IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 December 2023, accepted 8 January 2024, date of publication 18 January 2024,
date of current version 25 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3355815

RESEARCH ARTICLE

Edge Computing Resource Management for Cross-Camera Video Analytics: Workload and Model Adaptation

HUAN-TING CHEN, YAO CHIANG, (Member, IEEE), AND HUNG-YU WEI, (Senior Member, IEEE)
Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
Corresponding author: Hung-Yu Wei (hywei@ntu.edu.tw)

This work was supported by the National Science and Technology Council (NSTC) of Taiwan under Grant 112-2628-E-002-025-.

ABSTRACT Multi-camera systems are now widely employed across numerous domains. The exponential growth of deep learning has simplified the implementation of advanced video analytics applications. While current systems strive to enhance live video analytics from several aspects, they overlook the potential degradation in performance resulting from dynamic content changes, such as the variation in the quantity and types of objects of interest over a period and across different cameras. The authors introduce Workload and Model Adaptation (WMA), a two-stage resource allocation strategy for a three-tiered, cross-camera video analytics system. This system not only supports model fine-tuning but also ensures workload balance. Notably, both the system architecture and control workflow fully comply with the IEEE 1935 edge standard. This paper delves into the GPU utilization performance of a vehicle re-identification application and examines the workload dynamics spanning multiple cameras. Furthermore, the challenges related to multi-process execution are explored. The system is evaluated using a commonly employed dataset and a popular open-source project. The results demonstrate that the proposed design surpasses the baseline and enhances the overall throughput and latency across cameras within the system.

INDEX TERMS Continuous learning, edge computing, re-identification, resource allocation, offloading, video analytic.

I. INTRODUCTION

In recent years, cameras have been widely deployed at many places, such as road intersections, grocery stores, and the university campus. The rapid advancement of deep learning has made it increasingly convenient to deploy powerful video analytics applications. Furthermore, the utilization of multiple cameras further expands the range of applications that can be supported, such as vehicle counting, traffic control, and object re-identification, as shown in Figure 1 [1].

Additionally, some studies leverage the spatial and temporal correlations among cameras to enhance system performance in multi-camera systems [2], [3]. While the demand for video analytic applications continues to grow, it becomes increasingly important to have powerful computing resources in order to achieve high throughput and low latency. Edge computing aims to provide cloud capabilities closer to end-users by offering computing, caching, and communication

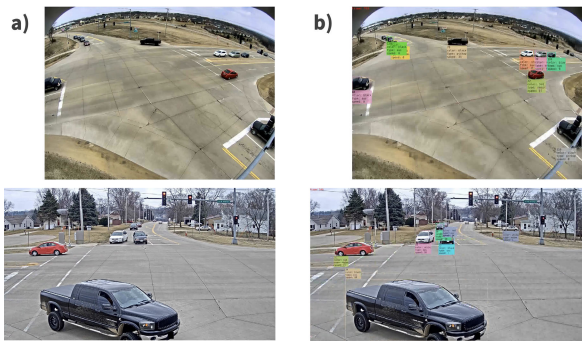

FIGURE 1. Vehicle re-identification example. a) The original video stream. b) The video stream with re-identification results displayed using bounding boxes and labels of different colors.

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Tsun Cheng.

resources at the network edge [4]. Recent studies have shown that integrating edge/fog computing architecture with video analytics applications is an effective approach to mitigate latency and throughput bottlenecks in the system. By processing the application closer to the data source, edge/fog computing reduces the need for data transmission, minimizes latency, and improves overall system performance [1], [5]. Various standards have been established to define the behavior of edge computing, including the ETSI MEC standard [6] and the IEEE 1935 Edge standard [7]. Among them, the IEEE 1935 standard defines a three-tiered architecture to leverage the management and orchestration of edge resources and applications. With the defined workflow, the paradigm guarantees the system's flexibility, stability, and security. Due to its hierarchical structure, it allows for more efficient and diverse data processing at the edge. These advantages benefit video analytic applications as the system can process video analytics dispersedly and keep resources and bandwidth efficient.

In real-world deployments, the number and types of objects of interest, such as people, vehicles, and trucks, can vary significantly both over time and across different cameras. This variation is due to factors such as changes in environmental conditions, different camera viewpoints, and dynamic scenarios. The variation in number and type presents two challenges: workload imbalance and data drift.

For an introductory video analytics system with cloud/edge architecture, workload imbalance can cause serious issues. Each camera's video processing stream is fixed on a centralized cluster or the nearest edge server. This setup results in some servers being overloaded with work while others are underutilized, leading to lower processing throughput and higher latency. Some recent studies have explored methods for dynamically distributing the processing pipeline across the cluster, which reduces latency and improves overall throughput [8], [9], [10], [11], [12], [13].

Data drift is another significant issue for video analytics on the edge. Since edge servers have limited computational resources, they can only support deep neural network (DNN) models with fewer weights. This limitation makes the model likely to lose accuracy when faced with significant data variations. A promising approach to address data drift is continuous learning. It involves incrementally retraining edge DNNs on new video samples while retaining previous knowledge. By continuously updating the models with new data every few minutes, the edge DNNs can adapt to changing environments, evolving object behaviors, and other variations over time [14], [15].

Both workload imbalance and data drift are crucial issues in multi-camera video analytics systems. Moreover, there are several challenges to address with both issues. Continuous learning requires enormous computing resources, offloading requires large network bandwidth between servers, and the time scale of data drift and workload imbalance is different.

As a result, the authors propose Workload and Model Adaptation (WMA), a cross-camera video analytic framework to tackle these challenges. The framework utilizes a three-tiered edge system, which adheres to the IEEE 1935 Edge standard. Besides, it decouples complex multi-hierarchical problems, tackles each issue independently, and optimizes the overall system performance. In the edge system, the middle-layer server aims to monitor the workload and address workload imbalance. The top-layer server is responsible for monitoring accuracy and sharing the fine-tuned model. The key contributions are as follows:

- A multi-camera vehicle tracking system, which incorporates retraining and workload balancing techniques, is proposed. The results show that the proposed algorithm can enhance the system's overall throughput across cameras.
- The utilization of GPU for real-time vehicle tracking is analyzed and leveraged in the proposed mechanism. Therefore, the system is more consistent in practical usage.
- The cross-camera vehicle tracking system is fully compatible with the IEEE 1935 edge standard, which ensures compatibility across heterogeneous devices and enables exceptional flexibility and scalability.

The rest of this paper is organized as follows. Section II delves into a discussion of work related to this research. Section III outlines the proposed system model. Following this, the problem formulation is introduced in Section IV. Section V details the evaluation setup and outlines the experiments conducted. Lastly, the research findings are concluded in Section VI.

II. RELATED WORK

This section introduces the IEEE 1935 edge standard architecture, which is the primary edge system architecture used in this study. Additionally, it also examines the multi-camera video analytics systems research and categorizes them into four domains: video analytics optimization, resource management, configuration improvement, and envisioned system.

A. IEEE 1935 EDGE STANDARD

IEEE 1935 edge standard provides better availability, flexibility, and scalability in the Edge/Fog systems. The standard defines a three-level architecture for different functions to enable the management and orchestration of the Edge/Fog system [7]. Figure 2 shows the overall architecture of the Edge/Fog system. The IEEE 1935 standard also offers a set of APIs to manage and configure the resources and applications in the Edge system [16].

The functionalities of each level are described as follow:

1) **Edge/Fog Orchestrator (EFO)**: The top layer of the edge computing system. The Edge/Fog Orchestrator is designed to manage and control the Edge system. It is responsible for interacting with users, handling

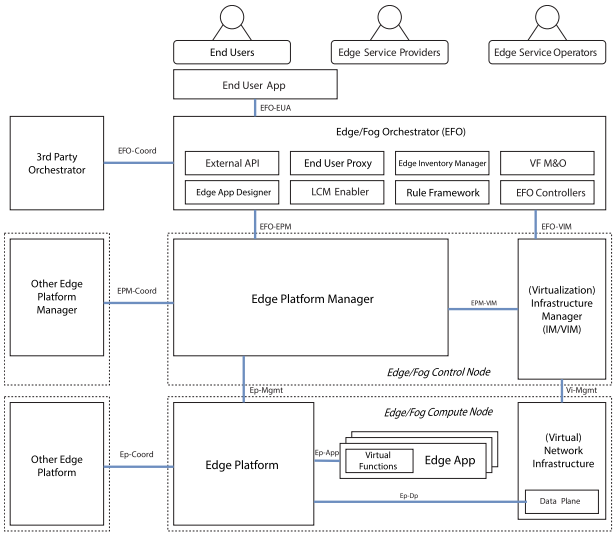

FIGURE 2. The architecture of IEEE 1935 edge standard (Adapted from the standard document.)

applications properly, and providing system functionalities and infrastructures. There are many components in the EFO, including Virtual Function Management and Orchestration (VF M&O), Rule Framework, and Edge Inventory Manager (EIM). VF M&O is responsible for the start-up and operation of the application. Rule Framework deals with all conditions, requirements, and reactions. EIM provides a real-time view of the system's various inventories, including resources and services.

2) **Edge/Fog Control Node**: The middle layer of the edge computing system. Control nodes oversee the Compute nodes and manage related resources. The entities can be divided into two categories based on their management targets: the (Virtualization) Infrastructure Manager, which focuses on resource management, and the Edge Platform Manager, which focuses on application management.
3) **Edge/Fog Compute Node**: The bottom layer of the edge computing system. Compute nodes are responsible for practical computing tasks. Edge applications interact with the Edge Platform and communicate with the internet via the Data Plane.

B. VIDEO ANALYTICS OPTIMIZATION

The video analytics application comprises a sequential series of stages, encompassing frame partitioning, region of interest (ROI) detection, and model inference. Various studies enhance the performance by optimizing individual stages in specific contextual scenarios.

Spatula [3] exploited the spatial and temporal correlations in multi-camera scenarios to reduce computing costs. Reducto [17] improved the resource consumption by optimizing the filtering mechanism in the video analytic pipeline. AMS [18] utilized a remote server for model training and updated the edge device to improve performance. A. Aliouat et al. dug into the video coding strategy and proposed a fast ROI detection method [19].

C. RESOURCE MANAGEMENT

Video analytics is a resource-hungry application. Due to limited resources, processing multiple video streams simultaneously on the server poses a challenge. Thus, it is critical to have an intelligent resource management mechanism. The techniques for better resource management include video pipeline partition and deployment, video pipeline integration, and server workload balancing. The balance mechanism for both pipeline and workload can be activated within seconds, ensuring swift adaptation to variations.

VideoEdge [8] identified the best video pipeline configuration by resources and accuracy. The system also placed the pipeline across the hierarchy of clusters and merges typical components across pipelines. VideoStorm [20] designed a scheduler that efficiently considered the resource-quality profile and lag tolerance. Distream [9] adaptively balanced the workloads across smart cameras and partitioned the workloads between cameras and the edge cluster. Faticanti et al. proposed a method to determine the placement of video pipelines on infrastructure and aimed to improve the coverage of the camera network [21]. Rong et al. scheduled massive camera streams and tasks on end-edge-cloud architecture with comprehensive consideration of computation and networking resources [11]. Zhang et al. designed a blockchain-based edge system to reduce the execution time by mapping video pipeline to edge [22].

D. CONFIGURATION IMPROVEMENT

The configuration of the application includes video resolution, hyper-parameters, and model selection. Improving the video pipeline configuration involves leveraging the correlation between cameras, such as region of interest, object density, and field of view. Another approach to enhance configuration is implementing continuous learning to adapt the model. The inference model will undergo updates every few minutes through retraining with newly arrived data.

Chameleon [23] adapted the configurations over time and utilized both spatial and temporal correlations to amortize the overhead of adaptation. CrossRoI [2] generated a better region of interest for each camera based on offline processing of multiple video streams. Convince [24] utilized spatiotemporal correlation to eliminate redundant frames and reduce bandwidth utilization. Li [25] adjusted the camera pose by jointly considering other camera poses in active object tracking applications. Ekya [14] scheduled the training and inference process to achieve higher frequency. RECL [15] shared the fine-tuned model across edges and designed the model selection algorithm.

E. ENVISIONED SYSTEM

Some systems also claim to provide potent video analytics applications but do not provide any implementations. Yi et al. [26] aimed to build a software-defined video analytics system with a holistic resource orchestrator. The proposed system exploited cross-camera collaboration and could run

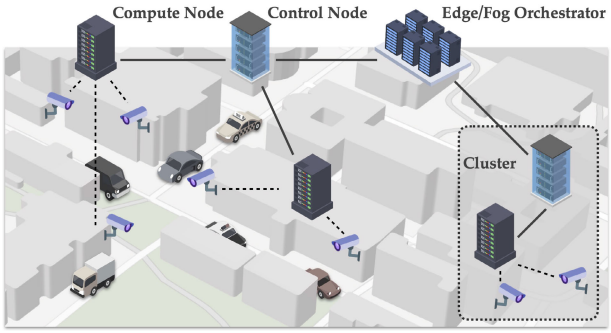

FIGURE 3. The system architecture of WMA.

various applications simultaneously. Apostolo et al. [27] proposed Live video analytics as a service to achieve flexibility, agility, and efficiency. The system included a declarative interface, an adaptive analytics engine, and an efficient runtime system.

Previous studies focused on either resource management or configuration improvement. These existing works do not build a comprehensive system for video analytic applications. This gap highlights the need for integrated frameworks that combine resource management and configuration improvement to optimize resource allocation and enhance performance.

III. SYSTEM MODEL

This paper proposed WMA, the three-tiered edge computing system aligned with the IEEE 1935 edge standard, as shown in Figure 3. The Edge/Fog Orchestrator can collect metrics from all Compute nodes. The Control node, situated at road intersections, manages these Compute nodes and monitors their workloads. Each Compute node is a worker node, running applications for nearby cameras. The Control node and the underlying Compute nodes form a **Cluster**. The dashed lines represent connections between the cameras and servers, whereas the solid lines depict connections between two servers. The cameras are deployed across the road intersections as the input of the video analytic service.

The proposed system's detailed component design and workflow consistent with IEEE 1935 standard is illustrated in Figure 4. The system's workflow involves the VF M&O, Rule Framework, and Edge Inventory Manager in EFO, Edge Platform Manager in the Control node, and Edge Platform and Edge App in the Compute node.

The brown arrows indicate the data flow of the application, while the green and blue arrows represent the control flow of the management scheme, which will be explained in more detail in the next section.

Consider that there are V cameras and N Compute nodes in the cluster. Each camera can stream the video to any Compute node in the cluster in the Compute node. The mapping from the camera to the Compute node is defined as binary variable e_{vn}. If the camera v is processing on Compute node n, e_{vn} equals to 1.

The parameters and detailed definition of overall system architecture are denoted as follows:

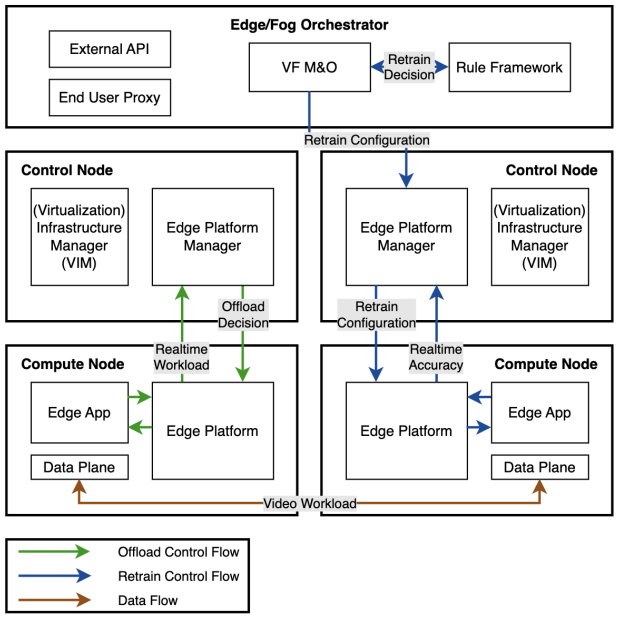

FIGURE 4. The IEEE 1935 edge standard compatible system architecture and system workflow.

- $\mathcal{V}$: The set of video streams (cameras). Defined as $\mathcal{V} = \{1, 2, \cdots, V\}$.
- $\mathcal{N}$: The set of Compute nodes in a cluster. Defined as $\mathcal{N} = \{1, 2, \cdots, N\}$.
- $\mathcal{E}$: The set of binary variable e_{vn}. Defined as $\mathcal{E} = \{e_{vn} | v \in \mathcal{V}, n \in \mathcal{N}\}$.

Other parameters used in this article are summarized in Table 1.

This paper adopts vehicle re-identification as the authors' focus for the cross-camera video analytics application on the edge system. Vehicle re-identification is a more complex task compared to conventional video analytics tasks such as object detection and classification. While vehicle re-identification is a challenging and less explored area in previous studies, it holds significant potential in multi-camera systems for real-world applications.

The analysis of GPU utilization and memory consumption is performed during the application execution, and issues related to multi-process execution are noticed when running multiple tasks simultaneously on a single server. Specifically, the concurrent execution of several tasks on a single server revealed limitations in GPU utilization.

A. VEHICLE RE-IDENTIFICATION

Vehicle re-identification has become a popular and complex application in vehicle analytics today. Several competitions, such as the AICity Challenge, have included this topic for many years. The solutions for this application are diverse, yet they generally follow a common framework.

The general pipeline of vehicle re-identification applications includes three steps: object detection, feature extraction, and tracking [28]. Figure 5 shows the re-identification

TABLE 1. List of notations.

Notation	Definition
$\mathcal{V}$	Set of video streams (cameras)
v	A video stream ($v \in \mathcal{V}$)
$\mathcal{N}$	Set of Compute nodes
n	A compute node ($n \in \mathcal{N}$)
$\mathcal{E}$	Set of video stream, compute node mapping
e_{vn}	A set of binary variables ($e_{vn} \in \{0,1\} \forall v \in \mathcal{V}, \forall n \in \mathcal{N}$). $e_{vn} = 1$ if video stream v process on compute node n.
C_n	Compute resource of edge server n
C^R	Compute resource requirement for retraining
ϕ_n	A set of binary variables ($\phi_n \in \{0,1\}$). $\phi_n = 1$ if retrain on compute node n
C^I	Compute resource available for inference
C_n^I	Actual compute resource for inference allocated on server n
C_v^I	The actual compute resource allocated of each video stream v.
M^I	The GPU memory cost required for each application
M_n	The total GPU memory size on compute node n.
v_τ	workload of video stream v at time τ
T_r	The set of timestamps of each retraining decision window
τ_r	the time to make decision ($\tau_r \in T_r$).
T_o	The set of timestamps of each offloading decision window
τ_o	the time to make decision ($\tau_o \in T_o$).
$A_{v\tau}$	MOT Accuracy of video stream v at time τ
$D_{v\tau}$	Accumulated object numbers of video stream v at time τ
γ	compute resource discount factor for multi-process executing

TABLE 2. Executing time for each step in re-identification.

	Object Detection	Feature Extraction	Tracking
Time (ms)	7.6	76.7	12.3

pipeline of each video frame. The application will first get the vehicles at each video frame by objection detection. Second, use a deep learning model to generate each vehicle's feature vector, representing the vehicle's color and type. Third, the tracking module performs algorithms using features and history data to get the ID of each vehicle. The feature extraction and tracking step are called re-id in this paper for simplicity.

The result of analyzing the executing time of each step per frame can be found in Table 2. This application's feature extraction step takes significantly longer than other steps, indicating that it can be considered the critical step. It stems from the fact that the inference step for feature extraction is notably more intricate than other steps. Numerous objects in a single frame necessitate running inference multiple times.

Another important observation is that vehicle re-identification is a stateful application because the tracking algorithm relies on historical data to identify and track vehicles over time. Consequently, parallel execution of the re-identification process is impossible due to its stateful nature. This limitation gives rise to GPU utilization issues, which will be discussed in the following subsection.

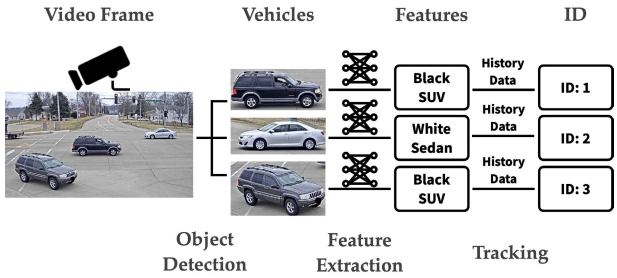

FIGURE 5. The re-identification pipeline of each video frame.

B. GPU UTILIZATION AND MEMORY

GPU utilization and memory are crucial parameters in video re-identification applications. Both parameters present individual challenges that can limit the overall system performance.

- **GPU Utilization**: GPU utilization, defined as the percentage of time over the past sample period during which one or more kernels are executing on the GPU, is a crucial metric for evaluating the efficiency of CNN inference [29]. However, it has often been overlooked in previous studies [30]. One reason is that many popular deep learning-based applications, such as object detection, classification, and anomaly detection, only contain CNN inference tasks, which inherently tend to achieve high GPU utilization in theory.
  The execution of vehicle re-identification must occur sequentially, and only some steps utilize the GPU so that the GPU utilization will not reach 100% theoretically. As a result, instead of modeling the execution time proportional to the total computing power of the server [22], the computing resource cost C^I is modeled as a fixed value even when the server has a larger capacity.
- **GPU Memory**: GPU memory refers to the amount of storage available on a GPU for storing and processing data. When the available GPU memory on a Compute node is insufficient, it can lead to application failures when running on the GPU. The size of the GPU memory on the Compute node becomes a limiting factor in determining the maximum number of applications that can be executed simultaneously on a single server.
  The limitations on server n with M_n GPU memory can be modeled using the following equation. M^I represents the memory cost of each application and indicates that the total GPU memory used by all applications running on a server cannot exceed the server's limit.

$$\sum_{v \in \mathcal{V}} e_{vn} \cdot M^I \leq M_n \quad \forall n \in \mathcal{N} \tag{1}$$

C. MULTI-PROCESS EXECUTING MODEL

Several GPU-related processes would execute simultaneously on a single server in the multi-camera video analytics system. It is observed that when multiple processes run

on a single server, the GPU utilization does not increase proportionately to the number of processes. To address this issue, a modifier γ is added to the compute resource of each process C^I. C_n^I and C_v^I represent the actual resource cost for all applications and each video stream.

The total compute resource of inference process C_n^I on the server n can be modeled as the following equation.

$$\begin{aligned} 0 < \gamma \leq 1 \\ \gamma \cdot \sum_{v \in \mathcal{V}} e_{vn} \cdot C^I = C_n^I \quad \forall n \in \mathcal{N} \\ \sum_{v \in \mathcal{V}} e_{vn} \cdot C_v^I = C_n^I \quad \forall n \in \mathcal{N} \end{aligned} \tag{2}$$

D. VIDEO ANALYTIC THROUGHPUT MODEL

System throughput emerges as a crucial metric in evaluating the performance of the vehicle analytics system. This study defines the throughput in terms of frame processed per second (FPPS). Therefore, the throughput of the video stream v at time τ is modeled by following formula:

$$\text{FPPS} = \frac{C_v^I}{v_\tau \cdot k} \tag{3}$$

The throughput is the resource allocated for this stream divided by the resource required by the application. Here, k is a constant representing the execution speed per workload. The total throughput on Compute node n can be calculated by summing up all the video streams on this server, as indicated by the following formula:

$$\text{FPPS}_n = \sum_{v \in \mathcal{V}} e_{vn} \cdot \frac{C_n^I / \sum_{v \in \mathcal{V}} e_{vn}}{v_\tau \cdot k} \tag{4}$$

IV. PROBLEM FORMULATION

This section provides the mathematical formulation of this work. The objective is to optimize the overall throughput of the video analytics service across all Compute nodes. The overall problem can be formulated as:

$$\begin{aligned} \max \text{FPPS}_{\text{system}} &= \max \sum_{n \in \mathcal{N}} \text{FPPS}_n \\ &= \max \sum_{n \in \mathcal{N}} \sum_{v \in \mathcal{V}} e_{vn} \cdot \frac{C_v^I}{v_\tau \cdot k} \\ &= \max \sum_{n \in \mathcal{N}} \sum_{v \in \mathcal{V}} e_{vn} \cdot \frac{C_n^I / \sum_{v \in \mathcal{V}} e_{vn}}{v_\tau \cdot k} \end{aligned} \tag{5}$$

Constraints:

$$\begin{aligned} C1: &\quad \phi_n \cdot C^R + C_n^I \leq C_n \quad \forall n \in \mathcal{N} \\ C2: &\quad C^I \cdot \gamma \sum_{v \in \mathcal{V}} e_{vn} = C_n^I \quad \forall n \in \mathcal{N} \\ C3: &\sum_{v \in \mathcal{V}} e_{vn} \cdot M^I \leq M_n \quad \forall n \in \mathcal{N} \\ C4: &\begin{cases} e_{vn} \in \{0, 1\} & \forall v \in \mathcal{V}, \forall n \in \mathcal{N} \\ \sum_{n \in N} e_{vn} = 1 & \forall v \in \mathcal{V} \end{cases} \\ C5: &\begin{cases} \phi_n \in \{0, 1\} & \forall n \in \mathcal{N} \\ \sum_{n \in \mathcal{N}} \phi_n = 1 \end{cases} \end{aligned} \tag{6}$$

Constraint C1 states that the overall compute resource cannot exceed the server capacity. Constraint C2 represents the multi-process limitation, as discussed in Section III-C. Constraint C3 sets the maximum application process limit for each server, as described in Section III-B. Constraint C4 means that one video stream can only be processed by one Compute node. Constraint C5 guarantees that the retraining process can only run on one Compute node in the cluster.

A heuristic algorithm is designed to schedule the retraining process and fully utilize the compute resource to maximize the overall throughput. The algorithm is decoupled into two stages, the model retraining and workload offloading, and the Edge/Fog orchestrator and Control node manage these two stages, respectively.

There are three benefits to decomposing and managing the solutions by different roles.

- **Network bandwidth**: Workload offloading requires significant network bandwidth. The transmission overhead can only be ignored in a small area in a single cluster. Second, the fine-tuned model generated by model retraining can be reused for different regions.
- **Model reusing**: The fine-tuned model generated by model retraining can be reused for different regions. The Edge/Fog orchestrator can manage these models effectively.
- **Time scale**: The timescale of the two problems is different. The workload on each camera is dynamic over seconds, while the accuracy only drops over a long period. The algorithm has to perform offloading much more frequently than model retraining.

Figure 6 illustrates the system workflow of the two-staged algorithm. In the first stage, the system makes the model retraining decision on EFO using metrics collected from the Compute node. In the second stage, the system balances the workload across the system based on calculations performed by the Control node.

A. FIRST STAGE: MODEL RETRAINING SCHEDULER

The model retraining scheduler is a heuristic method for Edge/Fog orchestrator that aims to decide the retraining decision ϕ_n. The cameras in the same cluster are near each other geographically, and the video captured by cameras within a cluster is similar. As a result, multiple retraining processes executing simultaneously are restricted in the same cluster, which is the constraint C5.

The model retraining scheduler makes the retraining decisions over some time τ_r. Let the set including τ_r be T_r. For the sake of simplicity, in the following part, it is assumed that the algorithm operates at time τ.

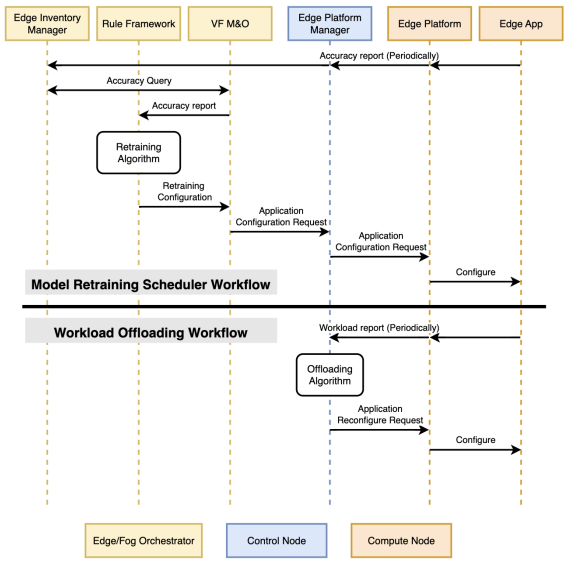

FIGURE 6. The system workflow of proposed algorithm.

The algorithm begins with initiating the retraining candidates set $\mathcal{M}$ as an empty set. For each Compute node n, calculate the average accuracy A_n and the accumulated data size D_n.

If A_n is lower than threshold α and D_n is larger than δ to initiate the training progress, make Compute node n as a candidate to start the retraining progress and append n into retraining candidates set $\mathcal{M}$. After calculating on all Compute nodes, select the node with the lowest accuracy among all retraining candidates. Since each Compute node is traversed once, the overall computational complexity is $O(N)$. For a detailed step-by-step description of the algorithm, refer to Algorithm 1.

Algorithm 1 Model Retraining Scheduler

Input: Compute node capacity C_n, camera-server mapping e_{vn}
Output: Retraining decision ϕ_n

$\mathcal{M} \leftarrow \emptyset$
foreach $n \in N$ **do**
  $A_n \leftarrow \frac{1}{\sum_{v\in\mathcal{V}} e_{vn}} \sum_{v\in\mathcal{V}} e_{vn} \cdot A_v$
  $D_n \leftarrow \sum_{v\in\mathcal{V}} e_{vn} \cdot D_v$
  if $A_n \leq \alpha$ *and* $D_n \geq \delta$ **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{n\}$
  end
end
$i \leftarrow \arg\min_{m\in\mathcal{M}} A_m$
$\phi_i \leftarrow 1$

B. SECOND STAGE: WORKLOAD OFFLOADING

In this stage, workload offloading is the heuristic algorithm for the Compute node that aims to balance the workload between servers in the cluster. In general, the Control node monitors the workload of each video analytic pipeline in the cluster and modifies the execution server of each pipeline. Given that the Compute nodes are close to one another and benefit from extensive network bandwidth, the transmission overhead from offloading data within the cluster is disregarded.

The algorithm will execute continuously at intervals of τ_o, as the video content is streamed without interruption, and the workload needs to be adjusted dynamically. Let the set including τ_o be T_o. For the sake of simplicity, in the following part, it is assumed that the algorithm operates at time τ.

Let the total workloads on Compute node n be w_n. Since the compute resource on each server may not be the same, calculate the adjusted workload w'_n based on the computing power of each server. These parameters can be computed as

$$w_n = \sum_{v\in\mathcal{V}} e_{vn} \cdot v_\tau \tag{7}$$

$$w'_n = w_n \cdot \frac{C_n}{\bar{C_n}} \tag{8}$$

Let the Compute node with the maximum and minimum workload be node i, and node j, respectively, and the workload difference between i and j be Δ. If Δ is larger than two times the minimum workload u on node i, offload this video stream and continue to perform the previous balancing procedure. The complexity of this algorithm is constrained by the quantity of video streams, expressed as $O(V)$. Refer to Algorithm 2 for a detailed step-by-step algorithm description.

Algorithm 2 Workload Offloading

Input: Video stream workload v_τ, compute resource C_n, GPU memory M_n
Output: The video process decision e_{vn}

$w_n \leftarrow \sum_{v\in\mathcal{V}} e_{vn} \cdot v_\tau$
$w'_n \leftarrow w_n \cdot \frac{C_n}{\bar{C_n}}$
$i \leftarrow \arg\max_{n\in\mathcal{N}} w'_n, j \leftarrow \arg\min_{n\in\mathcal{N}} w'_n$
$\Delta \leftarrow w'_i - w'_j$
$u \leftarrow \arg\min_{v,e_{vi}=1} v_\tau$
while $\Delta > 2 \cdot u_\tau$ **do**
  if $\sum_{v\in\mathcal{V}} e_{vj} \cdot M^I > M_j$ **then**
    break
  end
  $e_{ui} \leftarrow 0$
  $e_{uj} \leftarrow 1$
  $w_n \leftarrow \sum_{v\in\mathcal{V}} e_{vn} \cdot v_\tau$
  $w'_n \leftarrow w_n \cdot \frac{C_n}{\bar{C_n}}$
  $i \leftarrow \arg\max_{n\in\mathcal{N}} w'_n, j \leftarrow \arg\min_{n\in\mathcal{N}} w'_n$
  $\Delta \leftarrow w'_i - w'_j$
  $u \leftarrow \arg\min_{v,e_{vi}=1} v_\tau$
end

V. EVALUATION

This section first provides an overview of the experimental environment for evaluating the proposed scheme,

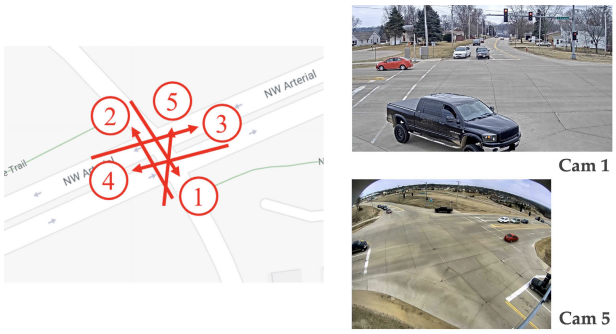

FIGURE 7. A part of the CityFlowV2 dataset includes five cameras situated at a road intersection. The images at the right hand side are the view of Camera 1 and Camera 5.

FIGURE 8. Example vehicle bounding-box and whole scene images of the VRIC benchmark.

including the dataset and the testbed information. Subsequently, the proposed algorithms are evaluated by comparing them against existing algorithms under different system configurations.

A. SYSTEM IMPLEMENTATION

1) DATASET

Two different datasets are used for the vehicle re-identification application training and testing: CityflowV2 [31] and VRIC [32].

- **CityFlowV2**: CityFlowV2 is an official dataset of the 6th AI City Challenge Track 1: City-Scale Multi-Camera Vehicle Tracking, the updated version of CityFlow. The dataset consists of videos captured by 46 cameras in a real-world traffic surveillance environment. There are 215.03 minutes of videos in total, and most of the videos have a frame rate of 10 FPS. Part of the CityFlowV2 dataset is shown in Figure 7, which includes five cameras in the same intersection from different perspectives.
- **VRIC**: VRIC is another vehicle re-identification dataset that includes variations in resolution, motion blur, illumination, occlusion, and viewpoint. VRIC contains 60,430 images of 5,622 vehicle identities that are captured by 60 different cameras at heterogeneous road traffic scenes. The example images that cropped from the original paper are shown in Figure 8.

2) VIDEO ANALYTIC APPLICATION

This section introduces how the application is built and highlight the available configurable options.

- **Video re-identification application**: The general application is a modification of a Python project developed by Regob [33]. The project is well-structured and offers simple configuration options for customization, such as model usage, tracking algorithm selection, and dataset format.
- **Object detection model**: As the first step in the vehicle re-identification pipeline, there are several options, such as simple background subtraction, YOLO [34], and Mask R-CNN [35]. Both YOLO and Mask R-CNN have been hugely influential in the field of object detection. YOLO is known for its speed and efficiency, and Mask R-CNN achieves high accuracy but can be slower.
  The YOLO model is selected, specifically implementing YOLOv5s weights due to its effectiveness and lightweight design.
- **Feature extraction CNN model**: VRIC dataset is utilized to train the main feature extraction model and perform the vehicle analytic application on the CityflowV2 dataset. For the model's architecture, the Resnet-IBN network is adopted and trained for 20 epochs. In the retraining process, the model is fine-tuned for five epochs.
- **Tracking**: Among the various tracking algorithms available, including DeepSORT [36], and ByteTrack [37]. DeepSORT introduces a neural network to compute the appearance descriptors, and this deep association metric significantly improves tracking performance. ByteTrack is a newer tracking algorithm that simplifies the tracking problem into a top-k list update problem and achieves high accuracy. DeepSORT is chosen for the application, given its popularity and widespread usage in several benchmark datasets and applications.
- **Logging and Monitoring**: The Weights and Biases (W&B) library is used for metrics logging and monitoring. W&B is a machine learning toolset that helps researchers and developers track and visualize their models' performance. This research utilizes its logging system and dashboard service that can plot metrics in real-time. Additionally, Weights and Biases can log system usage simultaneously, which is extremely helpful in the experiments.

It is noted that the application benefits from edge computing because it contains monitoring, tracking, and object detection, which requires real-time surveillance, low latency, and well-organized resource distribution.

3) HARDWARE

The server, which acts as a Compute node, is equipped with an AMD Ryzen 7 5800X CPU and an NVIDIA GeForce RTX 3080 GPU. The server has 8 CPU cores, 64 GiB of memory, and 10 GiB of GPU memory.

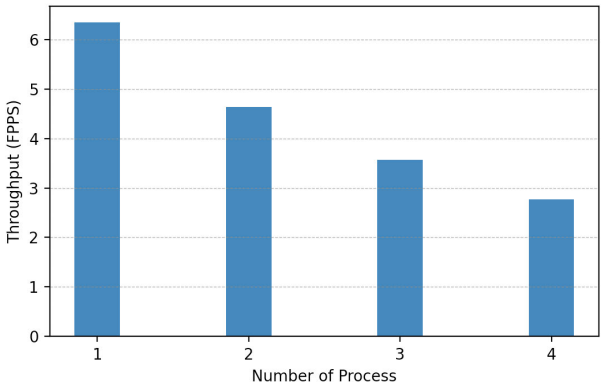

FIGURE 9. Relationship of throughput and process number on a single server.

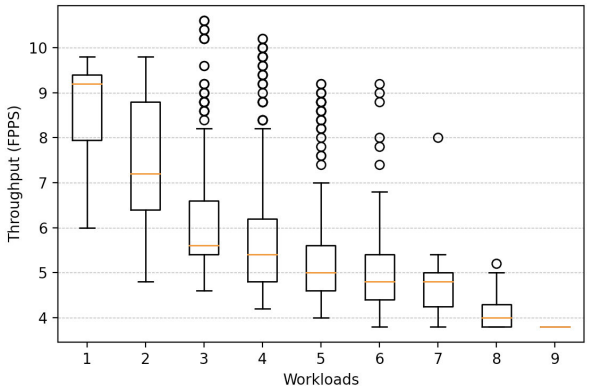

FIGURE 10. Relationship of throughput and workloads.

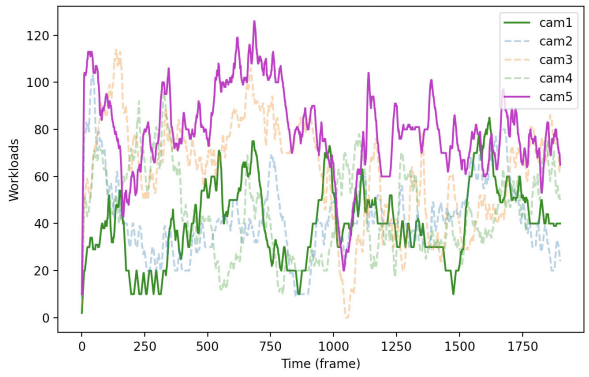

FIGURE 11. Workload dynamics of five cameras in CityFlowV2 dataset.

B. EXPERIMENTAL SETTINGS

The metrics of system throughput and latency are adopted to evaluate the performance of the proposed approach. The throughput is related to FPPS and the latency is to the average task execution time and queueing time for all streams.

The system schedules the video analytics tasks with offloading. The system performance is evaluated and compared against several benchmark solutions.

- **Randomly mapping**: The cameras randomly map to the Compute node to start the video analytic application.
- **Evenly mapping**: The cameras map to the Compute node evenly. That is, the number of video streams executing on each Compute node is evenly distributed.
- **WMA**: The cameras map to Compute nodes by workload and perform the offloading mechanism when the workload changes.

The impact of the retraining mechanism is also assessed within the system. The label **noR**, appended at the end of each scheme, signifies that the retraining mechanism is not being utilized in that particular scheme.

C. ANALYSIS OF SYSTEM PERFORMANCE

The system performance is observed and analyzed based on various factors, including the number of processes, dynamic workloads, Compute nodes, and cameras.

1) IMPACT OF MULTIPLE PROCESSES EXECUTION

As mentioned in Section III-C, performance decreases when multiple processes are executed on a single server. In the evaluation, the processing throughput of a single video stream is measured when multiple processes are executing simultaneously. This analysis aims to understand how the concurrent execution of processes affects the performance of the video stream processing.

As depicted in Figure 9, the throughput decreases as the number of processes increases. Specifically, the throughput decreases by a factor of 75% with each additional process.

2) RELATIONSHIP BETWEEN WORKLOAD AND THROUGHPUT

The video re-identification pipeline's processing speed is highly influenced by the objects or workload in each frame, primarily due to the critical feature extraction step mentioned in Section III-A.

Figure 10 depicts the relationship between workloads and throughput, illustrating a negative proportionality. As observed, there is an inverse correlation between workload and processing speed, meaning that as the workload increases, the processing speed decreases.

3) IMPACT OF DYNAMIC WORKLOAD

The performance of WMA and various baselines are evaluated under dynamic workloads. Figure 11 shows that the workloads across five cameras in the dataset are dynamic over time. One notable observation is that Camera 5 has a higher average workload than the other cameras, achieving more than two times the workload of Camera 1. As shown in Figure 7, Camera 5 is a wide-angle camera with a larger field of view than the others. This hardware difference results in more vehicles appearing in the view of Camera 5, leading to higher workloads.

As shown in Figure 12, the average system throughput of WMA outperforms the baselines by 12% and 32%. In the schemes without the retraining algorithm, WMA still outperforms the baselines, achieving improvements of 10% and 27%, respectively. The retraining decision is made at frame 1000, leading to a significant drop in the throughput

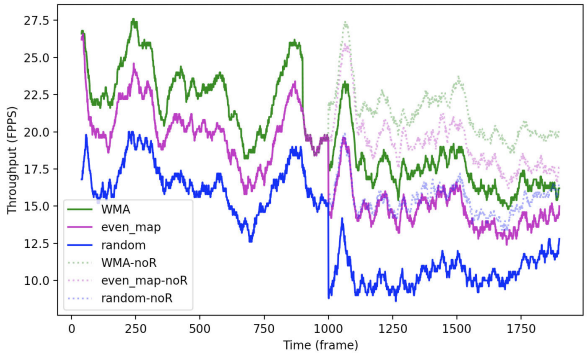

FIGURE 12. Relationship of the system throughput and dynamic workloads.

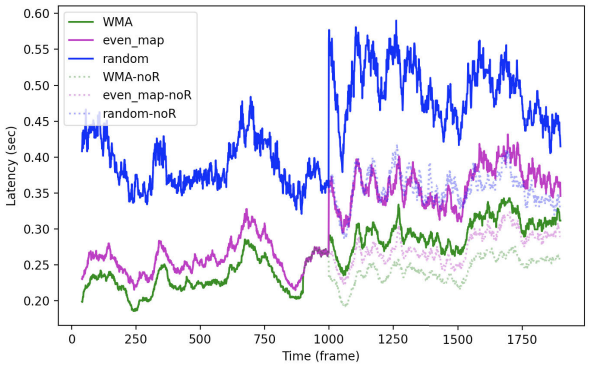

FIGURE 13. Relationship of the system latency and dynamic workloads.

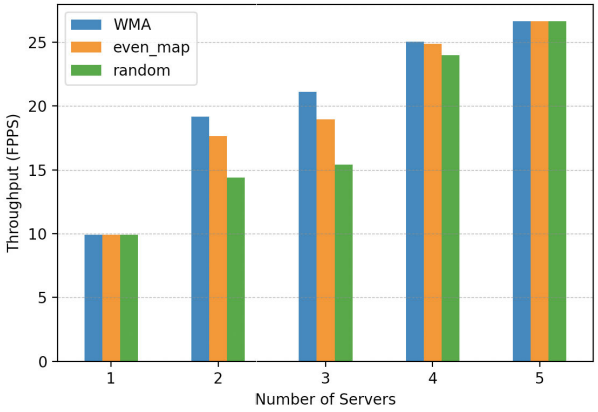

FIGURE 14. Relationship of system throughput and the number of compute nodes.

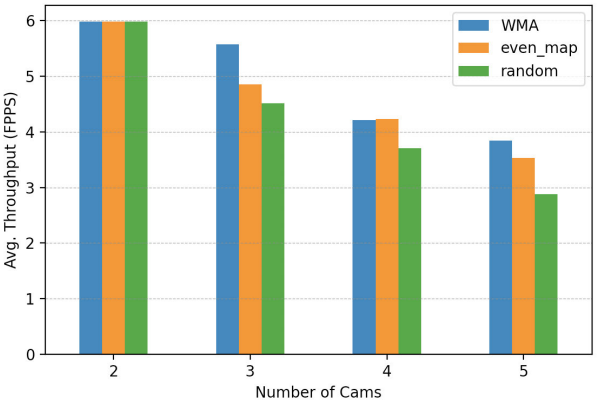

FIGURE 15. Relationship of average throughput and the number of cameras.

for each scheme. The retraining mechanism results in an 8%, 11%, and 15% drop in WMA and the baselines, respectively. The trends over time of the three schemes are similar since the throughput is highly related to the workloads. The evenly-mapping scheme can achieve relatively high performance because the workload between the dataset's cameras is not significantly imbalanced. When the workload distribution is relatively even across cameras, the evenly-mapping scheme can allocate resources efficiently and avoid overloading specific cameras.

The average latency of each video stream under dynamic workloads is also evaluated. As shown in Figure 13, the average latency of WMA surpasses the baselines by 15% and 65%. Even in the schemes that do not employ the retraining algorithm, the system continues to outperform the baselines, demonstrating improvements of 12% and 55% improvement, respectively. Introducing the retraining mechanism leads to an increase of 7%, 13%, and 14% in WMA and the respective baselines.

4) IMPACT OF THE NUMBER OF COMPUTE NODES

The number of the input video stream is set to 5. The WMA is evaluated by different numbers of Compute nodes from 1 to 5, as shown in Figure 14. The performance of one Compute node and five Compute nodes is identical to the baseline in this context because no decision-making is involved in either situation. When the number of Compute nodes is set to 1, all video streams can only process on a single server. Each video stream is processed at a different server if the number of Compute nodes is set to 5. Each video stream is processed on a different server when the number of Compute nodes is set to 5. Both configurations need no offload decision.

The performance of the configuration with 2 and 3 Compute nodes shows significant improvements compared to the randomly mapped baseline. In terms of overall throughput, these configurations exhibit a notable gain, with an improvement of 25% and 27%, respectively.

5) IMPACT OF THE NUMBER OF CAMERAS

The number of the Compute nodes is set to be 2, and WMA is evaluated by different numbers of cameras from 2 to 5, as shown in Figure 15. The throughput is normalized by the number of cameras. The performance of the 2-camera setup is identical to the baseline because there is no decision-making involved in this scenario, where each camera has its own isolated Compute node to utilize. In the cases of 3 and 5 cameras, the throughput improved by 19% and 25%, respectively. The case of 4 cameras does not show improvement since having two cameras on two servers is

already well-balanced in the authors' dataset. Besides, it is observed that as the number of cameras increases, the average throughput decreases since the size of the compute resource is fixed.

VI. CONCLUSION

This paper proposes WMA, a two-staged resource allocation method on a three-tiered cross-camera video analytic system. The system supports both model fine-tuning and workload balancing. The system architecture and control workflow are consistent with the IEEE 1935 edge standard. The research first digs into the GPU utilization performance of vehicle re-identification applications and then investigates the camera workload dynamics. The system evaluation is performed with a commonly-used dataset. The results show that the proposed design outperforms the baseline and increases the system's overall throughput across cameras. As part of future work, the authors would like to add configurable options to the camera input, such as video resolution, FPS, and hardware information. The selection of these parameters can provide more options for the proposed algorithm, further enhancing overall performance. Moreover, the system can be evaluated using larger datasets or practical settings. These add-ons will provide a more significant variation in video content, enhancing the robustness of the evaluations.

In considering future work, exploring the generality and robustness of the proposed system model in light of advanced software and hardware optimizations would be valuable. The workload during inference can fluctuate in computer vision pipelines based on image or video content. The impact of practical optimizations, such as adaptive resource allocation for varying workloads or spatial sharing of GPU resources, should be investigated to understand how these optimizations influence the results of the system model. In addition, it is valuable to discuss the spatial sharing of the GPU among multiple processes, an optimization extensively explored in both academic and industry settings for enhancing inference efficiency. A comprehensive exploration of these factors could provide insights into the scalability and adaptability of the proposed model in real-world, dynamic computing environments.

REFERENCES

[1] G. Ananthanarayanan, P. Bahl, P. Bodík, 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.

[2] H. Guo, S. Yao, Z. Yang, Q. Zhou, and K. Nahrstedt, "CrossRoI: Cross-camera region of interest optimization for efficient real time video analytics at scale," in *Proc. 12th ACM Multimedia Syst. Conf.* New York, NY, USA: Association for Computing Machinery, Jul. 2021, pp. 186–199, doi: 10.1145/3458305.3463381.

[3] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, P. Bahl, and J. Gonzalez, "Spatula: Efficient cross-camera video analytics on large camera networks," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Nov. 2020, pp. 110–124.

[4] 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.*, Feb. 2019, pp. 9–14, doi: 10.1145/3301293.3302366.

[5] Y. Chiang, Y. Zhang, H. Luo, T.-Y. Chen, G.-H. Chen, H.-T. Chen, Y.-J. Wang, H.-Y. Wei, and C.-T. Chou, "Management and orchestration of edge computing for IoT: A comprehensive survey," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14307–14331, Aug. 2023.

[6] (2020). *Multi-Access Edge Computing-Standards for MEC-ETSI*. [Online]. Available: https://www.etsi.org/technologies/multi-access-edge-computing

[7] *IEEE Standard for Edge/Fog Manageability and Orchestration*, IEEE Standard 1935-2023, 2023. [Online]. Available: https://standards.ieee.org/ieee/1935/7181/

[8] 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. Networked Syst. Design Implement. (NSDI)*. Boston, MA, USA: USENIX Association, Mar. 2017, pp. 377–392. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/zhang

[9] X. Zeng, B. Fang, H. Shen, and M. Zhang, "Distream: Scaling live video analytics with workload-adaptive distributed edge intelligence," in *Proc. 18th ACM Conf. Embedded Netw. Sensor Syst.* New York, NY, USA: Association for Computing Machinery, 2020, pp. 409–421.

[10] Y. Zhang, J.-H. Liu, C.-Y. Wang, and H.-Y. Wei, "Decomposable intelligence on cloud-edge IoT framework for live video analytics," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8860–8873, Sep. 2020.

[11] C. Rong, J. H. Wang, J. Liu, J. Wang, F. Li, and X. Huang, "Scheduling massive camera streams to optimize large-scale live video analytics," *IEEE/ACM Trans. Netw.*, vol. 30, no. 2, pp. 867–880, Apr. 2022.

[12] Y.-T. Yang and H.-Y. Wei, "A coalition formation approach for privacy and energy-aware split deep learning inference in edge camera network," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 3, pp. 3673–3685, Sep. 2022.

[13] Y. Chiang, C.-H. Hsu, G.-H. Chen, and H.-Y. Wei, "Deep Q-learning-based dynamic network slicing and task offloading in edge network," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 1, pp. 369–384, Mar. 2023.

[14] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *Proc. 19th USENIX Symp. Networked Syst. Design Implement. (NSDI)*. Renton, WA, USA: USENIX Association, Apr. 2022, pp. 119–135. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/bhardwaj

[15] M. Khani, G. Ananthanarayanan, K. Hsieh, J. Jiang, R. Netravali, Y. Shu, M. Alizadeh, and V. Bahl, "RECL: Responsive resource-efficient continuous learning for video analytics," in *Proc. 20th USENIX Symp. Networked Syst. Design Implement. (NSDI)*. Boston, MA, USA: USENIX Association, Apr. 2023, pp. 917–932. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/khani

[16] T.-Y. Chen, Y. Chiang, J.-H. Wu, H.-T. Chen, C.-C. Chen, and H.-Y. Wei, "IEEE P1935 edge/fog manageability and orchestration: Standard and usage example," in *Proc. IEEE Int. Conf. Edge Comput. Commun. (EDGE)*. Cincinnati, OH, USA: Standard, Jul. 2023, pp. 96–103.

[17] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, Jul. 2020, pp. 359–376, doi: 10.1145/3387514.3405874.

[18] M. Khani, P. Hamadanian, A. Nasr-Esfahany, and M. Alizadeh, "Real-time video inference on edge devices via adaptive model streaming," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4552–4562.

[19] A. Aliouat, N. Kouadria, M. Maimour, S. Harize, and N. Doghmane, "Region-of-interest based video coding strategy for rate/energy-constrained smart surveillance systems using WMSNs," *Ad Hoc Netw.*, vol. 140, Mar. 2023, Art. no. 103076, doi: 10.1016/j.adhoc.2022.103076.

[20] 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.

[21] F. Faticanti, F. Bronzino, and F. De Pellegrini, "The case for admission control of mobile cameras into the live video analytics pipeline," in *Proc. 3rd ACM Workshop Hot Topics Video Analytics Intell. Edges*. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 25–30, doi: 10.1145/3477083.3480151.

[22] M. Zhang, J. Cao, Y. Sahni, Q. Chen, S. Jiang, and L. Yang, "Blockchain-based collaborative edge intelligence for trustworthy and real-time video surveillance," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1623–1633, Feb. 2023.

[23] 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.*, Aug. 2018, pp. 253–266, doi: 10.1145/3230543.3230574.
[24] H. B. Pasandi and T. Nadeem, "CONVINCE: Collaborative cross-camera video analytics at the edge," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2020, pp. 1–5.
[25] J. Li, J. Xu, F. Zhong, X. Kong, Y. Qiao, and Y. Wang, "Pose-assisted multi-camera collaboration for active object tracking," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 1, pp. 759–766.
[26] J. Yi, C. Min, and F. Kawsar, "Vision paper: Towards software-defined video analytics with cross-camera collaboration," in *Proc. 19th ACM Conf. Embedded Networked Sensor Syst.*, Nov. 2021, pp. 474–477, doi: 10.1145/3485730.3493453.
[27] G. H. Apostolo, P. Bauszat, V. Nigade, H. E. Bal, and L. Wang, "Live video analytics as a service," in *Proc. EuroMLSys*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 37–44, doi: 10.1145/3517207.3526973.
[28] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. H. Hoi, "Deep learning for person re-identification: A survey and outlook," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 6, pp. 2872–2893, Jun. 2022.
[29] (2023). *NVML API Reference Guide*. Accessed: Jul. 7, 2023. [Online]. Available: https://docs.nvidia.com/deploy/pdf/NVML_API_Reference_Guide.pdf
[30] J. Kosaian and A. Phanishayee, "A study on the intersection of GPU utilization and CNN inference," 2022, *arXiv:2212.07936*.
[31] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "CityFlow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8789–8798.
[32] A. Kanacı, X. Zhu, and S. Gong, "Vehicle re-identification in context," in *Proc. 40th German Conf. Pattern Recognit. (GCPR)*, Stuttgart, Germany. Cham, Switzerland: Springer, Oct. 2019, pp. 377–390.
[33] Regob. (Jan. 2023). *Multi-Target Multi-Camera (MTMC) Object Tracking*. [Online]. Available: https://github.com/regob/vehicle_mtmc
[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 2980–2988.
[36] 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.
[37] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-object tracking by associating every detection box," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 1–21.

HUAN-TING CHEN received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2021 and 2023, respectively. His research interests include edge computing, video analytics, and machine learning.

YAO CHIANG (Member, IEEE) received the B.S. and M.S. degrees in management information systems from the National Pingtung University of Science and Technology, Pingtung, Taiwan, in 2014 and 2016, respectively, and the Ph.D. degree in electrical engineering from National Taiwan University (NTU), in 2021. He is currently a Postdoctoral Research Fellow with NTU. His research interests include data mining, machine learning, and mobile communications design for multi-access edge computing systems.

HUNG-YU WEI (Senior Member, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, in 1999, and the M.S. and Ph.D. degrees in electrical engineering from Columbia University, in 2001 and 2005, respectively.

He was a Summer Intern with Telcordia Applied Research, in 2000 and 2001. He was with NEC Labs America, from 2003 to 2005. He joined the Department of Electrical Engineering, National Taiwan University, in July 2005. He served as an Associate Department Chair and an Interim Department Chair, from 2019 to 2022. He is currently a Professor with the Department of Electrical Engineering and the Graduate Institute of Communications Engineering, National Taiwan University. His research interests include next-generation wireless broadband networks, the IoT, vehicular networking, fog/edge computing, cross-layer design and optimization in wireless multimedia communications, and game theoretical models for communications networks.

Dr. Wei received the NTU Excellent Teaching Award, in 2008 and 2018. He also received the Recruiting Outstanding Young Scholar Award from the Foundation for the Advancement of Outstanding Scholarship, in 2006; the K. T. Li Young Researcher Award from ACM Taipei/Taiwan Chapter and the Institute of Information and Computing Machinery, in 2012; the Ministry of Science and Technology Research Project for Excellent Young Scholars, in 2014; the Excellent Young Engineer Award from the Chinese Institute of Electrical Engineering, in 2014; the Wu Ta You Memorial Award from MOST, in 2015; and the Outstanding Research Award from MOST, in 2020. He was a Consulting Member of the Acts and Regulation Committee of the National Communications Commission from 2008 to 2009. He served as the Division Director for the NTU Computer and Information Networking Center, from 2016 to 2017. He has been actively participating in NGMN, IEEE 802.16, 3GPP, IEEE P1934, and IEEE P1935 Standardization. He serves as the Vice Chair for the IEEE P1934 Working Group to standardize fog computing and networking architecture. He serves as the Secretary for the IEEE Fog/Edge Industry Community. He also serves as an Associate Editor for IEEE Internet of Things Journal. He is an IEEE Certified Wireless Communications Professional. He was the Chair of IEEE VTS Taipei Chapter, from 2016 to 2017. He is currently the Chair of the IEEE P1935 Working Group for Edge/Fog Management and Orchestration Standard.