in this situation. Similarly, the tracking delay $t^t(r_k)$ is proportional to the resolution of the input frame,

$$t^t(r_k) = T_t r_k^2, \tag{7}$$

where T_t is the tracking time for each pixel. Hence, in the CPU-image, $n_k(\gamma_k^t + \gamma_k^d)$ frames are resized and $n_k\gamma_k^t$ frames are tracked. The CPU processing delay of a single video clip is the sum of all frames delay,

$$D_k^{cpu} = n_k((\gamma_k^t + \gamma_k^d)t^r(r_k) + \gamma_k^t t^t(r_k)). \tag{8}$$

GPU processing delay. In the GPU-image, the object detection time of the DNN/CNN model $m_{i,k}$ with the frame resolution r_k is denoted by $t_i^d(r_k)$. With the binary indicator variable $x_{i,k}$, the GPU-image processing delay can be calculated as

$$D_k^{gpu} = n_k \gamma_k^d \sum_{i=1}^{M} x_{i,k} t_i^d(r_k). \tag{9}$$

Similar to the previous works [9], [10], we use the floating-point operations (FLOPs) to measure the detection time,

$$t_i^d(r_k) = F_i r_k + c_i^t, \tag{10}$$

TABLE II
TABLE OF NOTATIONS

Variable	Description
τ_k^c	The proportion of the CPU-image running time for kth video clip
τ_k^g	The proportion of the GPU-image running time for kth video clip
T^c	The total CPU-image running time
T^g	The total GPU-image running time
T	The time constraint in an epoch
r_k	The frame resolution for kth video clip
$x_{i,k}$	The indicator of the jth DNN/CNN model for kth video clip
γ_k^t	The proportion of tracking frames for kth video clip
γ_k^d	The proportion of detection frames for kth video clip
a_k	The accuracy of the kth video clip
D_k^{cpu}	The CPU-image delay for the kth video clip
D_k^{gpu}	The GPU-image delay for the kth video clip
D_k	The total delay for the kth video clip
D	The system delay

where the F_i, c_i^t represents the number of FLOPs required per unit resolution by the selected DNN/CNN model i, and c_i^t denotes the model-specific constant time overhead. All notations are listed in Table II.

E. Problem Formulation and Complexity Analysis

1) The Single-Camera Dual Computing Resource Control (SC-DCRC) Problem: We first consider the DCRC problem when there is only one camera in the system. The case is valuable when the FPGA-assisted smart cameras are available as works [11], [12] do. This problem provides insights into the optimal video analytics configuration selection and the CPU-GPU resource control.

The single-camera problem indicates $K = 1$ and therefore we drop the user index k in this scenario for brevity. In this case, we do not need to consider the time allocation among multiple cameras, but we should determine the optimal video analytics configuration selection and CPU-GPU resource control. We formulate the SC-DCRC problem as follows,

$$P1: \max_{\mathbf{x}, r, \gamma^c, \gamma^d, T^c, T^g} a(\mathbf{x}, r, \gamma^c, \gamma^d) \tag{11}$$

$$\text{s.t. } D^{cpu} \le T^c \tag{12a}$$
$$D^{gpu} \le T^g \tag{12b}$$
$$T^c + T^g \le T \tag{12c}$$
$$\sum_{i=1}^{M} x_i = 1, x_i \in \{0, 1\}. \tag{12d}$$

Theorem 1. *The SC-DCRC problem can be solved in polynomial time.*

Proof. We can design an algorithm to search the solution space. We need to search all possible values for each parameters, i.e., the DNN/CNN model in $\mathcal{M}$, resolution in R, tracking frames and detection frames $\{\frac{1}{n}, \frac{2}{n}, ..., 1\}$, CPU-image and GPU-image resource in $\{\frac{1}{n_T}, \frac{2}{n_T}, ..., 1\}$. As such, the total solution space is $O(M \times R \times n^2 \times n_T^2)$. □

Though the problem is polynomial, a brute force searching of the solution space is still time consuming. We will develop a much more efficient algorithm in Section III.

2) The Multi-Camera Dual Computing Resource Control (MC-DCRC) Problem: In the multi-camera system, the FPGA acts as an edge server that provides video analytics services for multiple cameras as works [13], [14] do. In this scenario, we should allocate CPU-GPU resources and select the optimal video analytics configuration for each video clip. We formulate the MC-DCRC problem as follows,

$$P2: \max_{\mathbf{x}, \mathbf{r}, \vec{\gamma}^d, \vec{\gamma}^t, \vec{\tau}^c, \vec{\tau}^g, T^c, T^g} \frac{1}{K} \sum_{k=1}^{K} a_k(x_k, r_k, \gamma_k^c, \gamma_k^d) \tag{13}$$

$$\text{s.t. } D \le T \tag{14a}$$
$$D_k^{cpu} \le \tau_k^c T^c \tag{14b}$$
$$D_k^{gpu} \le \tau_k^g T^g \tag{14c}$$
$$T^c + T^g \le T \tag{14d}$$
$$\sum_k \tau_k^c = 1, \sum_k \tau_k^g = 1 \tag{14e}$$
$$\sum_{i=1}^{M} x_{i,k} = 1, x_{i,k} \in \{0, 1\}, \forall v_k \in V. \tag{14f}$$

The objective function (13) aims to maximize the average accuracy in an epoch. The problem follows the constraints: The system delay should not exceed the time constraint (14a); The CPU-image delay of a single video clip should not exceed the assigned CPU-image time (14b); The GPU-image delay of a single video clip should not exceed the assigned GPU-image time (14c); The total CPU-image and GPU-image running time should be less than the time constraint (14d); All cameras share the CPU-image and GPU-image processing time (14e); One and only one DNN/CNN model can be selected (14f).

Theorem 2. *The MC-DCRC problem is NP-hard.*

Proof. We prove this theorem by transforming the problem into the 2-D bounded knapsack problem (BKP). Consider a special case that video clips are processed by the GPU-image only, i.e., $r_k = r_{ori}, \gamma_k^t = 0, \tau_k^c = 0, \forall k \in \{1, 2, ..., K\}$. In this way, only constraint (14a) and (14f) remain. Let DNN/CNN models be the items and hence the MC-DCRC problem is equivalent to the 2-D BKP, which aims to maximize the value by determining the number of each item $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ within two constraints. □

III. ALGORITHMS

A. The SC-DCRC Algorithm

Though we could find a polynomial-time brute force algorithm as Theorem 1 indicates, it is impractical especially when there is a large number of candidate parameters to select. We need to reduce the algorithm complexity by leveraging the mathematical structure of the problem.

The search space can be divided into two parts, i.e., the video analytics configuration selection and CPU-GPU resource control. In the video analytics configuration selection, Theorem 3 helps reduce the search space of resolution by providing a resolution threshold. Theorem 4 provides the optimal relationship between detection and tracking frames. In the CPU-GPU resource control part, Theorem 5 gives the best resource division strategy. With these theorems, we can reduce the search space significantly.

Definition 1. *The detection configuration $c_1^d = (m_{i_1}, r_{i_1})$ is said to be* ***superior*** *to the detection configuration $c_2^d = (m_{i_2}, r_{i_2})$, if $a^*(c_1^d) > a^*(c_2^d)$ where $a^*(c^d)$ is the optimal value of the objective function* (11) *with the corresponding detection configuration.*

Lemma 1. *The detection configuration $c_1^d = (m_{i_1}, r_{i_1})$ is superior to the detection configuration $c_2^d = (m_{i_2}, r_{i_2})$, if the detection accuracy $a^d(c_1^d) \geq a^d(c_2^d)$ and the accuracy per latency $\frac{a^d(c_1^d)}{t_1^d} > \frac{a^d(c_2^d)}{t_2^d}$.*

Proof. For simplification, we use a_1^d, a_2^d to represent the detection accuracy $a^d(c_1^d), a^d(c_2^d)$ in the proof. We assume that the relationship between two detection configurations is $a_1^d = k_a a_2^d$ and $t_1^d = k_t t_2^d$ where $k_a, k_t > 1$ and $k_a > k_t$. Let γ_2^d, γ_2^t be the best proportion of detection and tracking frames with the detection configuration c_2^d. We have the optimal accuracy as

$$a_2^* = a_2(\gamma_2^d, \gamma_2^t) = a_2^d a^s(\gamma_2^d, \gamma_2^t) \tag{15}$$

We choose $\frac{\gamma_2^d}{k_a}, \gamma_2^t$ for the detection configuration c_1^d which satisfies the total time constraint Eq. (12c) because $D_1(\frac{\gamma_2^d}{k_a}, \gamma_2^t) < D_1(\frac{\gamma_2^d}{k_t}, \gamma_2^t) \leq D_2(\gamma_2^d, \gamma_2^t) \leq T$. Then, we have the accuracy as

$$a_1(\frac{\gamma_2^d}{k_a}, \gamma_2^t) = k_a a_2^d a^s(\frac{\gamma_2^d}{k_a}, \gamma_2^t) = a_2^d k_a a^s(\frac{\gamma_2^d}{k_a}, \gamma_2^t) \tag{16}$$

As we mentioned in section II-C, the sampling accuracy function is a concave function with both γ_d, γ_k and hence we have $k_a a^s(\frac{\gamma_2^d}{k_a}, \gamma_2^t) > a^s(\gamma_2^d, \gamma_2^t)$. Because $a_1^* > a_1(\frac{\gamma_2^d}{k_a}, \gamma_2^t)$ and $a_1(\frac{\gamma_2^d}{k_a}, \gamma_2^t) > a_2^*$, we have $a_1^* > a_2^*$. □

Based on the above definition and lemma, Theorem 3 provides a smaller search space of resolution.

Theorem 3. *There exists a resolution threshold.*

$$r_i^{th} = f^{-1}(c_{i,3}^d c_{i,1}^d F_i) \tag{17}$$

where $f(r) = c_{i,2}^d e^{-r/c_{i,3}^d}(F_i r + c_i^t + c_{i,3}^d F_i)$, for each DNN/CNN model i. With the same DNN/CNN model, the detection configuration with the resolution threshold is superior to the ones with lower resolutions.

Proof. For the same DNN/CNN model, the detection accuracy increases with the resolution, i.e., $a^d(i, r_i^{th}) > a^d(i, r_i), r_i^{th} > r_i$. According to the Lemma 1, we are able to find the resolution threshold r_i^{th} by solving the following problem,

$$P1.1 : \max_{r_i^{th}} \frac{a^d(i, r_i^{th})}{t_i^d(r_i^{th})}. \tag{18}$$

The resolution threshold is $r_i^{th} = f^{-1}(c_{i,3}^d c_{i,1}^d F_i)$. □

Lemma 2. *The optimal detection-tracking frame assignment (γ^d, γ^t) satisfies $D^{cpu} + D^{gpu} = T$.*

Proof. We assume that the optimal frame assignment solution (γ^d, γ^t) does not satisfy $D^{cpu} + D^{gpu} = T$, we are able to find a new solution $(\gamma^d + \epsilon, \gamma^t)$ satisfies the equation. Because the objective function increases with γ^d, the new solution has higher accuracy than the original one. □

Theorem 4 indicates the optimal detection-tracking frame assignment so that we do not need to try all detection and tracking frames combinations.

Theorem 4. *The optimal detection-tracking frame assignment (γ^d, γ^t) satisfies*

$$\gamma^t = \frac{\frac{T}{f} - \gamma^d(t^d + t^r)}{t^t + t^r}. \tag{19}$$

Proof. According to the Lemma 2, we have $f\gamma^t t^t + f(\gamma^t + \gamma^d)t^r + f\gamma^d t^d = T$. Thus, we can get the relationship between the proportion of detection frames and tracking frames $\gamma^t = \frac{\frac{T}{f} - \gamma^d(t^d + t^r)}{t^t + t^r}$. □

Definition 2. *The direct resource division T^{c*}, T^{g*} is a feasible resource division for the video analytics configuration $c = (m, r, \gamma^d, \gamma^t)$ which satisfies $D^{gpu}(m, r, \gamma^d) = T^{g*}$. Other feasible resource division schemes are said to be indirect resource divisions.*

Theorem 5 describes that resource divisions do not affect accuracy as long as they are feasible for the same video analytics configuration. Therefore, we only need direct resource division instead of searching the whole resource division space.

Theorem 5. *Direct resource division has the same accuracy as indirect resource divisions for the same configuration.*

Proof. We prove it by contradiction. We assume the accuracy for the video analytics configuration c with the direct resource division is a^{di} and the accuracy with the indirect resource division is a^{in} and $a^{di} \neq a^{in}$. According to the objective function (11), we have $a^{di} = a(c)$ and $a^{in} = a(c)$ which indicates that $a^{di} = a^{in}$ and it is contradict to the assumption. □

Algorithm 1: The SC-DCRC algorithm

```
   Input: $T, \mathcal{M}, \mathcal{R}, a(\cdot)$
   Output: $\gamma^{d*}, \gamma^{t*}, r^*, m^*, T^{c*}, T^{g*}$
1  $a^* \leftarrow 0$;
2  foreach DNN/CNN model $i \in \mathcal{M}$ do
3      Obtain the resolution threshold $r_i^{th} = f^{-1}(c_{i,3}^d c_{i,1}^d F_i)$ by
       Theorem 3;
4      foreach Resolution $r \in \mathcal{R}, r \geq r_i^{th}$ do
5          $n_d^{\max} \leftarrow \min\{n, \lfloor \frac{T}{t^s(r)+t^r(r)} \rfloor\}$;
6          for Detection frames $n_d \leftarrow 1$ to $n_d^{max}$ do
7              $\gamma^d = \frac{n_d}{f}$;
8              Obtain the proportion of tracking frames
               $\gamma^t = \frac{\frac{T}{f} - \gamma^d(t^d+t^r)}{t^t+t^r}$ by Theorem 4;
9              if $\gamma^t \geq 0$ and $a(i, r, \gamma^t, \gamma^d) > a^*$ then
10                 $a^* \leftarrow a(i, r, \gamma^t, \gamma^d)$;
11                 $m^* \leftarrow i, r^* \leftarrow r$;
12                 $\gamma^{d*} \leftarrow \gamma^d, \gamma^{t*} \leftarrow \gamma^t$;
13 Obtain the direct resource division $T^{g*}$, $T^{c*}$;
14 return $\gamma^{d*}, \gamma^{t*}, r^*, m^*, 1 - T^{g*}, T^{g*}$
```

Based on the above theorems, we propose an efficient SC-DCRC algorithm as the Alg. 1 shows. First of all, we try each DNN/CNN model (line 2). Given the DNN/CNN model i, we find the resolution threshold $\bar{r}_i$ based on the Theorem 3 (line 3). Only the resolutions greater than $\bar{r}_i$ are taken into consideration (line 4). Then according to the Theorem 4, we are able to find the best proportion of detection and tracking frames γ^d, γ^t (lines 5-8). We keep the video analytics configuration with the highest accuracy (lines 9-12). Finally, a direct CPU-GPU resource division for the optimal video analytics configuration by Theorem 5 is provided (line 13).

The computation complexity of the SC-DCRC algorithm is $O(|\mathcal{M}| \times |\mathcal{R}| \times n)$ as it searches the space of the DNN/CNN models, partial resolutions and detection frames.

B. The MC-DCRC Algorithm

We propose an efficient heuristic algorithm for the MC-DCRC problem based on several observations.

Observation 1. *The MC-DCRC problem can be regarded as a two-step problem. The first step is to allocate time resources for each user and the second step is to find the optimal video analytics configuration for each camera.*

Given the time allocation, the optimal accuracy $a_k^*(t_k)$ can be derived from the SC-DCRC algorithm. Therefore, we only need to focus on the time allocation problem in the first step which reduces the original problem as follows,

$$P2.1: \max_{\mathbf{t}, \vec{\tau}^c, \vec{\tau}^g, T^c, T^g} \frac{1}{K}\sum_{k=1}^{K} a_k^*(t_k) \tag{20}$$

$$\text{s.t. } \tau_k^c T^c + \tau_k^g T^g \le t_k \tag{21a}$$
$$T^c + T^g \le T \tag{21b}$$
$$\sum_k \tau_k^c = 1, \sum_k \tau_k^g = 1. \tag{21c}$$

Theorem 6. *The reduced time allocation problem is NP-hard.*

Proof. The reduced time allocation problem can be transformed to the generalized knapsack problem [15] (GKP). Each camera is the item and the value function is the accuracy function with the allocated time $a^*(t)$. We can reduce the 0-1 knapsack problem to the GKP and the 0-1 knapsack problem is NP-hard. Therefore, the GKP is NP-hard and the reduced time allocation problem is NP-hard. □

Though the reduced problem is still NP-hard, compared with the original problem, the property of the accuracy function $a_k^*(t_k)$ provides us with good insights into the solution.

Observation 2. *The optimal accuracy $a_k^*(t_k)$ is a non-decreasing function with the allocated time t_k and it is not always smooth.*

The optimal accuracy function $a_k^*(t_k)$ is a piecewise function whose value is the maximum of each configuration accuracy given the assigned time. Intuitively, this function is non-decreasing with respect to time, as more time can only improve or maintain the same accuracy. With the increase of allocated time, when the DNN/CNN model and resolution remain constant, the accuracy function $a_k^*(t_k)$ varies smoothly with changes in the detection and tracking frames γ_k^d and γ_k^t. However, when discrete changes are made to the DNN/CNN model or resolution, the accuracy function becomes non-smooth. Then, cusp points appear in the curve.

Algorithm 2: The MC-DCRC algorithm

```
   Input: $T, \mathcal{M}, \mathcal{R}, a_k(\cdot), \delta$, initial time assignment $\mathbf{t}$
   Output: $\mathbf{C} = \{m_k, r_k, \gamma_k^d, \gamma_k^t\}, (T^c, T^g), (\vec{\tau}^c, \vec{\tau}^g)$
1  Obtain $c_k, \vec{\tau}_k^c, \vec{\tau}_k^g$ given $t_k$ by Algorithm 1;
2  Calculate the objective accuracy $a$ and slop value $\mathbf{s}$;
3  repeat
4      Select a pair of videos with highest and lowest slope $(v_{k1}, v_{k2})$;
5      $\bar{t}_{k1} \leftarrow t_{k1} + \delta, \bar{t}_{k2} \leftarrow t_{k2} - \delta$;
6      $\bar{\mathbf{t}} \leftarrow \{t_1, ..., \bar{t}_{k1}, ..., \bar{t}_{k2}, ..., t_K\}$;
7      Obtain new accuracy $\bar{a}$ and results given $\bar{\mathbf{t}}$ by Algorithm 1;
8      if $\bar{a} < a$ then
9          $\delta = \alpha\delta$;
10     else
11         Update the result and keep the new values;
12 until The maximum iterations have been reached or the step length exceeds the limitation;
13 $T^g \leftarrow \sum D_{gpu}(\gamma_k^d, r_k, m_k)$;
14 return $\mathbf{C}, (1 - T^g, T^g), (\vec{\tau}^c, \vec{\tau}^g)$
```

Observation 3. *When the allocated time is short, the accuracy function $a_k^*(t_k)$ increases rapidly with time. When the allocated time is long, the accuracy function $a_k^*(t_k)$ improvement becomes slow and reaches the plateau frequently.*

When the allocated time is limited, only a portion of frames can be processed, resulting in a rapid improvement in accuracy as more frames are processed. However, when the time becomes long, almost all frames are processed. To continue enhancing accuracy, it becomes necessary to allocate additional time towards selecting more complex DNN/CNN models and/or higher resolutions. However, these adjustments lead to a slower rate of improvement, and frequent plateaus. Therefore, we should assign more time to those who have fast accuracy increment tendency while decreasing time for those who have a slow accuracy increase trend. However, it is hard to tell the time allocation threshold for different users directly. We use the slope $s_k = \frac{a_k^*(t_k+\delta) - a_k^*(t_k)}{\delta}$ to measure the sensitiveness to time. The value of δ influences the measurement of the slope. When the δ is extremely small, the value of the slope is 0 mostly because the small change of the allocated time is not able to change the optimal video analytics configuration. When the δ is extremely large, we are not able to estimate the tendency around the allocated time. Therefore, we should select a suitable value for it. Besides, we should increase the value of δ when the allocated time reached the cusp point to overcome the plateau.

Based on the above observations and analysis, we propose the MC-DCRC algorithm as the Alg. 2 shows. The key idea is to assign more time to the videos with a high improvement rate while allocating less time to those who have a low accuracy increase rate. In the Alg. 2, given the initial solution, we obtain the optimal video analytics configuration and resource allocation by the Alg. 1 and calculate the objective accuracy and slop value of the current solution (lines 1-2). Then, we select a pair of video clips with the highest and lowest slop and re-assign time between them (lines 3-7). To mitigate the risk of getting stuck in local optima, we adaptively adjust the step length. Specifically, if the algorithm fails to improve, we increase the step length by the adjustment factor α (lines 8-9). Otherwise, the results and solution are updated (lines 10-11). This adaptive adjustment expands the search space, allowing the algorithm to escape suboptimal solutions and continue converging toward an improved allocation. The adjustment parameter $\alpha > 1$ controls the step length growth rate, ensuring a gradual yet effective adaptation. The iterative process terminates when either the maximum number of iterations is reached or the step length exceeds a predefined threshold (line 12), preventing unnecessary computational overhead.

Parameter selection is crucial for the MC-DCRC algorithm under varying video analytics scenarios. When videos are highly diverse, a smaller step length δ and adjustment factor α are preferable to avoid sucking in local optimal. Conversely, when videos are more similar, larger values of δ and α can be employed without significant accuracy decrease. Moreover, parameter selection is influenced by the delay requirement. Smaller δ and α values result in higher accuracy but slower convergence, making them suitable for applications with longer processing times. In contrast, larger parameters enable faster convergence, making them preferable for latency-sensitive tasks that require real-time processing. The detailed selection of α and δ will be discussed in Section IV.

IV. Evaluations

A. Experiment Setup

Dataset and DNN/CNN Models. The input videos come from the VIRAT [16] and Auburn dataset [17]. The original resolution of the video is $1080p$ and we offer six more candidate resolutions, i.e., $720p$, $480p$, $360p$, $240p$ $180p$, $90p$. The video frame rate can be set from 2 to 30 fps. We assume that each epoch is one second long, and hence, the frames to process for each video clip vary from 2 to 30 as well. There are three different DNN/CNN models, i.e., Yolov3, Yolov2, and Yolov2-tiny. They have different accuracy performance and processing time.

Evaluation Metrics. We use the vehicle detection task to evaluate the proposed algorithms. We measure analytics accuracy using the metric of the intersection over union (IOU) [18], similar to Deepdecison [6]. The IOU of a single object is the proportion of the intersection area to the union area of the bounding boxes. The frame IOU is the average IOU of multiple objects in this frame, and the video IOU is the average IOU of multiple frames. The ground truth is given by the Yolov5 model executed on $1080p$ raw videos.

Accuracy function profile. As we mentioned before, the accuracy function differs from video clip to video clip. However, it is impractical to profile an accuracy function for each individual video clip due to the extensive time required. To facilitate lightweight profiling, we pre-define several accuracy functions based on the size and moving speed of targets. Then, we select suitable accuracy functions for coming video clips by measuring the size and moving speed of targets in the first two frames. Though pre-defined profiling is not as accurate as individual profiling, it is more time-efficient and feasible for real-time applications.

Processing time profile. The processing time of resizing, tracking, and model inference usually remains stable, and hence, we simply regard the processing time in the current epoch as the same as the time in the previous epoch. Besides, the running time of the SC-DCRC algorithm is less than 1 ms, and the running time of the MC-DCRC algorithm can be controlled by the iteration times. The switching time between CPU-image and GPU-image is around 1 ms. They are negligible compared to the epoch duration, i.e., 1 s.

Baselines. We compare the SC-DCRC algorithm with one state-of-the-art GPU-dominant video analytics framework, one system with fixed CPU-GPU resources, one flexible CPU-GPU system, and the offline optimal algorithm.

- **Deepdecision (DD) [6].** It is a GPU-dominant optimization framework. It selects the resolution and DNN/CNN model for each video clip.
- **MARLIN [3].** It is a video analytics pipeline with fixed CPU-GPU resources. We modify the original MARLIN by replacing the change detector module with a tracking frame proportion selection. The CPU and GPU resources are set to a fixed 50%-50% split.
- **Gemini [5].** It is a video analytics pipeline with flexible CPU-GPU resources. It has multiple versions for different video applications. We select the one with the frame filtering module in CPU-image for comparison.
- **Offline optimal (OPT).** This approach is assumed to have foreknowledge of processing outcomes. It simulates running all possible video analytics configurations to identify the one yielding the highest accuracy, serving as a benchmark for the best achievable performance.

We compare the MC-DCRC algorithm with other time allocation schemes.

- **Random time allocation.** The processing time for each video clip is assigned randomly, which is the initial state of the MC-DCRC algorithm.
- **Uniform time allocation.** An equal amount of processing time is allocated to each video clip, disregarding their individual contents and frame rates.
- **Proportional time allocation.** The processing time is allocated in proportion to the frame rate, aiming to align resource distribution with the total workload.

B. The Impact of Video Analytics Configurations on Accuracy

Impact of resolution and DNN/CNN model. In Fig. 3, we plot detection accuracy versus resolution under various DNN/CNN models. To demonstrate the generality of the impact, we include additional DNN/CNN architectures beyond the YOLO-based structures. It is an increasing function with the resolution no matter which DNN/CNN model is selected, as Fig. 3(a) shows. However, the impact of the resolution differs from the DNN/CNN model to the DNN/CNN model. The complex DNN/CNN models, such as Yolov3, DETR [19],

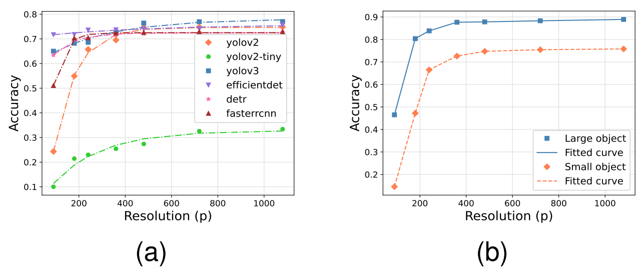


Fig. 3. Impact of resolution. (a) Accuracy VS. Models. (b) Accuracy VS. Video contents.

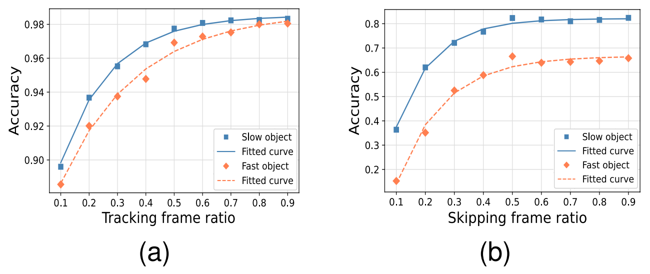


Fig. 4. Impact of detection frames. (a) Accuracy VS. Tracking proportion. (b) Accuracy VS. Detection proportion.

TABLE III
IMPACT OF THE FIXED CPU-GPU RESOURCE IN OPTIMAL CONFIGURATION SELECTION.

CPU-GPU	10fps Video	20fps Video	30fps video
10% - 90%	81.06%	48.60%	32.18%
30% - 70%	**81.90%**	**73.74%**	58.00%
50% - 50%	78.65%	72.62%	**68.35%**
70% - 30%	73.96%	69.28%	67.16%

and EffcientDet [20], are not as sensitive to changes in resolution as the simple DNN/CNN models, such as Yolov2-tiny, Yolov2, and Faster R-CNN [21]. According to the previous work [8], we have fitted the accuracy function with resolution as a concave function. For example, the blue line can be fitted as $0.782 - 0.189e^{-\frac{r}{0.261}}$. On the other hand, the video content also has an influence on the accuracy function. We show the results in Fig. 3(b) where the blue line represents a video with a large object and the orange line indicates a video with a small object. Even with the same resolution and DNN/CNN model, the large object gains higher accuracy than the small object.

Impact of sampling. We conduct experiments to measure the impact of sampling, as Fig. 4 shows. We explore the relationship between the accuracy and the tracking frames in Fig. 4(a) given the fixed skipping frames. In Fig. 4(b), we investigate the impact of detection frames given the fixed tracking frames. Both relationships can be modeled as a concave function. Meanwhile, video content has an impact on accuracy. For instance, slower objects maintain higher accuracy than faster ones, as the blue and orange lines show.

Impact of CPU-GPU resource control. To measure the impact of CPU-GPU resource control, we select optimal video analytics configurations for three video clips with different frame rates, i.e., 10 fps, 20 fps, and 30 fps, given different CPU-GPU resource divisions. Table III shows that lower frame rates generally lead to higher accuracy because the edge server has more processing time per frame. However, the accuracy improvements differ in situations. For example, with 10% CPU and 90% GPU, the accuracy in 10 fps is around 50% higher than the accuracy in 30 fps video clips. But with 70% CPU and 30% GPU, the accuracy in 10 fps is only around 5% higher than the 30 fps video clips.

This variation arises due to distinct characteristics of the CPU and GPU. The CPU can process workloads quickly but with low accuracy, while the GPU has a longer processing time but achieves higher analytics accuracy. In the 10%-90% CPU-GPU partition, the GPU has the most computing resources but it cannot handle excessive workloads. At lower frame rates, the total workload is low, allowing the GPU to handle most of the processing, resulting in higher accuracy. Conversely, at higher frame rates, the GPU cannot manage increased workloads with the same DNN/CNN model. Hence, it switches to a simpler DNN/CNN model, leading to lower accuracy. In 70%-30% partition, too much CPU resources decreases accuracy at lower frame rates, but it shares most workloads for GPU at higher frame rates, minimizing the accuracy drop.

Besides, the optimal CPU-GPU resource allocation varies in different situations. There does not exist a fixed CPU-GPU resources setting that always outperforms others. For example, the 30% CPU and 70% GPU setting has the highest accuracy in 10 and 20 fps video clips, while 50% CPU and 50% GPU has the highest accuracy in 30 fps video clips.

C. The Performance of the SC-DCRC Algorithm

Maximum number of configurations supported by SC-DCRC. Since the computation complexity of the SC-DCRC algorithm is $O(|\mathcal{M}| \times |\mathcal{R}| \times n)$, its execution time scales linearly with the number of configurations, as shown in Fig. 5. The maximum number of configurations that SC-DCRC can process depends on the allocated execution time within each epoch. For instance, if 0.5% of an epoch (i.e., 0.005 s) is allocated to SC-DCRC, the algorithm can support approximately 5,000 configurations while maintaining real-time processing. These results confirm that SC-DCRC remains computationally feasible in real-time scenarios.

The performance of SC-DCRC under different frame rates. In Fig. 6(a), we compare the performance of algorithms across different frame rates but similar video contents. Higher frame rates lead to lower accuracy due to increased computational workloads. The GPU-dominant method (DD) is affected by the increased workload most, with accuracy dropping from 0.6 at 10 fps to 0.2 at 20 fps, and further to 0.1 at 30 fps. The fixed resource system (MARLIN) performs worse than DD at 10 fps but better at 20 and 30 fps, because the tracking module in the CPU-image shares workload with the GPU-image. Gemini struggles to adjust workload partitioning, leading to similar performance across different frame rates, despite increasing computational demands. SC-DCRC dynamically adjusts CPU-GPU workloads, significantly improving accuracy. It consistently outperforms DD, MARLIN, and Gemini across all frame rates. Even compared to the OPT method, SC-DCRC achieves comparable accuracy, with a gap of less than 8%, mainly due to a minor difference between profiled and actual accuracy.

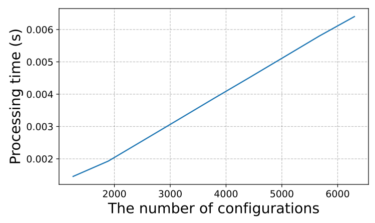


Fig. 5. Processing time of SC-DCRC with increasing number of configurations.

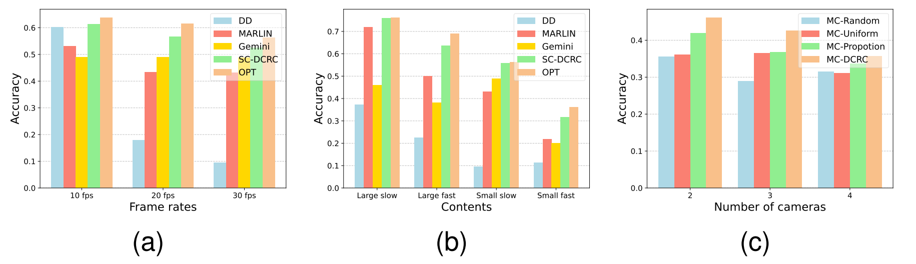


Fig. 6. Performance in single-camera and multi-camera system. (a) Single-camera frame rates. (b) Single-camera video contents. (c) Multi-camera user numbers.

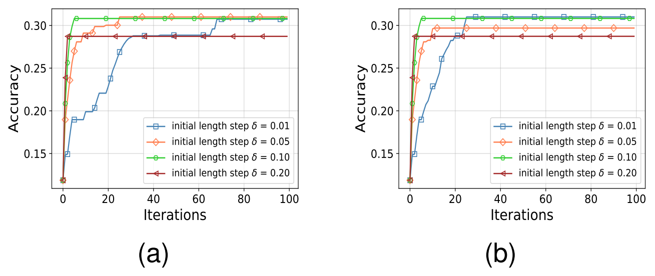


Fig. 7. Convergence with initial step length δ. (a) Adjustment factor $\alpha = 1.05$. (b) Adjustment factor $\alpha = 1.5$.

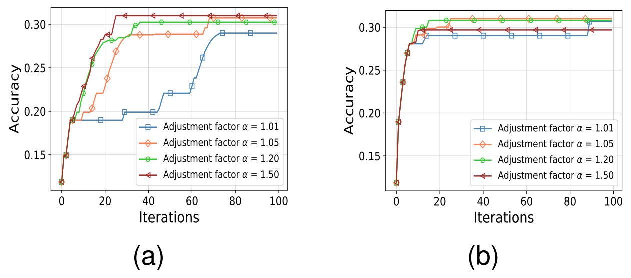


Fig. 8. Convergence with adjustment factor α. (a) Initial step length $\delta = 0.1$. (b) Initial step length $\delta = 0.2$.

TABLE IV
VIDEO FEATURES OF DIFFERENT CAMERAS

Camera No.	Object size	Object speed	Frame rate
1	Small	Fast	10
2	Large	Fast	30
3	Small	Slow	10
4	Large	Slow	20

The performance of SC-DCRC under different video contents. The performance varies among different video contents, as Fig. 6(b) shows. For large, slow-moving objects, all approaches have better performance than with other video contents except for Gemini. This improvement occurs because large objects are easier for DNN/CNN models to detect, and the slower movement facilitates more accurate tracking. However, Gemini performs better with small, slow-moving objects because Gemini tends to feed more frames to the GPU-image for large, slow objects compared to small, slow objects, since the frame differences among large moving objects exceed the filtering threshold more frequently. It causes an imbalance in CPU-GPU workloads, leading to lower accuracy. All approaches exhibit the worst performance in videos with small fast objects, except for DD. This is because DD relies solely on DNN/CNN models, which are less impacted by object speed than methods that incorporate object tracking modules. In videos with slow objects, SC-DCRC achieves accuracy comparable to OPT because the accuracy function profile is more precise in these situations.

D. The Performance of Multi-Camera Algorithm

Performance. Fig. 6(c) shows the accuracy of different time allocation methods across different numbers of cameras in Gemini+. The video features of these cameras are listed in Table IV. The average accuracy decreases as the number of cameras increases due to increasing computation workloads. Among these methods, MC-DCRC always achieves the highest accuracy. Random time allocation exhibits instability owing to its inherent randomness. MC-DCRC can improve the accuracy by modifying the random time allocation. Uniform time allocation performs poorly when there is significant diversity among video clips. For example, camera 1 and camera 2 have varying video contents and frame rates. Consequently, uniform time allocation exhibits worse accuracy with two cameras compared to three. Proportional time allocation outperforms both random and uniform time allocations because it accounts for the differing computational workloads across cameras. Nevertheless, the optimal time allocation is not only decided by the computation workloads but also by video contents. Hence, it still has worse performance than MC-DCRC.

Convergence. The convergence of the MC-DCRC algorithm depends on two parameters, i.e., the initial step length δ and the adjustment factor α. In Fig. 7, we compare the performance of different initial step lengths δ with two adjustment factors, i.e., $\alpha_l = 1.05$ and $\alpha_r = 1.5$. Generally, a larger initial length leads to a faster convergence. When $\delta = 0.01$ and $\alpha = 1.05$, the algorithm converges within around 60 times. But when $\delta = 0.2$, the number of iterations to converge decreases to only 3. However, blindly increasing δ would lead to a local optimal and hence achieve low accuracy. In Fig. 8, we measure the influence of different adjustment factors α with two initial step lengths $\delta_l = 0.1$ and $\delta_r = 0.2$. Similarly, a larger adjustment factor leads to a quick convergence. For example, in Fig. 8(a), it takes around 70 times to reach convergence when $\alpha = 1.01$ while it halves the iteration times when $\alpha = 1.5$. However, a larger adjustment factor might get stuck in the inferior solutions, as Fig. 8(b) shows. In our experiment, when $\delta = 0.1$ and $\alpha = 1.05$, it achieves a good trade-off between the convergence speed and performance. We adjust these parameters according to empirical observations.

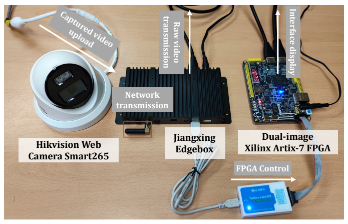


Fig. 9. The system prototype of Gemini+.

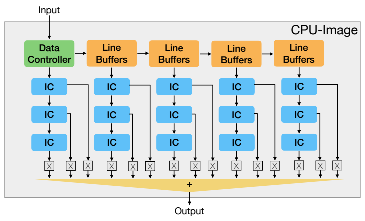


Fig. 10. The CPU-image implementation.

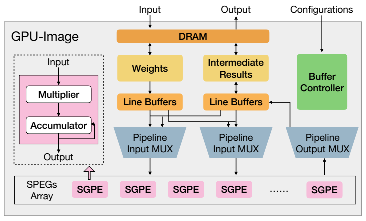


Fig. 11. The GPU-image implementation.

TABLE V
ABLATION STUDY: ACCURACY EVALUATION OF SC-DCRC WITH AND WITHOUT ACCURACY/DELAY MODELS

	Large Slow	Large Fast	Small Slow	Small Fast
SC-DCRC	0.76	0.64	0.56	0.32
SC-DCRC-AVG	0.61	0.64	0.54	0.20
SC-DCRC-LAST	0.74	0.64	0.54	0.20

E. Ablation Studies

We conduct ablation studies to show the effectiveness of accuracy and delay models. We compare the SC-DCRC algorithm with two heuristic baselines: 1). **SC-DCRC-AVG**, which replaces the models with the pre-computed average accuracy and delay values, and 2). **SC-DCRC-LAST**, which uses accuracy and delay from the last epoch, assuming that the video characteristics remain stable.

Table V presents a comparative analysis of SC-DCRC and its variants under different video scenarios. Among all methods, SC-DCRC consistently achieves the highest accuracy across all conditions, demonstrating the effectiveness of accuracy and delay models. On average, it improves accuracy by 14% over SC-DCRC-AVG and 7% over SC-DCRC-LAST. The performance gap arises due to the ability of SC-DCRC to dynamically adapt to varying video contents by leveraging real-time accuracy and delay estimations. In contrast, SC-DCRC-AVG assumes a static average accuracy and delay, leading to poor adaptability. The accuracy drops particularly when the video used for profiling differs significantly from the actual videos for analytics. SC-DCRC-LAST partially mitigates this issue by leveraging past analytics results. However, it struggles when video content fluctuates rapidly, as it lacks the capability to handle dramatic changes.

Since the accuracy and delay models are profiled offline, they introduce minimal computational overhead for the SC-DCRC algorithm. The main overhead comes from the content analyzer module, which is responsible for feature extraction. These extracted features are then used for accuracy model selection, ensuring that the most suitable model is applied based on video content. To minimize computational costs, we adopt lightweight algorithms, such as frame difference, which provide a balance between accuracy and processing speed.

V. PROTOTYPE AND CASE STUDY

A. Prototype Implementation

We implement the Gemini+ prototype by reorganizing the hardware designs of Gemini. The prototype consists of a Hikvision web camera smart265 [22], a Jiangxing edgebox [23], and a dual-image Xillinx Artix-7 [24] FPGA, as Fig. 9 shows. We replace the Intel Max 10 FPGA in Gemini with Xilinx Artix-7 because it provides more processing units to support advanced DNN/CNN models. The camera captures video and uploads it wirelessly to the edgebox for further analysis. The edgebox is responsible for video analytics optimization and FPGA control. Specifically, it receives frames from cameras to perform content analysis. Based on the content information, it generates video analytics optimization policies and FPGA control signals, guiding the FPGA to execute detailed video analytics tasks. After analysis, the FPGA sends the results back to the edgebox.[2]

The edgebox is equipped with a 6-core ARM Cortex-A53 processor, 4GB memory, and runs on Ubuntu 18.04. It receives frames from the camera via wireless transmission and controls the FPGA through a wired connection. The edgebox transmits raw video data to the FPGA via HDMI for analytics, and the inference results are subsequently transmitted back to the edgebox through the same HDMI connection, as the current prototype does not include an additional transmission module in the FPGA. The Xilinx Artix-7 FPGA has 101,440 configurable logic cells and 4GB DDR3 memory. The FPGA operates at a maximum clock frequency of 464 MHz. These logic cells are configured into different processing units for the CPU-image and GPU-image, and the DDR3 memory facilitates data storage and exchange. Specifically, we implement instruction cores (ICs) as the basic processing unit for the CPU-image, as shown in Fig. 10. These ICs calculate the feature values in pixels for bilinear interpolation resizing and optical flow object tracking. To optimize execution efficiency, we adopt a buffer-instruction-core architecture, where data is preloaded into buffers via the Data Controller. This design ensures that ICs operate continuously without idling, significantly reducing total execution time. For the GPU-image, we design a simple and generic processing element (SGPE) which consists of a multiplier and an accumulator, as illustrated in Fig. 11. These SGPEs can be combined together to support different types of NN computation. During inference, the buffer controller regulates the NN's structure, while the weight buffers store the model parameters. The previous intermediate results are fetched and processed by the SGPEs, and the newly computed results are sent back to the line buffers. This data exchange is efficiently managed by pipeline multiplexers. The input and output results are stored in DDR3 memory and periodically exchanged with the intermediate results buffers. This design enables flexible model inference by loading different model structures and weights in the buffer. In this prototype, we provide Yolov3, Yolov2, and Yolov2-tiny [25].

[2]The FPGA design and implementation details can be found in Gemini [5].

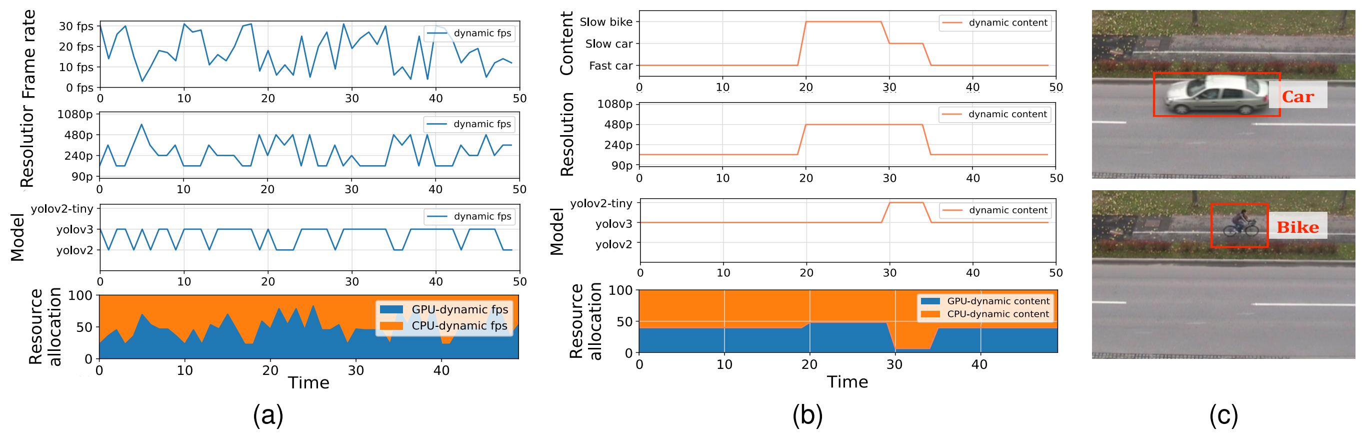


Fig. 12. Runtime video analytics configuration adaptation and resource division of Gemini+. (a) Dynamic frame rate. (b) Dynamic content. (c) Intelligent traffic monitoring task.

TABLE VI
ENERGY CONSUMPTION UNDER DIFFERENT FPGA CONFIGURATIONS

Configuration	Power (W)	Throughput (FPS)	Energy/Frame (J)
Gemini+	4.65	30	0.155
CPU-image	3.61	53	0.0681
GPU-image	6.54	14	0.467

B. A Case Study

We verify the adaptability of resource control and video analytics configuration selection under the task of intelligent traffic monitoring. The Gemini+ system aims to detect and track various traffic participants, such as vehicles, bicycles, and pedestrians, as Fig. 12(c) shows. We consider two cases: dynamic frame rates and varying video contents.

In the dynamic frame rate scenario, the smart camera adjusts its frame rate according to the available bandwidth. The edge server adapts the CPU-GPU resource allocation and selects the most suitable resolution and DNN/CNN model in response to the changing frame rates, as shown in Fig. 12(a). When the frame rate is low, the system tends to allocate more resources to the GPU since its workload is relatively light compared to high frame rate scenarios.

In the dynamic content scenario, video content changes over time at a fixed frame rate of 60. There are high-speed cars in epochs 0-19, slow bikes in epochs 20-29, followed by slow cars in epochs 30-34, and fast cars in epochs 35-50. In response to these changes, the system adjusts its configuration, as shown in Fig. 12(b). The initial video analytics configuration is Yolov3 at $180p$ for fast cars. It then switches to Yolov3 at $480p$ to better handle slow bikes. This switch is because the small contents require a larger resolution for model inference. A simpler DNN/CNN model, Yolov2-tiny, is selected for the slow cars because it achieves similar accuracy for the large content compared to a more complex DNN/CNN model. Moreover, since the object tracking module in the CPU-image performs better with slower-moving targets, the allocation of GPU resources is reduced significantly.

Moreover, we measure the energy consumption of the dual-image FPGA with Monsoon Power Monitor [26] in this case study. Specifically, we first measure the energy consumption of the overall system (Gemini+). Then we evaluate the energy consumption of the CPU-image and GPU-image respectively, where we only run a single image for video analytics. Table. VI shows the power, throughput, and energy per frame for these three FPGA configurations. The CPU-image achieves the highest energy efficiency, consuming only 0.0681 J per frame. However, it can not run for a long time due to a quick accuracy drop. GPU-image incurs the highest energy cost, requiring 0.467 J per frame, due to its deep learning workload. Gemini+ balances energy and performance, achieving 30 FPS at 0.155 J per frame, 66.8% lower than GPU-image, demonstrating its ability to optimize resource allocation dynamically.

Additionally, we measure the image switching power as 1.65 W. While the instantaneous power during switching is high, the total energy overhead is only 0.104 J due to the extremely short switching time. This results in an overall switching overhead of around 2% of the total energy consumption of Gemini+, confirming that the energy impact of dynamic switching remains negligible in practical deployment. Since FPGAs are well-known for their energy efficiency, we focus on their own energy consumption instead of comparison against other hardware such as CPU and GPU [27]–[29].

VI. RELATED WORK AND DISCUSSION

A. Related Work

Video Analytics with Hardware Accelerators. While CPU and GPU provide high computational power, their fixed architectures and general-purpose design make them less efficient for real-time video analytics. In contrast, hardware accelerators, including dedicated chips and FPGAs, provide hardware-level acceleration for video analytics tasks [30]. Dedicated chips such as TPU [31] and NPU have lower energy consumption and faster processing time than GPU. These dedicated chips are typically implemented using application-specific integrated circuits (ASICs). While they provide high efficiency for deep learning inference, they do not efficiently handle instruction-intensive tasks, which are crucial for video analytics. In contrast, FPGAs offer reconfigurability, enabling task-specific acceleration within the same system. Traditional feature extraction methods are accelerated with FPGAs [32].

Deepburning [33] proposes a design automation tool to build different NN models. Unlike the traditional FPGAs which take minutes for reconfiguration, the dual-image FPGAs allow seamless switching between two pre-configured images. Such flexibility makes them particularly suitable for video analytics which involves dynamic instruction-intensive and data-intensive workloads. Gemini [5] leverages dual-image FPGAs to propose a real-time video analytics system with flexible computing resources. Our work builds on these advancements by utilizing such dual-image FPGAs to accelerate video analytics, incorporating novel CPU-GPU resource control and video analytics configuration adaptation.

Tracking-based Video Analytics. Due to the high processing latency of the DNN/CNN model and the temporal correlation of the video frames, the tracking method is a good way to share the heavy workload with the DNN/CNN model. MARLIN [3] proposes a sequential video analytics pipeline with an object detector and tracker while AdaVP [4] implements a parallel detection and tracking pipeline. An online change detector threshold selection algorithm is studied in [2]. It adjusts the frame filtering threshold to adapt to dynamic video content. Except for building a direct tracking-based video analytics system, authors in [34] design a video action recognition accelerator by utilizing temporal optical flow feature among frames. Our work considers the video analytics configuration selection and CPU-GPU resource control in the tracking-based video analytics system.

Computing Resource Allocation for Video Analytics. Efficient computing resource allocation is essential for balancing accuracy and latency in video analytics. Several approaches leverage edge and cloud computing to optimize performance. Neurosurgeon [35], DNN Surgery [36], and JALAB [37] partition DNN models between devices and servers, while FilterForward [38] accelerates video analytics by running lightweight NN models at the edge and reusing intermediate results. Other works focus on heterogeneous resource scheduling. DART [39] ensures deterministic response times for real-time DNN tasks, while LaLaRand [40] optimizes CPU-GPU scheduling for DNN inference by leveraging CPU-friendly quantization. Additionally, the work [41] proposes a CPU-GPU-utilization-aware scheduling strategy to improve energy efficiency in heterogeneous systems. Unlike these methods, our work focuses on flexible CPU-GPU resource allocation enabled by dual-image FPGAs, allowing dynamic adaptation to changing workloads in real-time video analytics.

B. Discussion

Image Switching Time Reduction. With the increasing video streams, it is critical to reduce image switching overhead. One potential direction is to leverage partial reconfiguration (PR). The PR reconfigures part of the FPGA instead of the whole by sharing common processing units between the CPU-image and GPU-image. Hence, the overhead of a single switch can be decreased. Additionally, batch processing can further mitigate frequent switches by grouping and executing similar tasks together. This approach can optimize execution efficiency and reduce the overall switching overhead.

Adaptive Parameter Selection for the MC-DCRC Algorithm. While our current approach empirically evaluates the impact of step lengths and adjustment factors on algorithm performance, these parameters are manually selected based on experimental insights. An automated adaptive parameter selection strategy will ensure the dual-image FPGA achieves high performance across diverse deployment scenarios. One promising solution is to leverage reinforcement learning for parameter tuning by learning from past parameters and algorithm performance.

Dual-Image FPGAs for Multi-Camera Multi-Object Tracking. Though our current approach considers the multi-camera scenario, each video stream is processed independently. Multi-camera multi-object tracking introduces additional challenges, such as spatiotemporal object association and cross-camera identity matching. This necessitates a novel dual-image FPGA design and new algorithms that efficiently allocate computing resources across multiple cameras.

VII. Conclusion

In this work, we propose Gemini+, a video analytics pipeline with flexible CPU-GPU resources based on dual-image FPGAs. By optimizing CPU-GPU time allocation and video analytics configurations, our framework enhances performance across both single-camera and multi-camera scenarios. Efficient algorithms are proposed for both single-camera and multi-camera scenarios. Our evaluations demonstrate that flexible CPU-GPU resources significantly improve detection accuracy compared to fixed CPU-GPU resources and GPU-dominant systems. Gemini+ also makes better use of dual computing resources compared to our previous work, Gemini. Additionally, we build a prototype to verify the benefits of flexible CPU-GPU computing resources.

References

[1] Aws deeplens deep learning enabled video camera for developers. [Online]. Available: https://aws.amazon.com/cn/deeplens/

[2] M. Hanyao, Y. Jin, Z. Qian, S. Zhang, and S. Lu, "Edge-assisted online on-device object detection for real-time video analytics," in *Proc. IEEE INFOCOM*, 2021, pp. 1–10.

[3] K. Apicharttrisorn, X. Ran, J. Chen, S. V. Krishnamurthy, and A. K. Roy-Chowdhury, "Frugal following: power thrifty object detection and tracking for mobile augmented reality," in *Proc. ACM SenSys*, 2019, pp. 96–109.

[4] M. Liu, X. Ding, and W. Du, "Continuous, real-time object detection on mobile devices without offloading," in *Proc. IEEE ICDCS*, 2020, pp. 976–986.

[5] C. Hu, R. Lu, Q. Sang, H. Liang, D. Wang, D. Cheng, J. Zhang, Q. Li, and J. Peng, "An edge-side real-time video analytics system with dual computing resource control," *IEEE Transactions on Computers*, vol. 72, no. 12, pp. 3399–3415, Dec 2023.

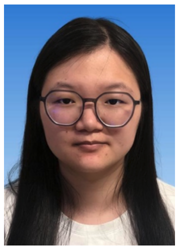
[6] R. Xukan, C. Haolianz, X. Zhu, L. Zhenming, and C. Jiasi, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE INFOCOM*, 2018, pp. 1421–1429.

[7] T. Tan and G. Cao, "Deep learning video analytics through edge computing and neural processing units on mobile devices," *IEEE Transactions on Mobile Computing*, vol. 22, no. 3, pp. 1433–1448, Mar 2023.

[8] 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. IEEE INFOCOM*, 2020, pp. 257–266.

[9] Y. He, P. Yang, T. Qin, and N. Zhang, "End-edge coordinated joint encoding and neural enhancement for low-light video analytics," in *Proc. IEEE GLOBECOM*, 2023, pp. 7363–7368.

[10] K. Jain, K. S. Adapa, K. Grover, R. K. Sarvadevabhatla, and S. Purini, "A cloud-fog architecture for video analytics on large scale camera networks using semantic scene analysis," in *Proc. IEEE/ACM CCGrid*, 2023, pp. 513–523.
[11] S. Wang, C. Zhang, Y. Shu, and Y. Liu, "Live video analytics with fpga-based smart cameras," in *Proc. HotEdgeVideo@MobiCom*, 2019, pp. 9–14.
[12] A. Muscoloni and S. Mattoccia, "Real-time tracking with an embedded 3d camera with fpga processing," in *Proc. IEEE IC3D*, 2014, pp. 1–7.
[13] R. Li, D. Su, and H. Kan, "A fast, scalable, and energy-efficient edge acceleration architecture based on fpga cluster," in *Proc. ACM CoNEXT*, 2021, pp. 479–480.
[14] Y. Wang, H. Kan, D. Su, Y. Shen, W. Liu, and M. Ou, "Energy efficient computing offloading mechanism based on fpga cluster for edge cloud," in *Proc. ACM EITCE*, 2020, pp. 1120–1125.
[15] H. Suzuki, "A generalized knapsack problem with variable coefficients," *Mathematical Programming*, vol. 15, pp. 162–176, Dec 1978.
[16] S. Oh, A. Hoogs, A. Perera, N. P. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, "A large-scale benchmark dataset for event recognition in surveillance video," in *Proc. IEEE CVPR*, 2011, pp. 3153–3160.
[17] City of auburn toomer's corner webcam 2. [Online]. Available: https://www.youtube.com/watch?v=hMYIc5ZPJL4
[18] M. Dantone, L. Bossard, T. Quack, and L. Van Gool, "Augmented faces," in *Proc. IEEE ICCV workshops*, 2011, pp. 24–31.
[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. ECCV*, 2020, p. 213–229.
[20] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proc. IEEE CVPR*, 2020, pp. 10 781–10 790.
[21] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, June 2017.
[22] Hikvision. [Online]. Available: https://www.hikvision.com/cn/
[23] Jiangxing intelligence inc. [Online]. Available: https://www.jiangxingai.com
[24] Amd artix 7 fpgas. [Online]. Available: https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga/artix-7.html
[25] R. Joseph, D. Santosh, G. Ross, and F. Ali, "You only look once: Unified, real-time object detection," in *Proc. IEEE CVPR*, 2016, pp. 779–788.
[26] High voltage power monitor. [Online]. Available: https://www.msoon.com/online-store/High-Voltage-Power-Monitor-p90002590
[27] M. Qasaimeh, K. Denolf, J. Lo, K. Vissers, J. Zambreno, and P. H. Jones, "Comparing energy efficiency of cpu, gpu and fpga implementations for vision kernels," in *Proc. IEEE ICESS*, 2019, pp. 1–8.
[28] S. Wang, C. Zhang, Y. Shu, and Y. Liu, "Live video analytics with fpga-based smart cameras," in *Proc. HotEdgeVideo*, 2019, pp. 9–14.
[29] R. Al Amin, M. Hasan, V. Wiese, and R. Obermaisser, "Fpga-based real-time object detection and classification system using yolo for edge computing," *IEEE Access*, May 2024.
[30] R. Xu, S. Razavi, and R. Zheng, "Edge video analytics: A survey on applications, systems and enabling techniques," *IEEE Communications Surveys Tutorials*, vol. 25, no. 4, pp. 2951–2982, Oct 2023.
[31] Y. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking tpu, gpu, and cpu platforms for deep learning," 2019, *arxiv:1907.10701*.
[32] G. V. D. Wal, D. C. Zhang, I. Kandaswamy, J. Marakowitz, K. Kaighn, J. Zhang, and S. Chai, "Fpga acceleration for feature based processing applications," in *Proc. IEEE CVPRW*, 2015, pp. 42–47.
[33] Y. Wang, J. Xu, Y. Han, H. Li, and X. Li, "Deepburning: Automatic generation of fpga-based learning accelerators for the neural network family," in *Proc. ACM/EDAC/IEEE DAC*, 2016, pp. 1–6.
[34] P. Zhen, X. Yan, W. Wang, H. Wei, and H.-B. Chen, "A highly compressed accelerator with temporal optical flow feature fusion and tensorized lstm for video action recognition on terminal device," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 10, pp. 3129–3142, Jan 2023.
[35] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. ACM ASPLOS*, 2017, p. 615–629.
[36] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *Proc. IEEE INFOCOM*, 2019, pp. 1423–1431.
[37] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "Jalad: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution," in *Proc. IEEE ICPADS*, 2018, pp. 671–678.
[38] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. Andersen, M. Kaminsky, and S. R. Dulloor, "Scaling video analytics on constrained edge nodes," in *Proc. MLSys*, 2019.
[39] Y. Xiang and H. Kim, "Pipelined data-parallel cpu/gpu scheduling for multi-dnn real-time inference," in *Proc. IEEE RTSS*, 2019, pp. 392–405.
[40] W. Kang, K. Lee, J. Lee, I. Shin, and H. S. Chwa, "Lalarand: Flexible layer-by-layer cpu/gpu scheduling for real-time dnn tasks," *Proc. IEEE RTSS*, pp. 329–341, 2021.
[41] X. Tang and Z. Fu, "Cpu–gpu utilization aware energy-efficient scheduling algorithm on heterogeneous computing systems," *IEEE Access*, vol. 8, pp. 58 948–58 958, Mar 2020.



Jingrou Wu received the B.E degree in computer science and engineering from the Southern University of Science and Technology, Shenzhen, China, in 2019. She is currently pursuing the Ph.D degree with the School of Computer Science, University of Technology, Sydney, Australia and with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. Her research interests focus on edge computing and volumetric video streaming.

Chuang Hu received his B.S and M.S. degrees in Computer Science from Wuhan University in 2013 and 2016, and Ph.D. degree from the Hong Kong Polytechnic University in 2019. He is an Associate Researcher in the School of Computer Science at Wuhan University. His research interests include edge learning, federated learning/analytics, and distributed computing.

Jin Zhang (Member, IEEE) received the B.E. and M.E. degrees from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2004 and 2006, respectively, and the Ph.D. degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. Her research interests are mainly in network economics, mobile computing in healthcare, cooperative communication, and networks.

Dan Wang (Senior Member, IEEE) received the B.Sc. degree in computer science from Peking University, Beijing, China, in 2000, M.Sc. degree in computer science from Case Western Reserve University, Cleveland, OH, USA, in 2004, and the Ph.D. degree in computer science from Simon Fraser University, Burnaby, BC, Canada, in 2007. His current research focuses on smart energy systems.

Jing Jiang received the PhD degree in information technology. She is currently an Associate Professor in the School of Computer Science and a core member of the Australian Artificial Intelligence Institute (AAII), at the University of Technology Sydney (UTS), Australia. Her research interests include data mining and machine learning applications with the focuses on deep reinforcement learning and sequential decision-making.