Blockchain-based Collaborative Edge Intelligence for Trustworthy and Real-time Video Surveillance

Mingjin Zhang, Jiannong Cao, *Fellow, IEEE*, Yuvraj Sahni, Qianyi Chen, Shan Jiang, Lei Yang

***Abstract*—Trustworthy and real-time video surveillance aims to analyze the live camera streams in a privacy-preserving manner for the decision-making of various advanced services, such as pedestrian re-identification and traffic monitoring. In recent years, edge computing has been identified as a promising technology for trustworthy and real-time video surveillance because it keeps confidential video data locally and reduces the latency caused by massive data transmission. Generally, a single edge device can hardly afford the computation-intensive video analytics tasks. Most existing solutions incorporate cloud servers to handle the overloaded tasks. However, such an edge-cloud collaboration approach still suffers from unpredictable latency and privacy concerns because the remote cloud is centralized and distant from the cameras. In this work, we designed a blockchain-based collaborative edge intelligence (BCEI) approach for trustworthy and real-time video surveillance. In BCEI, geo-distributed edge devices form a peer-to-peer network to maintain a permissioned blockchain and share data and computation resources to perform computation-intensive video analytics tasks. The video analytics results are written on the blockchain in an immutable manner to guarantee trustworthiness. To reduce task execution time, we formulate and solve a joint stream mapping and task scheduling problem to schedule video streams and machine learning models among edge devices. A pedestrian re-identification prototype is implemented and deployed based on BCEI with extensive performance evaluation, indicating the superiority of BCEI in latency reduction and system throughput improvement by leveraging collaboration among edge devices.**

***Keywords*-Collaborative Edge Computing; Edge Blockchain; Trustworthiness; Edge Intelligence; Video Surveillance.**

I. Introduction

Nowadays, cameras have been widely adopted and deployed in public and private areas, such as traffic intersections, campuses, and grocery stores [1]. Driven by the recent breakthrough in machine learning, especially deep learning, we can perform advanced video analytics from the camera streams with powerful deep learning models [2]. Collaborative video surveillance aims to analyze the live video streams from the distributed cameras to support a wide range of applications, including traffic control [3], security monitoring [4], and object re-identification [5]. Because many of these applications impose high requirements for real-time responses, it is highly demanded to achieve low-latency and high-throughput video stream processing.

In academia and industry, there are many existing cloud-based video surveillance solutions [6]. These solutions are centralized and reactive [7]. More specifically, video streams collected from different cameras are sent to a centralized cloud server for storage and analytics [8]. Due to the strong dependence on the central cloud server and massive data transmission among cameras and servers, such a cloud-based approach incurs bandwidth congestion, single point of failure, low scalability, and high latency. Furthermore, the cloud servers are not always reliable, raising privacy concerns [9], especially when the videos are confidential, e.g., smart home and warehousing [10].

In the past few years, there have been emerging studies of edge computing-based video surveillance that can reduce latency, improve scalability, and provide better services than cloud-based solutions. The magic behind edge computing lies in pushing computation tasks from the cloud to network edges closer to the data sources [11]. Some existing edge computing-based video analytics solutions have considered leveraging the spatial and temporal correlations among different cameras to fully discover the correlation among distributed cameras [12]. There are also proactive video surveillance solutions for vehicle tracking using edge devices [13]. Some studies further considered the aspects related to workload balancing to improve the performance of each edge device and the whole video analytics applications [14], [15].

To summarize, existing studies mainly consider sending the video streams to a nearby edge computing device, which may lead to overloading and degrade application performance dramatically [16]. It is because video analytic applications are usually computation-intensive while multiple video streams share the limited computation resources of a single edge device [17]. A bunch of work handles the resource constraints of a single edge device by offloading some computation tasks to the remote cloud [6], [18]. However, it bears unpredictable latency, limited bandwidth, and privacy concerns [8]. In addition, existing studies seldom consider the trustworthiness of video analytics results. In this work, we leverage the blockchain and collaborative edge intelligence, where geo-distributed edge devices share computation and data resources to accomplish the video surveillance tasks and collaborate to authenticate the video analytics results.

In this work, we propose a blockchain-based collaborative edge intelligence (BCEI) approach to support trustworthy and real-time video surveillance. BCEI enables proactive and trustworthy video surveillance. A large number of edge devices form a peer-to-peer network and maintain a permissioned blockchain to authenticate the video analytics results. In BCEI, the permissioned blockchain is employed because of its relatively low resource demands compared to the public

Mingjin Zhang, Jiannong Cao, Qianyi Chen, and Shan Jiang were with the Department of Computing, The Hong Kong Polytechnic University. Yuvraj Sahni was with the Department of Building Environment and Energy Engineering, The Hong Kong Polytechnic University. Lei Yang was with the School of Software Engineering, South China University of Technology.

Corresponding author: Shan Jiang

blockchain [19]. The cameras stream the videos to nearby edge devices sharing computation resources and data. One of the key challenges in the system is how to design approaches that can efficiently sense the status of underlying edge resources and intelligently assign the computation tasks among the edge devices. We design an online collaborative scheduler, taking the status of the resources of the distributed edge devices and the task characteristics as input and generating the distributed task execution policies. We formulate a joint stream mapping and task scheduling problem, which maps the video stream and distributes the computation tasks to multiple edge devices. By solving the problem, the scheduler decides which edge device the video stream should be assigned, how the tasks should be partitioned, and where to execute the distributed tasks. Our system enables efficient computation offloading and significantly improves resource utilization for edge computing-based video surveillance.

We have deployed a real-world prototype of pedestrian re-identification to examine the practicability and high efficiency of BCEI. More specifically, we deploy seven Internet Protocol (IP) cameras at different locations in an indoor environment to run the video surveillance task. These cameras send video streams to a cluster of geo-distributed edge devices. The results show that the proposed system can achieve nearly real-time performance. We have compared BCEI with several benchmark solutions concerning various performance metrics, including system throughput and latency. The results show that BCEI outperforms the state-of-the-art significantly.

The main contributions of this work are as follows:

- We propose blockchain-based collaborative edge intelligence (BCEI) approach, which is the first to study the blockchain-enabled collaboration among geo-distributed edge devices and generic for applications demanding trustworthiness and low latency.
- We apply BCEI for trustworthy and real-time video surveillance, in which we have studied a joint stream mapping and task scheduling problem for the first time and solved it using a heuristic algorithm.
- We deploy a real-world prototype of trustworthy and real-time video surveillance and conduct extensive performance evaluation. The results indicate the superiority of BCEI over the benchmark approaches in terms of latency reduction and system throughput improvement.

The remainder of this paper is organized as follows. Sec. II summarizes the related work and articulates the motivations of this work. Sec. III presents the system design of blockchain-based collaborative edge intelligence for trustworthy and real-time video surveillance. Sec. IV formulates the joint stream mapping and task scheduling problem and elaborates on the proposed heuristic algorithm. Sec. V shows the system implementation and performance evaluation. Finally, Sec. VI concludes this work and discusses the future directions.

II. Related Work

Real-time video analytics has been considered a killer application of edge computing [20]. Existing edge computing-based video analytics solutions mainly rely on the cooperation among cameras, edge devices, and cloud servers to accomplish real-time data analytics for computation-intensive and bandwidth-hungry video surveillance applications. Among them, solutions based on edge to cloud collaboration [6], [18], [7], [8] and edge to edge collaboration [15], [21], [22] have attracted most attention.

TABLE I
Comparison of the related work of video analytics

Video Analytics Solutions	Architecture	Adaptive Workload	Model Offloading	Stream Scheduling
VideoStorm [23]	Edge-Cloud	✗	✗	✗
Chameleon [24]	Edge-Cloud	✗	✗	✗
NoScope [6]	Edge-Cloud	✗	✗	✗
STVT [13]	Edge-Cloud	✗	✗	✗
LAVEA [22]	Edge-Edge	✗	✓	✗
VideoEdge [15]	Edge-Edge	✗	✗	✗
Distream [14]	Edge-Edge	✓	✓	✗
BCEI (this work)	Edge-Edge	✓	✓	✓

Edge-Cloud collaboration. Edge-cloud collaboration-based solutions can provide ample computation resources owing to the cloud platform. Many initial efforts on video analytics have considered edge-cloud collaboration. Zhang et al. [23] proposed VideoStorm, which leverages an online scheduler on a cloud server cluster to process queries of thousands of video streams sent by edge devices. Other studies also consider workload partition between edge and cloud. [18] divided the ResNet Model into three parts, which are deployed at the end, edge, and cloud, respectively. The three parts collaboratively perform the inference tasks submitted by users. Zhang et al. [7] adopted a serverless-based infrastructure to facilitate fine-grained and adaptive partitioning of cloud-edge workloads.

Although edge-cloud collaboration benefits from the paradigms of cloud computing and edge computing simultaneously, it still suffers from unpredictable latency because of the limited bandwidth between edge and cloud. Moreover, sending confidential video data to the cloud may cause privacy issues. Such solutions fail to explore the potential of local collaboration among edge devices to share resources.

Edge-Edge collaboration. Some work in the literature has considered edge-to-edge collaboration, where each edge device collaborates with others to provide surveillance services jointly. Neff et al. [21] proposed REVAMP^{2}T, a pedestrian re-identification algorithm that works on an encoded feature representation for each identified individual. Because no raw figure of pedestrians is shared among edge devices, privacy concerns are reduced. Another work in [25] also showed the advantage of leveraging spatial-temporal correlation to scale the video analytics systems to large camera deployments. The work in [26] presented the abstraction of camera clusters to provide video analytics service. However, those existing studies only support data-level cooperation, and the distributed computation resources are not fully utilized.

Few attempts have been made for sharing computation resources among multiple edge devices for video analytics. Yi et al. have designed LAVEA, a low-latency video edge analytic system that leverages nearby edge devices to reduce the overall task completion time [22]. However, this work fails to consider collaborative video surveillance, in which multiple video streams are from different locations. Recent work Distream [14] adaptively balances the workloads across multiple intelligent cameras and partitions the workloads between the smart cameras and the centralized edge cluster. However, the edge devices in the cluster are not geo-distributed and deployed centralized, which neglects the data transmission cost and network topology among edge devices.

Tab. I shows a comparison of the related work on video analytics with this work. Compared to existing studies, our work enables collaboration among geo-distributed edge devices by sharing both computation and data resources. It seamlessly integrates blockchain into the collaborative edge computing system to make the computation results trustworthy. Furthermore, we consider the stream transmission cost in the edge network and optimize the real-time performance of video analytics applications by jointly scheduling the camera streams and the machine learning models.

III. System Design

This section first introduces the system model of BCEI to deploy trustworthy and real-time video surveillance applications. Then, we discuss the system's key components to run the permissioned blockchain system, manage the heterogeneous edge resources, and distribute the computation tasks among geo-distributed edge devices.

A. System Model

Fig. 1 depicts the overall architecture of our proposed blockchain-based collaborative edge intelligence (BCEI) approach. Multiple cameras are deployed to perform trustworthy and real-time video surveillance tasks in a large-scale area. Unlike the specialized intelligent cameras that can perform some computation-intensive video analytical tasks, we use low-cost commodity-off-the-shelf cameras whose computation capabilities are meager. Several geo-distributed edge devices with heterogeneous computation capabilities are deployed near the cameras to reduce the data transmission cost and provide near real-time response. The edge devices collaborate to maintain a permissioned blockchain that stores the video analytics results in an immutable manner. Edge devices and cameras are interconnected within a network so that each camera can stream the videos to any edge device. Also, the computation tasks and data can be shared within the edge cluster.

Unlike cloud-based and traditional edge-based solutions, our system creates a blockchain-based shared resource pool among geo-distributed edge devices and cameras within a network. The benefits of the resource pool are as follows:

- *Efficient resource sharing and load balancing.* With the resource pool, the computation tasks can be migrated among edge devices to achieve load balancing and reduce the risk of single-node failure.
- *Flexible access to input video streams and intermediate data.* Instead of connecting one edge device with one camera, each device in the cluster can process multiple video streams, which facilitates effective data sharing.
- *Optimized scheduling to accelerate inference.* The computation tasks can be partitioned and scheduled among edge devices to reduce the execution time by jointly considering the computation and networking resources.
- *Inherent trustworthiness of video analytics results.* The edge devices contribute a small portion of computation resources to maintain a permissioned blockchain that keeps the video analytics results immutable. The results are confirmed by all the edge devices, making them trustworthy.

B. System Components

Our system enables blockchain-based intelligence among geo-distributed edge devices, which share computation resources and data to accomplish video surveillance tasks. The key is to sense the distributed edge resources and intelligently distribute the computation tasks among the resource-constraint edge devices, considering the underlying edge resources' task characteristics and status.

The system adopts a mixture of peer-to-peer and master-client architectures. More specifically, the permissioned blockchain runs in a peer-to-peer network, and the edge resource management employs the master-client architecture. The master is responsible for monitoring the edge resource status and scheduling decisions on partitioning and distributing the computation tasks among the edge devices. The critical components of the system and their roles are described below.

- *Scheduler.* The scheduler runs at the master node. It accesses the cluster status by communicating with the controller, which continuously monitors the workload and available resources of the cluster, such as network topology, data transmission rate, idle computation resources, and storage capacities of each edge device. The scheduler generates task partition and resource allocation policies with a scheduling algorithm running inside.
- *Controller.* It monitors the edge resources by communicating with the monitors on edge devices. Also, it manages the edge resource by messaging the task partition and resource allocation policies to edge devices.
- *Executor.* The executor receives the task execution policies from the controller and executes the assigned tasks.
- *Monitor.* It monitors the networking and computation status of the edge device and sends it to the controller.
- *Database.* Each edge device has its local database instead of directly sending the inference results to the master node. The local database should be synchronized periodically with the global database on the master node to facilitate distributed task execution and data sharing.
- *Permissioned blockchain.* It is maintained by edge devices. Its purpose is to store the video analytics results and keep them immutable.

The general workflow of the system is described as follows. The computation tasks of video analytics will be submitted to the scheduler, which generates the task execution policies by jointly considering the task performance metrics, available computation, storage, and networking resources of the edge devices. The policies decide how the task should be partitioned and where the task should be executed. More details will be presented in Sec. IV. The run-time characteristics of tasks

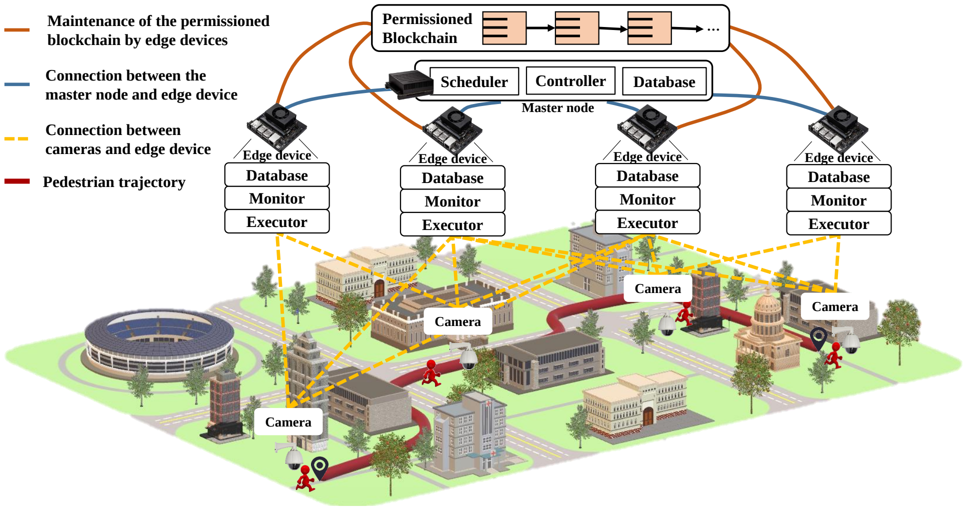


Fig. 1. System model of blockchain-based collaborative edge intelligence system for trustworthy and real-time video surveillance. The cameras are end devices generating video analytics tasks. The edge devices collaborate to perform video analytics tasks by sharing computation and data resources. Finally, the video analytics results are stored and kept trustworthy on the permissioned blockchain.

and the resource status will be sent back to the controller by the monitor and used for future decision-making. Finally, the video analytics results are kept on the permissioned blockchain.

IV. Joint Stream Mapping and Task Scheduling for Pedestrian Re-identification

We showcase the BCEI system with an edge video analytics application, i.e., pedestrian re-identification, empowered by deep learning models. We formulate a joint stream mapping and task scheduling problem to optimize the application latency, considering where to schedule the video stream and deploy the deep learning models. In this section, we first introduce the pedestrian re-identification pipeline. Then, we illustrate the benefits of jointly considering stream mapping and task scheduling. We mathematically formulate the problem and introduce optimization algorithms based on this.

A. *Pedestrian Re-identification Pipeline*

We target those applications that employ state-of-the-art neural network models to conduct various challenging video analytic tasks. Examples include object attribute recognition and object re-identification, human activity recognition, and many others [27]. Those applications typically adopt a cascaded architecture that leverages a pipeline of neural network models to accomplish the video analytic task [14]. The pipeline usually consists of an object detection model followed by task-specific models to perform various tasks on the detected objects within a video frame, e.g., object type, color, and shape.

Without loss of generality, we take the pedestrian re-identification application as an example. Pedestrian re-identification aims to associate images of the same person taken from different or the same cameras at different times. It is an essential technique in video surveillance and has been widely applied in security areas such as pedestrian tracking, criminal event detection, and children remote monitoring [21][5]. As shown in Fig. 2, a general process of pedestrian re-identification can be abstracted into a pipeline consisting of three phases, i.e., pedestrian detection, tracking, and re-identification. A detection model will first process an input image to detect the individual's location, and the identified individual is tracked by monitoring the trajectory. Afterward, discriminative features are extracted from the picture of tracked pedestrians for comparison and re-identification. The pedestrian is then annotated with the re-identification result during the later continuous tracking. The pipeline shows that the re-identification model is not frequently called because the individual will be assigned an ID and continuously tracked by the tracker once an individual is re-identified.

Fig. 2 also shows an example of distributed task execution policies. The pedestrian re-identification application is partitioned into pedestrian detection, tracking, and re-identification tasks. While the detection task and tracking are executed on edge device 1, the re-identification task is executed on edge device n. Edge device 1 sends the intermediate data to edge device n to accomplish the overall task.

We use two deep learning models for pedestrian detection and re-identification. The execution time of pedestrian tracking is much less than detection and re-identification as we do not adopt the computation-intensive deep learning (DL) models for tracking. Hence, we only consider the detection and re-identification models in the problem formulation part.

B. *Motivations of Joint Stream Mapping and Task Scheduling*

This section describes a concise example to show the motivations for joint scheduling of the video streams and deploying machine learning (ML) models on edge devices. We consider a network consisting of three edge devices that are fully connected. The application model consists of two dependent tasks, i.e., pedestrian detection and re-identification. We assume each task takes 1 unit of time to be executed on each device. Four cameras stream the videos to any edge

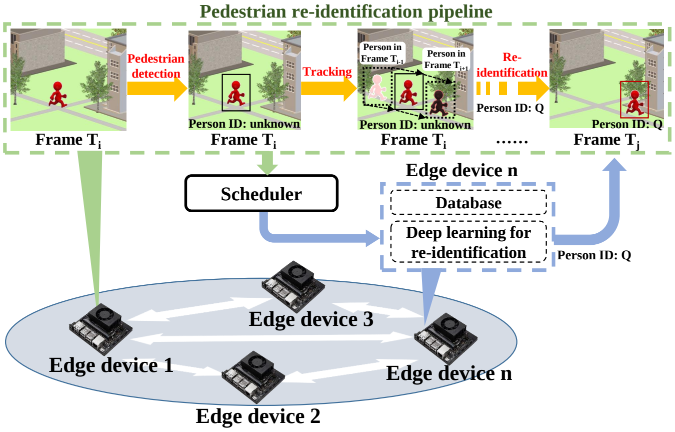


Fig. 2. Model pipeline and task offloading of pedestrian re-identification. The edge devices perform three tasks: pedestrian detection, tracking, and re-identification. The video data and analytics results are partially shared to facilitate the completion of the tasks with high performance.

device. The first scheduling problem is to decide on the edge device where each camera stream the video. Once the video is streamed to an edge device, the device will use the ML model for the detection task to detect any individual within the frame. The other ML model is to identify the detected individual for the re-identification task. More specifically, the second scheduling problem is to decide which edge device to offload the re-identification ML model. The scheduling problems' decisions are based on resource availability, bandwidth constraints, and computation workload.

The scheduling result shown in Fig. 3(a) is based on randomly deciding the mapping of video streams from cameras to edge devices and locally executing the ML models in the pedestrian re-identification application. Such random scheduling of camera streams and no offloading leads to the possibility of multiple cameras streaming the video to one overloaded edge device. In contrast, other edge devices can be lightly loaded or have no workload. In our example scenarios, three cameras (a, b, c) stream video to edge device 1, while camera d streams video to edge device 2. There is no video streamed to edge device 3. The completion time for this random scheduling is high (6 units in our example) and not optimal as it leads to inefficient usage of device resources. An improvement over the random scheduling is shown in Fig. 3(b), where we schedule the video streams from cameras while executing the ML models locally on edge devices. The scheduling of camera streams shown in Fig. 3(b) leads to a reduction in the completion time of the application from 6 to 4 units as it considers the transmission time between cameras and edge devices. Another alternative approach for improvement over the random scheduling is to make a scheduling decision on offloading the ML model depending on the resource availability instead of locally executing all the ML models as shown in Fig. 3(c). The scheduling result, shown in Fig. 3(c), also leads to a reduction in completion time from 6 to 4.5 units compared to random scheduling.

Although scheduling camera streams and offloading ML models improve the task completion time, there are still further spaces for efficiently utilizing the geo-distributed edge resources. Fig. 3(d) shows a joint approach of scheduling camera video streams and offloading the ML models, which leads to the best performance in terms of application completion time compared to the random or other scheduling approaches. In this work, we have designed an online scheduler in the proposed BCEI system to enable joint scheduling of camera streams and offloading ML models in the pedestrian re-identification application scenario. The following sections show the problem formulation and proposed solution for this joint scheduling problem in a dynamic distributed edge computing environment.

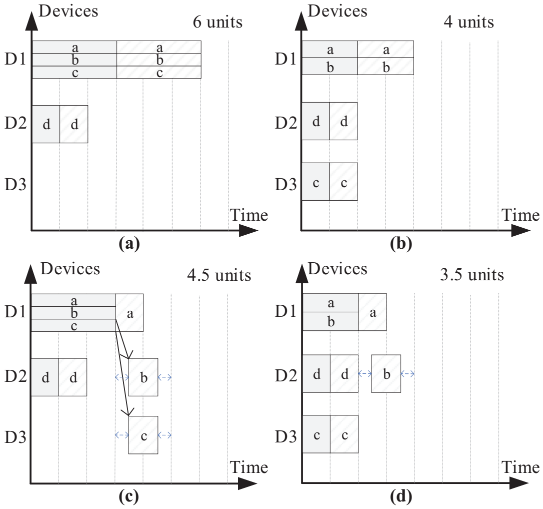


Fig. 3. A motivating example of joint stream mapping and task scheduling: (a) random camera streams and local execution in 6 units of time, (b) scheduling camera streams and local execution in 4 units of time, (c) random camera streams and offloading ML models in 4.5 units of time, and (d) joint scheduling of camera streams and offloading ML models (this work) in only 3.5 units of time.

C. Problem Formulation

Randomly mapping the camera video streams to edge devices leads to an unbalanced workload and inferior performance. We formulate a joint stream mapping and task scheduling problem to improve resource utilization and boost application performance.

We assume a quasi-static slotted time model, where it is assumed that in each time slot t, the controllers are aware of the different network and device metrics required to make the decision. The metrics for different resources are assumed to be constant in a time slot. The system consists of K cameras. Each camera can stream the video to any edge device in the cluster. The network and application model can be defined as:

Network model. The network is modelled as a graph $G = (V, E)$, where $V = \{i | 1 \leq i \leq M\}$ is the set of edge devices and $E = \{e_{ij} | i, j \in V\}$ is the set of links connecting different devices. The weight of each device is PS_i, representing the computation capacity of the device i. Each device also has a limited resource of R^i_{max}, and the available resource is indicated by R^i_{avail}. We only consider the memory in this work for the various resources of edge devices, e.g., CPU, memory, and storage. The weight of link e_{ij} represents the bandwidth between devices i and j. The devices and network links can be heterogeneous in computation and bandwidth capacities. The data rate for transmission between any two edge devices is R_{ij}. The video transmission time between camera k and edge device i is T_{ik}.

Application model. The application model for the pedestrian re-identification application studied in this work is considered a sequence of two dependent tasks, i.e., detection and re-identification. The computation load for detection and re-identification tasks corresponding to camera stream k is $CL_{1,k}$ and $CL_{2,k}$ respectively. The dependent data between the two tasks is $D^k_{1,2}$. Resource request of the detection model and re-identification model is R^{det}_{req} and R^{reid}_{req}, respectively.

Decision variables. Two decision variables correspond to mapping camera streams and scheduling DL models. The first decision variable x_{ik} is binary, equal to 1 if camera video stream k is scheduled to device i. It also indicates that the detection task is deployed on device i. Another decision variable y_{ik} is binary, which equals 1 if the re-identification task corresponding to camera video stream k is scheduled to device i.

Cost model. For each camera stream, either the detection and re-identification tasks can be locally executed, or the re-identification task can be offloaded to another device. The time for executing the models on each device depends on the overall workload and available computation capacity.

The total resource request on device i, notated as R_i, can be calculated as follows:

$$R_i = \sum_{k=1}^{K} (x_{ik} \cdot CL_{1,k} + y_{ik} \cdot CL_{2,k}) \tag{1}$$

The overall processing time for camera stream k, i.e., L_k, can be calculated as:

$$\begin{aligned} L_k = & \sum_{i=1}^{M} x_{ik} \cdot T_{ik} + \sum_{i=1}^{M} x_{ik} \cdot \frac{CL_{1,k}}{PS_i} + \sum_{j=1}^{M} y_{jk} \cdot \frac{CL_{2,k}}{PS_j} \\ & + \sum_{i=1}^{M} \sum_{j=1}^{M} x_{ik} \cdot y_{jk} \cdot \frac{D^k_{1,2}}{R_{i,j}} \cdot \sigma(i-j) \\ & \forall j \in V, \quad k \in \{1, 2, \cdots, K\} \end{aligned} \tag{2}$$

where $\sigma(\cdot)$ is an indicator function. Only when $\cdot$ is zero, $\sigma(\cdot)$ equals to 1, otherwise $\sigma(\cdot)$ equals to 0.

Objective function. The objective function of the problem is to minimize the sum of completion time for all applications from camera streams.

$$\min_{x_{ik}, y_{ik}} \sum_{k=1}^{K} L_k \tag{3}$$

Constraints:

$$R_i \leq R^i_{max}, \quad \forall i \in V \tag{4}$$

$$\sum_{i=1}^{M} x_{ik} = 1, \quad \forall j \in V, \quad k \in \{1, 2, \cdots, K\} \tag{5}$$

$$\sum_{j=1}^{M} y_{ik} = 1, \quad \forall i \in V, \quad k \in \{1, 2, \cdots, K\} \tag{6}$$

$$x_{ik} = \{0, 1\}, \quad \forall i, j \in V, \quad k \in \{1, 2, \cdots, K\} \tag{7}$$

$$y_{ik} = \{0, 1\}, \quad \forall i, j \in V, \quad k \in \{1, 2, \cdots, K\} \tag{8}$$

Eq. 4 indicates that the resource request on an edge device cannot exceed its maximum resource. Eq. 5 and Eq. 6 show that the detection model and the re-identification model of a stream can only be deployed on one edge device. The formulated optimization problem is nonlinear integer programming (NLP) problem. The problem is NP-hard because the offloading problem can be reduced to a generalized assignment problem, proven to be NP-hard in literature.

D. Optimization Solution

We have proposed a joint stream mapping and task scheduling heuristic algorithm (JSTSH) that determines where to schedule each video stream and which edge device to allocate the detection and the re-identification task. The algorithm is developed with two fundamental principles as follows.

- *High workload first.* Generally, the latency of handling video streams with high workloads is much larger than those with moderate workloads. Suppose we first allocate edge resources to the video streams with a moderate workload. In that case, those video streams with a high workload may suffer from prolonged latency when the rest of the edge resources are inadequate, further leading to increased average latency of all video streams. Hence, we leverage a priority list to rank all video streams according to their workloads.
- *Reusing re-identification model.* Following the pedestrian detection task, a re-identification model is used to perform the re-identification task. In our pedestrian re-identification pipeline, the re-identification model is not frequently called because an individual will be continuously tracked once the individual's ID is determined. For each video stream, if we jointly consider the deployment of the detection and re-identification models, it will lead to high resource consumption. Hence, we consider the deployment of the two models separately. When the available resources are constrained, the re-identification model can be reused, which means a re-identification model can handle multiple video streams.

Based on the two principles, we solve the problem in two stages. The first stage is to schedule the video stream and the detection task. The second stage is to schedule the re-identification task. We first determine the priority of stream scheduling by sorting the video streams according to their workloads. Then in the first stage, we filter the candidate edge devices with abundant resources to allocate the detection model. The video stream is allocated to the edge device with minimum execution time, including the raw video transmission time and the inference time of the detection model.

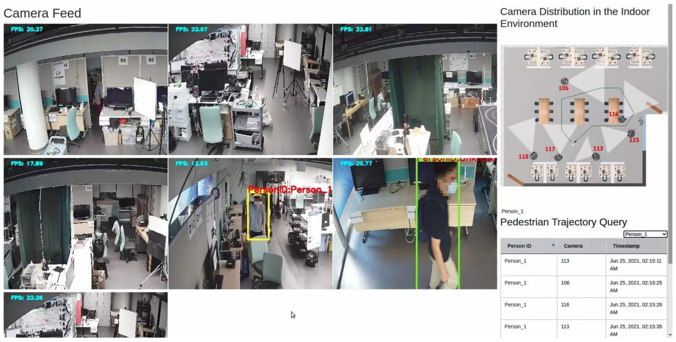
$$\min_{i} (x_{ik} \cdot \frac{D^k_{1,2}}{R_{i,j}} + \frac{CL_{2,k}}{PS_i}), \quad i \in M_{deployed}, k \in V_{rest} \tag{9}$$

After determining x_{ik}, we then allocate the re-identification models in stage 2. We use a similar greedy idea to allocate

Algorithm 1: Joint Stream Mapping and Tasks Scheduling Heuristic (JSTSH)

```
Input: Video stream $V = \{k_i\}_{i=1}^{K}$, computation
       capacity $\{PS_i\}_{i=1}^{M}$ and available resource
       $\{R_{avail}^{j}\}_{i=1}^{M}$
Output: Video and task allocation policy $x_{ik}$ and $y_{ik}$
Create index $I$ of streams in descending order of
  workload;
for $t \leftarrow 1$ to $K$ do // Stage 1
    $k \leftarrow I(t)$;
    for each device $i \leftarrow 1$ to $M$ do
        if $R_{avail}^{j} > R_{req}^{det}$ then
            Calculate the execution time
              $t_{exec}^{i} = T_{ik} + \frac{CL_{1,k}}{PS_i}$;
        end
    end
    Calculate device $i^*$ with the shortest execution
      time $i^* = min_i\{t_{exec}^{j}\}$;
    $x_{i^*,k} \leftarrow 1$, update $R_{i*}$ for device $i^*$;
end
for $t \leftarrow 1$ to $K$ do // Stage 2
    $k \leftarrow I(t)$;
    for each device $i \leftarrow 1$ to $M$ do
        if $R_{avail}^{j} > R_{req}^{reid}$ then
            Calculate the intermidiate data transmission
              time $t_{comm}^{i} = x_{ik} \cdot \frac{D_{1,2}^{k}}{R_{i,j}}$;
            Calcualte the inference time of detection
              model $T_{reid}^{i} = \frac{CL_{2,k}}{PS_i}$;
            Calculate the execution time
              $t_{exec}^{i} = t_{comm}^{i} + t_{reid}^{i}$;
        end
    end
    Calculate device $i^*$ with the shortest execution
      time $i^* = min_i\{t_{exec}^{j}\}$;
    Add $i$ to candidate list $M_deployed$;
    if $V_{rest} \neq \emptyset$ then
        Schedule remaining streams to the device that
          satisfies Eq. 9;
    end
    Update $R_{i*}$ for device $i^*$;
    $y_{i^*,k} \leftarrow 1$;
end
return $x_{ik}$, $y_{ik}$, $i = 1, 2, \cdots, M$, $k = 1, 2, \cdots, K$
```

the re-identification models as stage 1 does. The difference is that we consider the scenario that idle computation resources may be less abundant for deploying a re-identification model for each stream. Hence, we reuse the re-identification model, where a deployed model can serve multiple streams. To determine where to deploy the re-identification model, we allocate the re-identification models for those streams with high workloads. When the available resources cannot support the deployment of new re-identification models, we reuse the re-identification model. The rest of the streams V_{rest} will be allocated to the edge device, which satisfies Eq. 9. By solving Eq. 9, the rest of the streams will be scheduled to the deployed re-identification model that can provide the least intermediate data transmission and inference time.



Pedestrian Trajectory Query

Person ID	Camera	Timestamp
Person_1	113	Jun 25, 2021, 02:15:11 AM
Person_1	106	Jun 25, 2021, 02:15:25 AM
Person_1	116	Jun 25, 2021, 02:15:26 AM
Person_1	113	Jun 25, 2021, 02:15:35 AM

Fig. 4. Demo of pedestrian re-identification. The captured images of the seven cameras are shown on the left. The upper right corner illustrates the positions of the seven cameras in the environment. The bottom right corner presents the video analytics results.

V. Implementation and Performance Evaluation

This section presents the system implementation, performance metrics, and experimental results of the BCEI platform.

A. System Implementation

Testbed. We deploy the system in an indoor environment due to the privacy regulations of the campus. The floor is shown in the upper right of Fig. 4, with the size $30 \times 15m^2$. We implement the system with 7 IP cameras, 5 edge devices, and 1 master device. We use 3 Jetson Xavier NX and 2 Jetson Tx2 as the edge devices. The Jetson Xavier NX has a 384-core Volta GPU with 8GB RAM, and the Jetson Tx2 has a 256-core GPU with 8GB RAM. The computation capacity of the former device is much stronger than the latter. Also, to emulate the heterogeneous resource, we constrain the memory of Jetson Tx2 as 4GB. We use a powerful workstation with four Intel Cores i9-7100U with 32 GB RAM to emulate the master. Three routers connect the edge devices and cameras. To avoid single node failure, we use a backup server to synchronize the status of the master node and recover the system when the master node goes offline. To emulate the distance between the cameras and the edge devices, we use the Linux traffic control to configure the bandwidth between the cameras, edge devices, and edge devices. The edge devices maintain a permissioned blockchain of Hyperledger Fabric [28] to store the video analytics results. Note that the performance of the permissioned blockchain is evaluated in many other studies [29], [30] and not considered in this work.

Pedestrian re-identification pipeline. In the detection part, we use SSD-MobileNet-V2 [31] to process the input image and locate pedestrians. SSD-MobileNet-V2 is a lightweight deep convolution network that uses both depth-wise and point-wise convolutions to decrease model complexity, i.e., the number of parameters and operations. This work uses Kalman filter-based tracker due to its low computation cost compared with DL-based approaches. Our target is to extract discriminative features for re-identification. Considering the constrained resources of edge devices, we build the feature extractor network with OSNet-AIN [32], which is light-weighted compared to ResNet-50 [33] and DenseNet [34]. It can learn the global representation of the individual's appearance and capture the subtle details required for the re-identification of individuals. We finetune the pretrained models of SSD-MobileNet-V2 and OSNet-AIN. All the models are developed following serverless principles and called Restful APIs [35].

We test the accuracy of the deep learning models. We run the system for a whole workday, capturing around 346 pedestrians. Among these, 270 pedestrians are correctly re-identified with an overall accuracy of 78.3%. There are many false positives, i.e., different pedestrians are not differentiated and thus identified with the same identity, which is caused by differences in lighting and camera angles. More advanced approaches can be applied to improve accuracy. However, it is not the focus of this work.

Database. We use SQLite to record both the spatial and temporal information, the discriminative pedestrian features, and the video analytics results. The SQLite is a lightweight database often used in resource-constrained devices, such as mobile phones, cameras, and home electronic devices. Each edge device has its local database instead of directly sending the inference results to the master node, which is usually vulnerable to an unstable network connection. The local database should be synchronized periodically with the global database on the master node to facilitate collaborative inference and pedestrian tracing. The synchronization period is set to 10 seconds in our system. For example, video analytics results are still in edge devices when the data is not synchronized. The master will forward the query request to edge devices and get the latest status from edge devices.

Fig. 4 shows the system interface. The camera views are on the left side of the interface, where we can monitor pedestrian behavior in real time. Authorized users can also easily query the pedestrian trajectory, as shown on the right.

B. Evaluation Metrics and Experimental Settings

We use the metrics of *system throughput* and *latency* to evaluate the performance of the proposed approach.

- *System throughput.* The cameras continuously stream the input videos to the edge cluster, in which high system throughput is required to process these incoming video streams in real-time. We consider the system throughput the average frames per second (FPS).
- *Latency.* Live video analytics applications require producing analytics results within a short period. The task execution time for a stream includes video transmission time, local execution time, task offloading time, and remote execution time. Since there is multiple input video stream, we consider the latency as the average task execution time for all streams.

The system schedules the video analytics tasks *with mapping, with offloading (WMWO)*. We evaluate the system performance under different metrics and compare it against several benchmark solutions.

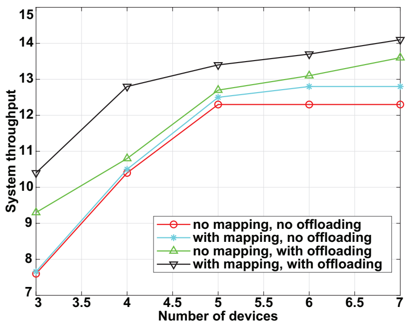


Fig. 5. System throughput vs. number of edge devices. Regardless of the employed approaches, the system throughput increases and the increasing speed gets lower with more edge devices. WMWO proposed in this work outperforms the other approaches.

- *No mapping, no offloading (NMNO).* The input video streams are randomly assigned to the edge devices, and all the computation tasks are executed locally.
- *With mapping, no offloading (WMNO).* In this case, the input video streams are assigned to the edge devices with the shortest video transmission time, and all the computation tasks are executed locally.
- *No mapping, with offloading (NMWO).* In this case, the input video streams are randomly assigned to the edge devices, and the re-identification task can be offloaded among the edge cluster.

NMNO and WMNO are non-offloading methods, and NMWO and WMWO are offloading methods. We test the performance of the proposed method and the baselines under various situations, i.e., the number of edge devices, the number of pedestrians in each input video stream, and different bandwidths of network links.

C. Influence of Number of Edge Devices

The number of the input video stream is set to be 5. We increase the number of edge devices from 3 to 7. As shown in Fig. 5, we can see that the proposed method, i.e., WMWO, achieves the highest system throughput with the variation of the number of edge devices. When there are more edge devices than the input video streams, the NMNO method keeps the system throughput consistent as it cannot utilize the computation resources of other edge devices. The WMNO achieves a slightly high system throughput than NMNO as it maps the input video streams to edge devices, considering the heterogeneous computation capacities of the edge devices. Compared with WMNO, NMWO does not schedule the input video streams. However, it dynamically offloads the re-identification tasks, which can alleviate the computation workload on a single edge device and leverage the heterogeneous computation capacity of edge devices to improve the average system throughput.

Though WMNO and NMWO can achieve higher system throughput by either optimizing the stream mapping decision or the task offloading decision, their performance is not better than WMWO as WMWO jointly considers stream mapping and task offloading. WMWO achieves 14%-36% system throughput improvement compared with baseline methods. It

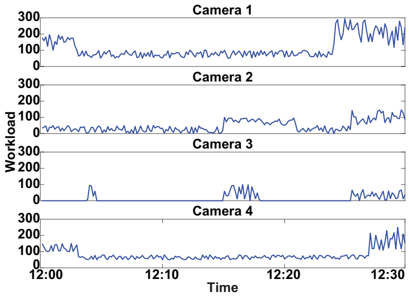


Fig. 6. Workload dynamics of four cameras in real-world deployment. We can observe that the workload is not evenly distributed among different cameras.

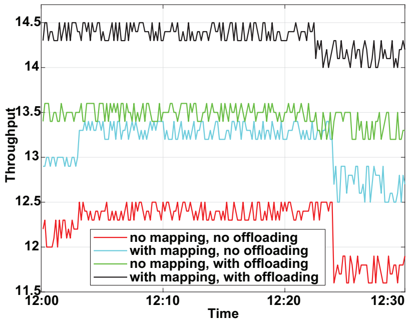


Fig. 7. System throughput vs. dynamic workload. Regardless of the employed methods, the system throughput slightly fluctuates (within 1 FPS).

shows that our proposed method can integrate the heterogeneous resources of geo-distributed edge devices and improve application performance by jointly optimizing the mapping and offloading decisions to improve the system throughput.

D. Influence of Dynamic Workload

We study the performance of the proposed method and various benchmarks under dynamical workloads with a varying number of pedestrians in the input video streams. The number of video streams and edge devices is set as 4 and 5, respectively. Fig. 6 shows the workloads generated from the 4 cameras on a weekday between 12:00 to 12:30. The workload of each video stream varies with the number of pedestrians in a video stream and is dynamic as the content captured by each camera changes over time. We can also see that the workloads are different across cameras. In particular, cameras 1 and 4 have a higher average workload than the other two cameras as they capture people entering or leaving out the doors.

As shown in Fig. 7, while the system throughput of NMNO and WMNO fluctuates a lot over time, there is no noticeable change in offloading methods, i.e., system throughput for WMNO and NMWO. NMNO and WMNO show apparent performance degradation when there is a high workload because edge devices with constrained resources may easily get overloaded, further leading to the deterioration of the average system throughput. Offloading methods show consistent performance because they leverage resource sharing and schedule the dynamic workloads among edge devices. Similar trends are also observed in terms of the latency of the application. From Fig. 8, we can also see that the latency of offloading the task is similar to that of without offloading when there is a low workload between 12:05 to 12:25, which is due to the data transmission latency of offloading the re-ID task and getting back the re-identification results. When there is a low workload, the data transmission latency is the main factor that constrains the end-to-end latency of offloading methods. As the workload increases, the transmission delay of intermediate data and model inference delay will increase. However, the inference latency caused by overloaded edge devices becomes the main factor in this case. In this case, offloading methods show apparent superiority.

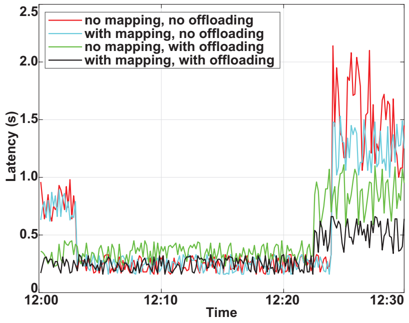


Fig. 8. Latency vs. dynamic workload. In the case of a moderated workload, the latency achieved by different methods is similar. In the case of a heavy workload, WMWO proposed in this work achieves much lower latency.

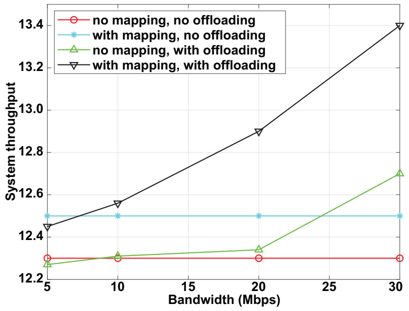


Fig. 9. System throughput vs. network bandwidth. With a higher network bandwidth, the approaches of WMNO and WMWO can achieve higher system throughput, while the approaches of NMNO and NMWO cannot.

E. Influence of Network Bandwidth

We also investigate the performance of the system in different bandwidth conditions. Specifically, we leverage the Linux traffic control to manually set the bandwidth of each link and study the system throughput under the average bandwidth of 5Mbps, 10Mbps, 20Mbps, and 30Mbps, respectively. The variance of bandwidth of network links is 30%. For example, we increase the average bandwidth from 10 to 20 by doubling the bandwidth of each link in the network.

It can be seen from Fig. 9 that while the system throughput of non-offloading methods keeps consistent, the performance of offloading methods degrades with a decreasing bandwidth due to the increased transmission delay. When the average bandwidth is 30Mbps, the system throughput of offloading methods is much better than that of the non-offloading methods. When the average bandwidths are 5Mbps or 10Mbps, the performance of offloading methods is similar to no-offloading methods. We find that the performance of offloading

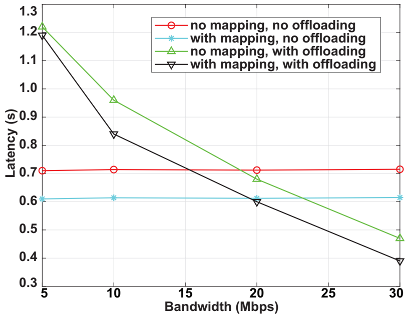


Fig. 10. Latency vs. network bandwidth. With a higher network bandwidth, the approaches of WMNO and WMWO can achieve much lower latency while the approaches of NMNO and NMWO cannot.

methods is highly affected by the transmission delay, as shown in Fig. 10. When there is a long transmission delay, the performance of offloading methods is similar to non-offloading methods, which means that the optimal policy in such cases is to execute the re-identification tasks locally, i.e., no-offloading.

The experimental results indicate that our proposed BCEI approach can dramatically improve the real-time performance of video analytics applications, especially when the resources of edge devices are constrained. BCEI also shows superior performance in handling dynamic workloads as it integrates the geo-distributed resources and intelligently schedules the video streams and the inference tasks.

VI. Conclusion

This work proposes a trustworthy and real-time video surveillance system with blockchain-based collaborative edge intelligence. We deploy the edge devices closer to the cameras and create a distributed edge devices cluster where computation and data resources can share within the cluster. We design a scheduler that jointly schedules the camera streams to edge devices and offload tasks in the application pipeline to improve resource utilization and performance. We have tested the efficacy of the proposed solution with a pedestrian re-identification on a real-world prototype. Extensive experiments show our proposed BCEI dramatically improve the real-time performance of video analytics applications under constraint edge resources and dynamic workloads.

VII. Acknowledgement

The work was supported by the Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University, and HK RGC General Research Fund No. PolyU 15220020 and No. PolyU 15204921.

References

[1] T. Zhang, S. Liu, C. Xu, and H. Lu, “Mining Semantic Context Information for Intelligent Video Surveillance of Traffic Scenes,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 149–160, 2012.

[2] R. Nawaratne, D. Alahakoon, D. De Silva, and X. Yu, “Spatiotemporal Anomaly Detection using Deep Learning for Real-time Video Surveillance,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 393–402, 2019.

[3] S. Wan, S. Ding, and C. Chen, “Edge Computing enabled Video Segmentation for Real-time Traffic Monitoring in Internet of Vehicles,” *Pattern Recognition*, vol. 121, p. 108146, 2022.

[4] S. Y. Nikouei, Y. Chen, A. J. Aved, and E. Blasch, “I-ViSE: Interactive Video Surveillance as an Edge Service using Unsupervised Feature Queries,” *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 16 181–16 190, 2020.

[5] A. Bedagkar-Gala and S. K. Shah, “A Survey of Approaches and Trends in Person Re-identification,” *Image and Vision Computing*, vol. 32, no. 4, pp. 270–286, 2014.

[6] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, “NoScope: Optimizing Neural Network Queries over Video at Scale,” *Proceedings of the VLDB Endowment*, vol. 10, no. 11, p. 1586–1597, 2017.

[7] M. Zhang, F. Wang, Y. Zhu, J. Liu, and Z. Wang, “Towards Cloud-edge Collaborative Online Video Analytics with Fine-grained Serverless Pipelines,” in *ACM Multimedia Systems Conference*, 2021, pp. 80–93.

[8] P. M. Grulich and F. Nawab, “Collaborative Edge and Cloud Neural Networks for Real-time Video Processing,” *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 2046–2049, 2018.

[9] S. Jiang, J. Cao, J. A. McCann, Y. Yang, Y. Liu, X. Wang, and Y. Deng, “Privacy-preserving and Efficient Multi-keyword Search over Encrypted Data on Blockchain,” in *IEEE International Conference on Blockchain*. IEEE, 2019, pp. 405–410.

[10] H. Du, L. Chen, J. Qian, J. Hou, T. Jung, and X.-Y. Li, “PatronuS: A System for Privacy-Preserving Cloud Video Surveillance,” *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1252–1261, 2020.

[11] Y. Sahni, J. Cao, S. Zhang, and L. Yang, “Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things,” *IEEE Access*, vol. 5, pp. 16 441–16 458, 2017.

[12] 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 *IEEE/ACM Symposium on Edge Computing*, 2020, pp. 110–124.

[13] Z. Xu, S. Sinha, S. Harshil S., and U. Ramachandran, “Space-Time Vehicle Tracking at the Edge of the Network,” in *The 3rd Workshop on Hot Topics in Video Analytics and Intelligent Edges*, 2019, pp. 15–20.

[14] X. Zeng, B. Fang, H. Shen, and M. Zhang, “Distream: Scaling Live Video Analytics With Workload-adaptive Distributed Edge Intelligence,” in *ACM Conference on Embedded Networked Sensor Systems*, 2020, pp. 409–421.

[15] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, “VideoEdge: Processing Camera Streams using Hierarchical Clusters,” in *IEEE/ACM Symposium on Edge Computing*, 2018, pp. 115–131.

[16] J. Zhang, M. Z. A. Bhuiyan, X. Yang, A. K. Singh, D. F. Hsu, and E. Luo, “Trustworthy Target Tracking with Collaborative Deep Reinforcement Learning in EdgeAI-Aided IoT,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1301–1309, 2021.

[17] J. Zhang, M. Z. A. Bhuiyan, X. Yang, T. Wang, X. Xu, T. Hayajneh, and F. Khan, “AntiConcealer: Reliable Detection of Adversary Concealed Behaviors in EdgeAI Assisted IoT,” *IEEE Internet of Things Journal*, 2021.

[18] Y. Chen, T. Yang, C. Li, and Y. Zhang, “A Binarized Segmented ResNet Based on Edge Computing for Re-Identification,” *Sensors*, vol. 20, no. 23, p. 6902, 2020.

[19] S. Jiang, J. Cao, H. Wu, and Y. Yang, “Fairness-based Packing of Industrial IoT Data in Permissioned Blockchains,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7639–7649, 2020.

[20] 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.

[21] C. Neff, M. Mendieta, S. Mohan, M. Baharani, S. Rogers, and H. Tabkhi, “REVAMP^{2}T: Real-Time Edge Video Analytics for Multicamera Privacy-Aware Pedestrian Tracking,” *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2591–2602, 2019.

[22] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, “Lavea: Latency-aware Video Analytics on Edge Computing Platform,” in *ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1–13.

[23] 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 *USENIX Symposium on Networked Systems Design and Implementation*, 2017, pp. 377–392.

[24] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, “Chameleon: Scalable Adaptation of Video Analytics,” in *Annual Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 253–266.

[25] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez, “Scaling Video Analytics Systems to Large Camera Deployments,” in

International Workshop on Mobile Computing Systems and Applications, 2019, pp. 9–14.

[26] J. Jiang, Y. Zhou, G. Ananthanarayanan, Y. Shu, and A. A. Chien, "Networked Cameras are the New Big Data Clusters," in *The 3rd Workshop on Hot Topics in Video Analytics and Intelligent Edges*, 2019, pp. 1–7.

[27] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang, "Elf: Accelerate High-resolution Mobile Deep Vision with Content-aware Parallel Offloading," in *Annual International Conference on Mobile Computing and Networking*, 2021, pp. 201–214.

[28] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains," in *ACM European Conference on Computer Systems*, 2018, pp. 1–15.

[29] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A Framework for Analyzing Private Blockchains," in *ACM International Conference on Management of Data*, 2017, pp. 1085–1100.

[30] M. Dabbagh, K.-K. R. Choo, A. Beheshti, M. Tahir, and N. S. Safa, "A Survey of Empirical Performance Evaluation of Permissioned Blockchain Platforms: Challenges and Opportunities," *Computers & Security*, vol. 100, p. 102078, 2021.

[31] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-arithmetic-only Inference," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

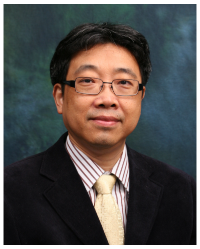
[32] J. Almazan, B. Gajic, N. Murray, and D. Larlus, "Re-ID Done Right: Towards Good Practices for Person Re-identification," *CoRR*, vol. abs/1801.05339, 2018.

[33] A. Hermans, L. Beyer, and B. Leibe, "In Defense of the Triplet Loss for Person Re-Identification," *CoRR*, vol. abs/1703.07737, 2017.

[34] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[35] Y. Chen, X. Xu, and W. Wang, "Efficient Web APIs Recommendation with Privacy-preservation for Mobile App Development in Industry 4.0," *IEEE Transactions on Industrial Informatics*, 2021.

Mingjin Zhang is currently a Ph.D. candidate with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China. He received the B.Eng. degree in communication engineering from Wuhan University of Technology, China, in 2019. His research interests include edge computing, edge AI, distributed machine learning, and Internet of Things.



Jiannong Cao is currently the Otto Poon Charitable Foundation Professor in Data Science and the Chair Professor of Distributed and Mobile Computing in the Department of Computing at The Hong Kong Polytechnic University (PolyU), Hong Kong. He is also the Dean of Graduate School, the director of the Research Institute for Artificial Intelligence of Things in PolyU, and the director of the Internet and Mobile Computing Lab. He was the founding director and is now the associate director of PolyU's University Research Facility in Big Data Analytics. He served as the department head from 2011 to 2017. Prof. Cao is a member of Academia Europaea, a fellow of IEEE, a fellow of the China Computer Federation (CCF), and an ACM distinguished member. His research interests include distributed systems and blockchain, wireless sensing and networking, big data and machine learning, and mobile cloud and edge computing.

Yuvraj Sahni received the B.Eng. (Hons) degree in Electrical and Electronics Engineering from Birla Institute of Technology and Science, Pilani, India, in 2015 and the Ph.D. degree from the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, in 2021. He is currently a Research Assistant Professor at the Department of Building Environment and Energy Engineering, The Hong Kong Polytechnic University, Hong Kong. Before that, he was a Postdoctoral Fellow at the Department of Computing, The Hong Kong Polytechnic University, from April 2021 to March 2022. His research interests include Edge Computing, Edge AI, Internet of Things, and Smart Buildings.

Qianyi Chen is currently a Ph.D. candidate in the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China. Before that, he received an M.Sc. (2019) degree in bridge and tunnel engineering from Zhejiang University and a B.Sc. (2016) degree in road and bridge engineering from Huazhong University of Science and Technology. His research interests include physics-enhanced machine learning, edge computing, and structural health monitoring.

Shan Jiang received a B.Sc. degree in computer science and technology from Sun Yat-sen University, Guangzhou, China, in 2015 and a Ph.D. degree in computer science from The Hong Kong Polytechnic University, Hong Kong, in 2021. He is currently a Research Assistant Professor in the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. Before that, he visited Imperial College London from November 2018 to March 2019. He won the best paper award of BlockSys 2021. His research interests include distributed systems and blockchain, blockchain-based big data sharing, and blockchain as a service.

Lei Yang is currently an associate professor at the School of Software Engineering, South China University of Technology, China. He received the B.Sc. degree from Wuhan University, in 2007, the M.Sc. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2010, and the Ph.D. degree from the Department of Computing, The Hong Kong Polytechnic University, in 2014. He has been a visiting scholar at Technique University Darmstadt, Germany from Nov 2012 to Mar 2013. His research interests include edge and cloud computing, distributed machine learning, and scheduling and optimization theories and techniques.