EdgeEye: A Data-driven Approach for Optimal Deployment of Edge Video Analytics

Hui Sun, Ying Yu, Kewei Sha, *Senior Member, IEEE*, Hong Zhong

Abstract—Deep neural network (DNN) based video processing methods are applied in mobile video analytics because of high accuracy. Edge computing is an efficient paradigm that improves the performance of mobile video analytics. However, due to the limited computing and storage resources at edge devices, deploying DNN-based video analytics at edge devices may have difficulty to meet user's requirements in terms of accuracy, delay, power consumption, and device costs, etc. Choosing optimal system configuration including resources on edge devices and parameters in video stream and DNN models can better satisfy user's performance requirements; however there lacks practical approaches to find such optimal configurations. In this paper, we take an initial step to investigate the optimal system configuration problem, and propose a data-driven approach, EdgeEye, which first models the above problem as a combinatorial optimization problem, and then designs an algorithm to find the solutions for the optimal configuration. These models and algorithms are applied in a real-world face detection and recognition application based on two edge computing models, including edge-only and edge-server. Comprehensive evaluation results show that EdgeEye can find both feasible configurations and optimal system configurations including optimal edge computing model to satisfy application user's different requirements under different network conditions.

Index Terms—Edge Computing, Optimization, Edge Video Analytics, System Configuration Solutions, Data-driven Approach

1 Introduction

In the Internet of Things era, large-scale video data has been transmitted to the cloud center for video analysis. This cloud computing based video analytics imposes a heavy burden on the network and increases delay, which makes it difficult to meet user's delay requirements [1] [2]. Edge computing is proposed as a new computing paradigm [3] to improve the performance of video analytics. Edge video analytics [4] [5] has been widely used in industry automation [6] [7] [8], smart campus [9], and smart connected vehicles [10] [11]. Two edge computing models, edge-only and edge-server, are widely employed in video analytics. In edge-only model, video analytics is conducted only on edge devices that are co-located with cameras. The edge-server model collaboratively performs video analytics among edge devices and edge servers/cloud servers.

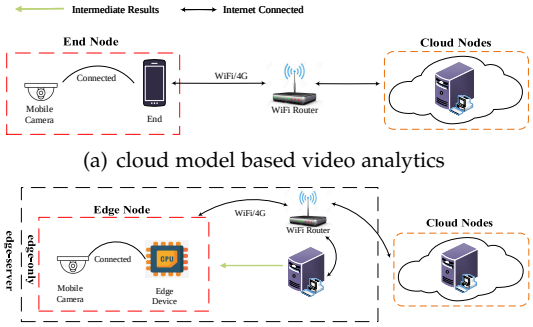
Deep neural network (DNN) models are applied in video analytics [12] [13] because of high accuracy. These applications include face recognition [14], bio-metric identification [15], and gait authentication [16]. Deploying DNN models on edge devices to perform video analytics can reduce the network overhead and the delay. However, due to limited resources at edge devices and the instability of the wireless connection, DNN models can cost a large amount of local resources to process high definition or full high definition video data; meanwhile, the latency of video analytics increases. Many efforts have been made to improve this process. For example, a DNN inference engine is implemented on Application Specific Integrated Circuit (ASIC) to improve the latency [17], but it cannot be widely applied because of low reconfigurability. An industrial computing platform based on FPGA is designed to reconstruct DNN models for video analytics [18] [19] [20]. However, this method is expensive for large-scale video systems, and it has a poor mobility. Besides, the DNN model compression is also proposed to reduce the latency at the edge but it hurts accuracy.

Based on the results from our previous face detection and recognition study [21], we observe that many factors, including the choice of DNN models, video stream settings (HD and FHD), configuration of available resources and costs of edge devices (ASIC and FPGA), can impact the system performance of mobile video analytics, such as accuracy, delay, and power consumption, which aim to meet user's requirements. It is important for system designers to obtain an appropriate combinatorial configurations (*e.g.*, parameters in the DNN models, parameters in the video stream, and edge device resources) to optimize the overall system performance while satisfying user's requirements, which may vary in different applications. For example, a license plate recognition task at the road intersection requires high accuracy but can tolerate a few seconds delay. However, vehicle counting application works with moderate accuracy but needs low delay for controlling the traffic lights. In addition, in edge computing, the choice of computing model will also have an impact on the performance of video analytics. In our work, we mainly exploit two edge computing models, edge-only and edge server, as examples to study the impact of computing models.

When finding both feasible configurations and optimal configurations for deploying the DNN based video

- *Hui Sun, Ying Yu, and Hong Zhong are with the School of Computer and Science, Anhui University, Hefei, China. E-mail: sunhui@ahu.edu, zhongh@ahu.edu, and cnyingyu@stu.ahu.edu.*
- *Kewei Sha is with the department of Computer Science at University of Houston - Clear Lake (UHCL), Houston, TX, USA. E-mail: sha@uhcl.edu.*

Manuscript received xxxx; revised xxxx.



(a) cloud model based video analytics

(b) Two edge models (edge-only and edge-server) based video analytics

Fig. 1. The cloud model and the edge model based video analytics system.

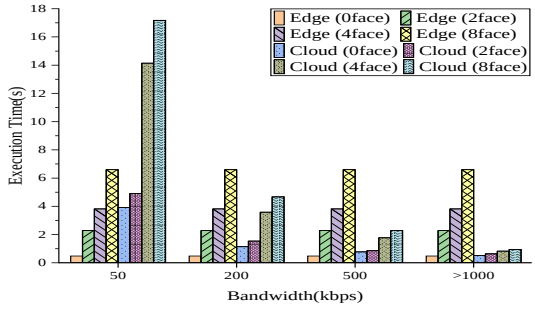


Fig. 2. The execution time of face detection and recognition on Raspberry Pi 3B under different network conditions using cloud and edge models.

analytics, we aim to answer the following three research questions. First, how can we select the right performance metrics for deploying video analytics and independent variables related to these performance metrics? Second, as edge devices have heterogeneous resources, how can we choose appropriable edge devices together with proper edge computing models for video analytics? Third, because both unstable wireless connection and inexpensive low-capable edge devices increase the delay, how can we trade off the delay and the costs of device and network bandwidth?

To tackle the above challenges, we propose a data-driven approach named EdgeEye, which first formulates the optimal configuration problem of video analytics as a combinatorial optimization problem (COP), having the objective of maximizing the system performance under the constraints of available resources while satisfying the user's requirements. EdgeEye also designs an efficient algorithm to find the solution of the COP. Thus, EdgeEye can choose optimal edge device, parameters in the DNN models, and parameters in the video stream to meet the performance and cost requirements.

Contributions of our work are summarized as follows.

◇ Based on our knowledge, we are the pioneer to formally model the relationship between performance metrics and the related independent variables. The proposed EdgeEye approach can be generalized to find optimal solution for various optimization objectives, which can be defined based on application's requirements.

◇ We apply EdgeEye in a face detection and recognition application to find optimal system configurations in the context of edge-only and edge-server models. This demonstrates an excellent example of applying EdgeEye in edge computing based system design.

◇ We build an EdgeEye supported video analytics system prototype to validate the efficiency of EdgeEye. Comprehensive performance evaluation shows that EdgeEye can provide optimal configurations to satisfy the user's requirements of deploying face detection and recognition.

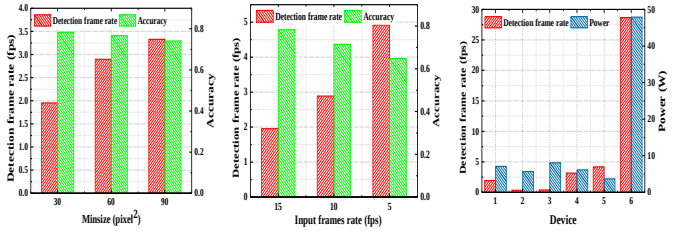
The rest of the paper is organized as follows. Section 2 introduces the background and motivation. In Section 3, we describe the EdgeEye approach in detail. Section 4 shows the experimental setup to study benefits of EdgeEye in video analytics deployment. In Section 5, we take face detection and recognition as an example to validate that EdgeEye can help designers achieve optimal configurations for video analytics. We present the related work in Section 6. We conclude the paper in Section 7.

2 Background and Motivation

Face detection and recognition, as one of the important technologies for identification and authentication, has been widely used in public safety [5]. For example, the multi-task cascaded convolutional network (MTCNN) [22] and FaceNet [23] are applied in face detection and recognition. Generally, face detection and recognition is employed in video analytics systems based on the cloud model and the edge model [3]. Fig. 1(a) illustrates the cloud model, where the device captures video data and sends it to the cloud center for face detection and recognition. As shown in Fig. 1(b), in edge computing model, face detection and recognition can all be performed at the edge device. We name it as edge-only model. Edge computing model can also be extended to edge-server model, where edge device performs MTCNN based face detection and edge server performs FaceNet based face recognition. Different edge computing models exhibits different performance.

Fig. 2 presents the execution time of face detection and recognition using the edge-only model and cloud model. The experiment is conducted with a video stream of 640 × 480 solution under four types of network bandwidth, and each frame in the video contains 0, 2, 4, and 8 faces, respectively. In Fig. 2, the execution time keeps stable using edge-only model under different network conditions, while it is dynamic using cloud model. When the bandwidth is higher than 1 Mbps, the cloud model outperforms the edge-only model for videos with two or more faces, but using this model, the low-bandwidth network can cause high delay. Thus, in the case of limited network resources, the edge-only model has the edge over the cloud model in the DNN based video analytics. We also observe that the delay increases with the increase of the number of processed faces and resource-constrained edge devices like Raspberry Pi significantly increases the delay.

To further study the impact of the above parameters and settings on system performance, we conduct experiments on face detection using MTCNN and recognition using FaceNet in the edge-only model. Without loss of generality, we use six edge devices with different level of computing power (see Section 4). We configure three types of the minimum



(a) (b) (c)

Fig. 3. The impact of the minimum detectable face size, the input frame rate, and the device with different resources on the performance of face detection and recognition.

detectable face size (30 pixel^2, 60 pixel^2, and 90 pixel^2) in MTCNN model and three types of input frame rate (15 fps, 10 fps, and 5 fps) for video detection to study the impact of these parameters on the performance in terms of detection frame rate[1], accuracy, and power consumption of the edge device.

Fig. 3 demonstrates the relationship between the detection frame rate, the detection accuracy and the minimum detectable face size, the input frame rate, and devices. We have the following observations. First, when the MTCNN model is deduced, the small detectable face size in the MTCNN model results in a high accuracy for face detection, but it also leads to low detection frame rate. Second, the small input frame rate can increase the detection frame rate, but it impairs the accuracy. Third, using powerful edge devices can reduce delay while maintain accuracy, but it may fail to meet user's requirements in terms of power consumption and device costs.

Based on the above observations, in order to find the optimal configurations to balance the performance and different user requirements, we need to formulate three types of relationship including (1) the minimum detectable face size vs. the face detection accuracy, delay, and power consumption; (2) the input frame rate of the video stream vs. delay, and accuracy; (3) computing resources at edge devices vs. accuracy, delay, and power consumption.

3 The Design of EdgeEye

In a video analytics system, system designers can choose different edge devices and configure various parameters in video stream and DNN models to meet users' requirements. However, it is challenging to determine the optimal configurations for DNN models, video stream, and edge devices to balance the delay, accuracy, and power consumption in video analytics. We propose a data-driven optimization approach named EdgeEye to study the feasible optimal configurations for video analytics. EdgeEye consists of the following several steps. First, we need to identify the system performance metrics that are most concerned in video analytics, together with a set of independent variables that are related to video analytics task deployment. Second, we need to model the relationship among the system performance and independent variables. Third, we need to design an algorithm to obtain the optimal configurations. After introducing these three steps, we demonstrate how to apply EdgeEye in a real-life video analytics application.

1. The term is also named output detection frame rate.

3.1 Performance Metrics and independent Variables.

In this work, we use three important system performance metrics including accuracy, delay, and power consumption to study the performance of video analytics. Please note that video analytics in this paper focuses on face detection and recognition. Other video analytics applications may define their own performance metrics.

Accuracy. *Accuracy* is one of the performance metrics of DNN based video analytics. In this paper, we mainly focus on the accuracy of targeted detection model, which is measured as the Intersection over Union (IoU) between the predicted bounding box and ground truth bounding box. Taking face detection and recognition task as an example, we consider the factor related to accuracy: the correctness of the position of the detected faces in the picture frame, while the IoU can be calculated as (Area of Overlap)/(Area of Union) [24]. In particular, the feature embedding accuracy of the trained face recognition model is fixed, and the accuracy of face recognition is related to the selection of the face data set of the face feature classifier trainer. Therefore, we does not consider the accuracy of face recognition in this work.

Delay (*i.e., real-time*). Video analytics is mostly deployed in real-time applications, which expect a low delay [25], *i.e.,* response should be delivered to users before a deadline. In particular, in the edge-server model, the delay includes the execution time on the edge device (l_{ec}) and the waiting time (l_{sc}). The metric waiting time including the transmission time from the edge device to the edge server, execution time in the edge server, and the feedback from the edge server to the edge device. However, for the edge-only model, we have $l_{sc} = 0$. The detection frame of the the video can directly affect the user's perception of the video. Thus, we use the ***detection frame rate*** [26] as the performance metric to evaluate the delay for video analytics, *i.e.,* $f = k/(l_{ec} + l_{sc})$, where k is the total number of frames in video stream.

Power Consumption. Besides accuracy and delay, edge devices used in video analytics mostly have limited amount of energy. The whole energy consumption of video analytics is largely depends on the execution time. An edge device is preferable to have lower power consumption and longer battery life. Thus, we use energy consumption per unit time as the *power consumption*, *i.e.,* $p = b/(l_{ec}+l_{sc})$, where b is the total energy consumption of tasks.

Next, we identify a set of independent variables, including parameters in the DNN model, parameters in the video stream, and resources on edge devices, which are closely related to the performance of video analytics.

Parameters in the DNN model. The parameters in the DNN model affects accuracy, delay, and power consumption. We define a set of parameters related to the DNN model as $M = \{m_1, m_2, ..., m_x\}$, which can be configured according to the DNN model. For example, in the MTCNN based face detection, the minimum detectable face size can be defined as one of the independent variables in the DNN model.

Parameters in the video stream. We configure parameters in the video stream to study their impact on performance. We set a parameter set related to the video stream as $V = \{v_1, v_2, ..., v_y\}$, which can be configured according to the video analytics. For example, resolution, bit rate, and input frame rate in the video stream can be defined as v_i. In edge video analytics, high-definition video stream transmits large-scale video data that aggravates burden on the network. Thus, different video analytics tasks should choose appropriate parameters in video stream under different network connections.

Resources on edge devices. The performance of video analytics depends on available computing resources and communication bandwidth at the edge device. In the collaborative edge video analytics, the edge device can conduct part or all of the offloading task. Thus, the network of the edge devices also have impact on the system performance. For example, the harsh network condition can bring transmission delay. We set the network bandwidth on the edge device as bw. Edge devices used in the system are represented by $D = \{\langle d_1, bw\rangle, \langle d_2, bw\rangle, ..., \langle d_n, bw\rangle\}$.

Multiple constraints. In addition to the aforementioned lists of parameters, there are also multiple constraints. First, the video analytics should be executed under the constraints of device price and mobility. Second, video analytics should satisfy users' requirements in terms of accuracy, delay, and power consumption. These constraints are denoted as $Con = \{con_1, con_2, ..., con_z\}$.

3.2 Problem Formulation

The previous analysis shows that the relationship between system performance and independent variables is complicated in an edge video system. EdgeEye aims to find an optimal configuration to obtain the optimal performance of video analytics on an edge device. In this process, the values of independent variables that affect system performance can be specified as a subset of the sample space that is defined by choosing various values of parameters in the DNN model, the video stream, and the choice of edge devices. When we choose a specific value for each parameter, the combination results in one specific system. Choosing different values of parameters, we will obtain all possible systems with various performance. There may be a subset of the above constructed systems satisfying the system constraints which include user's requirements. However, each system may have its own optimization objective. A system with a specific configuration can maximize the defined objective. It is hard to model the mathematical relationship between system performance and independent variables. When the independent variables are set to be M', V', and D', we set the abstract system performance models in EdgeEye as Eq. 2. It must be noted that D' includes the selected device d_i and the upload bandwidth of the device bw. Table 1 lists notation and description of symbols used in this section.

$$\begin{cases} a_i = \mathbb{A}_i\{M', V', D'\} \\ f_i = \mathbb{F}_i\{M', V', D'\} \\ p_i = \mathbb{P}_i\{M', V', D'\} \end{cases} \quad st. \quad Con' \tag{2}$$

In this paper, we aim to find the configurations to achieve optimal performance in terms of detection frame rate and accuracy under the constraint of power consumption and edge devices costs in deploying edge video analytics. We can defined this configuration problem of video analytics as a combinatorial optimization problem (COP). We set the objective as maximizing the weighted sum of detection frame rate and accuracy under user's requirement of the detection frame rate and accuracy. Meanwhile, independent variables of this COP are referred to as parameters in the DNN model (*i.e.,* M'), the parameters in the video stream (*i.e.,* V'), and the choice of edge device (*i.e.,* d_i), as presented in **Problem 1**.

Problem 1:

$$\begin{aligned} Objective: \quad & \max_{i, M', V'} a_i + \alpha \cdot f_i \\ s.t. \quad & \begin{cases} \forall i : a_i \geq A \\ \forall i : f_i \leq F \\ \forall i : p_i \leq P \\ \forall i : c_i \leq C \end{cases} \\ vars: \quad & M' \subset M, V' \subset V, d_i \in \{0, 1\}, i = 1, 2, 3, ..., N \end{aligned}$$

$\alpha \in [0, 1]$ is the weight between the detection frame rate and accuracy, which depends on users' requirement of the detection frame rate or accuracy in the real-life applications. The objective function in the **Problem 1** can be defined based on various combination of performance goals. Here we just show one example of optimization objectives.

Based on the work in [27], we propose the the data-driven approach EdgeEye to solve this COP, **Problem 1**. EdgeEye leverages the mathematical relationship - performance models - in Eq. 2 and tested results in terms of accuracy, detection frame rate, and power consumption

TABLE 1
Notation and Description

Notation	Description
M=$(m_1, m_2, ..., m_x)$	Configurable parameters in the DNN model
m_i	A component of M, $i = 1, 2, 3, ..., x$, x is the dimension of M
M′	A subset of M
V=$(v_1, v_2, ..., v_y)$	Configurable parameters in the video stream to be processed
v_i	A component of V, $i = 1, 2, 3, ..., y$, y is the dimension of V
V′	A subset of V
$D = \{\langle d_1, bw\rangle, \langle d_2, bw\rangle, ..., \langle d_n, bw\rangle\}$	Configurable parameters in the choices of edge devices
d_i	A component of D, $i = 1, 2, 3, ..., N$, N is the total number of selectable edge devices with different resources
D′	A subset of D
$Con(con_1, con_2, ..., con_z)$	The combination of constraints
con_i	A component of Con, $i = 1, 2, 3, ..., z$, z is the total number of constraints
Con'	A subset of Con
a_i	The accuracy value of video analytics on the i^{th} edge device
$\mathbb{A}_i$(M′,V′,D′)	The accuracy function of video analytics on the i^{th} edge device
A	Targeted accuracy
f_i(M′,V′,D′)	The detection frame rate value of video analytics on the i^{th} edge device
$\mathbb{F}_i$(M′,V′,D′)	The detection frame rate function of video analytics on the i^{th} edge device
F	Targeted detection frame rate
p_i(M′,V′,D′)	The power consumption value of video analytics on the i^{th} edge device
$\mathbb{P}_i$(M′,V′,D′)	The power consumption function of video analytics on the i^{th} edge device
P	Targeted power consumption
c_i	Costs of the i^{th} device
C	Targeted cost in edge video anlytics

Algorithm 1 Configuration parameters decision algorithm

Input: Targeted accuracy (A), targeted detection frame rate (F), targeted power consumption (P), the tested results vectors of the accuracy, detection frame rate, and the power consumption for video analytics, edge device costs ($c_i, i = 1, 2, ..., N$), the temple variables of the DNN model parameters (m), video stream setting (v), and value of the objective function (u) based on m and v.

Output: Configuration parameters in the DNN model (M'^*), parameters in the input video stream (V'^*), the choice of edge device (d^*).

1: Initialize configuration parameters on an edge device.
2: u_{max}=0
3: **for** $i \leftarrow 1\ to\ N$ **do** //N is the number of edge devices.
4: **for** each (m_i, v_i) in (M', V') **do**
5: $m \leftarrow m_i$ //Configuration parameter m_i is fixed
6: $v \leftarrow \arg\max_{v_i} (a_i + \alpha \cdot f_i)$
7: **if** $a_i \geq A\ and\ f_i \geq F$ **then**
8: $u \leftarrow a_i + \alpha \cdot f_i$
9: **end if**
10: **if** $u > u_{\max}\ and\ c_i \leq C\ and\ p_i \leq P$ **then**
11: $u_{\max} \leftarrow u$
12: $M'^* \leftarrow m,\ \ V'^* \leftarrow v,\ \ d^* \leftarrow e_i$ //e_i is the unit vector for device selection
13: **end if**
14: **end for**
15: **end for**
16: return M'^*, V'^*, d^*.

to improve the search efficiency for optimal configuration parameters. We describe the details in Algorithm 1.

- Lines 1∼5: traverse the tested results obtained under many parameter settings at different devices including accuracy, detection frame rate and power consumption of video analytcis.
- Lines 4∼8: when a configuration parameter in M' (or V') and the constraints of the targeted accuracy, detection frame rate and devices costs are fixed (e.g., M' is fixed to m), we can obtained the other parameter (e.g., V' can be fixed to v) that makes the system performance optimal.
- Lines 9∼11: if the current configuration can achieve a better performance combination and meet the constraints of power consumption and the edge device costs, then set the current configuration as the optimal configuration.
- Lines 12∼15: after repeated searches, we can obtain the optimal configurations, including the DNN model parameters (M'^*), video stream parameters (V'^*), and the choice of edge device (d^*).

Changing the configuration parameters (M', V', and D'), we can get extensive tested results vectors for video analytics at different edge devices based on edge computing model as the input of the **Algorithm 1**. The tested results vectors can be taken as the input of **Algorithm** 1. Executing Algorithm 1, first, we can record the feasible configuration solutions that satisfy the constraints in lines 7 and 10, which include the optimal configuration solution. In addition, according to the feasible configuration solutions, we can further extrapolate which edge computing model can achieve better performance. Specifically, in the Section 5.2, we will analyze the optimal configuration results under different edge computing models and targeted performance metrics. If the algorithm cannot find a configuration solution, it means the edge device cannot to used for the task under the specific edge computing model. Besides, we can also find the optimal edge computing model for video analytics.

3.3 Application of EdgeEye in Face Detection and Recognition

Taking the face detection and recognition task as an example, we showcase how EdgeEye can be applied in a real-life application based on edge-only model and edge-server model, where MTCNN is used for face detection and FaceNet is applied for face recognition. It must be noted that, in the edge-only model, face detection and recognition are executed on the edge device. In the edge-server model, the face detection and recognition are deployed on an edge device and an edge server, respectively.

3.3.1 Performance Models

Referring to the abstract system performance models in Eq. 2, we study relationship between the performance (*i.e.*, accuracy, detection frame rate, and power consumption) and independent variables of MTCNN based face detection and FaceNet based face recognition video analytics with two edge computing models.

Accuracy model. In this work, the term *accuracy* is the accuracy of MTCNN based face detection. Referring the abstract accuracy model in Eq. 2. In the MTCNN, a smaller minimum detectable face size ($\min\_size$) results in more levels in the image pyramid, which increases the accuracy of face detection [22]. In addition, the low input frame rate (f_{in}) can reduce the detectable rate of the video stream, thereby lowering the accuracy of face detection. In this work, we define the accuracy model of face detection as

$$a'_i = \mathbb{A}'_i\left(\min\_size, f_{in}, null\right), \tag{3}$$

where the $\min\_size$ represents the parameter in M', the f_{in} represents the parameter in V', and the $null$ means that the parameter is empty.

In this work, we measure accuracy by Intersection over Union (IoU). The average of face images' IoUs in the frame is referred to as frame IoU; meanwhile, the average of the frames in the video is video IoU. Thus, the accuracy of m video clips with n frames per video clips is denoted as

$$\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} IoU}{m \times n} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{area(R \cap P)}{area(R \cup P)}}{m \times n} \tag{4}$$

where notations R and P represent the bounding boxes of the ground truth and the detection model after setting parameters in the DNN model and video stream. In the ground truth scheme, we set the minimum detectable face size to 30 pixel^2 in the MTCNN while the input frame rate of the video has the same value as that of the source video, which means that we detect each frame.

Detection Frame Rate Model. Based on our prior knowledge, the minimum detectable face size in MTCNN ($\min\_size$)and the input frame rate (f_{in}) are independent variables of the delay. Besides, variant computing resources on different devices result in different delay. In addition, in edge-server model, delay is also related to the available bandwidth. As discussed before, the delay in this paper is measured by the detection frame rate, which is defined as follows for both edge-only and edge-server models.

$$\begin{cases} f'_{i(e-o)} = \mathbb{F}'_i(\min\_size, f_{in}, d_i) & , edge-only \\ f'_{i(e-s)} = \mathbb{F}'_i(\min\_size, f_{in}, (d_i, bw)) & , edge-server \end{cases} \tag{5}$$

The symbol bw means the average up-link network bandwidth at the edge device. It is noted that the detection frame rate model of the edge-only model and the edge-server model are different in that edge-only model does not need to consider bandwidth as all video analytics is performed locally. In addition, different network conditions in the edge-server model can impact on the tested results.

Power Consumption Model. The performance of video analytics is also impacted by user's requirement of energy consumption per second on an edge device. The energy consumption to complete a video analysis task relates to the total execution time and resources on the device. For the edge-server face recognition, we consider edge devices' energy consumption during computing and transmission. Thus, the energy consumption model for two edge computing models based face detection and recognition can be demonstrated as

$$\begin{cases} b'_{i(e-o)} = \mathbb{B}'_i(l_{ec}, d_i) & , edge-only \\ b'_{i(e-s)} = \mathbb{B}'_i(l_{ec}, d_i) + \mathbb{B}'_i(l_{sc}, d_i) & , edge-server \end{cases} \quad (6)$$

Symbols $b'_{i(e-o)}$ and $b'_{i(e-s)}$ represent the energy consumption of local computing in the edge-only model and the energy consumption of local computing and transmission in the edge-server model, respectively.

The power consumption is calculated as the ratio of the total energy consumption to the delay, such as $p = b/(l_{ec} + l_{sc})$. Thus, according energy consumption model in Eq. 6, we can define the power consumption model as

$$\begin{cases} p'_{i(e-o)} = \mathbb{P}'_i(\min\_size, f_{in}, (d_i, null)) & , edge-only \\ p'_{i(e-s)} = \mathbb{P}'_i(\min\_size, f_{in}, (d_i, bw)) & , edge-server \end{cases} \quad (7)$$

Multiple constraints. In a real application, the deployment of face detection and recognition task should meet the users' requirements in terms of targeted accuracy (A), detection (F) and power consumption (P). In additional, in this application, we consider the choice of edge devices should meet the targeted device costs (C).

3.3.2 Problem Definition and Algorithm

In this section, referring to **Problem 1** and **Algorithm 1**, we present the problem definition and the algorithm when EdgeEye is applied in the face detection and recognition application. In **Problem 2**, the objective of this application is to maximize the weighted sum of accuracy and detection frame rate under two edge computing models.

Problem 2:

$$\begin{aligned} \max_{i, min\_size, f_{in}} \quad & a_i + \alpha \cdot \varepsilon \begin{bmatrix} f'_{i(e-o)} \\ f'_{i(e-s)} \end{bmatrix} \\ s.t. \quad & \forall i : \varepsilon \begin{bmatrix} f'_{i(e-o)} \\ f'_{i(e-s)} \end{bmatrix} \leq F \\ & \forall i : \varepsilon \begin{bmatrix} p'_{i(e-o)} \\ p'_{i(e-s)} \end{bmatrix} \leq P \\ & \forall i : c_i \leq C \\ & \forall i : a'_i \geq A \\ & vars \;\; min\_size, f_{in} \geq 0, \varepsilon = [0,1] \; or \; [0,1], \\ & d_i \in \{0,1\}, \sum_{i=1}^{N} d_i = 1, i = 1, 2, ..., N \end{aligned}$$

In Problem 2, ε can have two choices, $[1\ 0]$ and $[0\ 1]$ for edge-only model and edge-server model respectively. The independent variables of the **Problem 2** are the minimum detectable face size ($\min\_size$) in the MTCNN, input frame rate (f_{in}), and appropriate edge device (d_i). Then, to obtain optimal configurations of face detection and recognition on the edge video system, the algorithm in EdgeEye is presented as **Algorithm 2.**

In EdgeEye, tested results vector (including the tested data of accuracy, detection frame rate, power consumption) and user requirements for face detection and recognition are taken as input parameters of **Algorithm** 2. These tested results are obtained through experiment with different configuration parameters on different devices, as shown in Section 5.1. Executing **Algorithm 2**, we can obtain feasible configurations including the parameters of input frame rate, the minimum detectable face size in MTCNN, and appropriate edge devices under different network connections. Further, we can analyze feasible configurations to find the optimal ones according to user requirements of targeted accuracy, detection frame rate, power consumption and edge device costs, see Section 5. In addition, according to the optimal configuration in the edge-only and edge-server models, we can further deduce which edge computing model can achieve better performance in face detection and recognition. To adapt the algorithm for different objectives, we can modify the objective function, see Lines 5 and 7 in **Algorithm** 2.

Algorithm 2 Configuration parameters decision algorithm in face detection and recognition

Input: Targeted accuracy (A), targeted detection frame rate (F), targeted power consumption (P), the tested results vectors including the accuracy, detection frame rate, power consumption for face detection and recognition and edge devices costs ($c_i, i = 1, 2, ..., N$)
Output: Input frame rate (f^*_{in}), the minimum detectable face size (min_size^*), the choice of edge device.

1: Initialize configuration parameters of an edge device.
2: u_{max}=0
3: **for** $i \leftarrow 1$ *to* N **do**
4: **for** $f_{in} \leftarrow 0$ *to* f_{raw} **do** //f_{raw} is the frame rate of the video source.
5: $\min\_size \leftarrow \arg\max_{\min\_size \geq \min\left(a_i^{-1}(A|f_{in}), f_i'^{-1}(F|(f_{in}, d_i))\right)} \left(a_i\left(\min\_size, f'_{in}\right) + \varepsilon \left[f'_{i(e-o)}\ f'_{i(e-s)}\right]^T\right)$
6: **if** $a_i \geq A$ *and* $\varepsilon \left[f'_{i(e-o)}\ f'_{i(e-s)}\right]^T \geq F$ **then**
7: $u \leftarrow a_i\left(\min\_size, f_{in}\right) + \varepsilon \left[f'_{i(e-o)}\ f'_{i(e-s)}\right]^T$
8: **end if**
9: **if** $u > u_{\max}$ *and* $c_i \leq C$ *and* $\varepsilon \left[p'_{i(e-o)}\ p'_{i(e-s)}\right]^T \leq P$ **then**
10: $u_{\max} \leftarrow u$
11: $\min\_size^* \leftarrow \min\_size, f_{in}{}^* \leftarrow f_{in}, d^* \leftarrow e_i$
12: **end if**
13: **end for**
14: **end for**
15: return $\min\_size^*, f_{in}{}^*, d^*$.

4 Experiment Setup

To validate EdgeEye, we conducted experiments to study the performance of face detection and recognition on an platform that supports edge-only and edge-server models. With various system configurations, we can obtain performance results of face detection and recognition in terms of accuracy, delay, and power consumption.

As shown in Fig. 4, the platform is composed of an edge node (mobile camera and edge device), an edge server, Wi-Fi router, and a cloud node.

In the edge node, an edge device can locally process video data from a mobile camera or transmit it to an edge server through wireless network. We select six devices (see

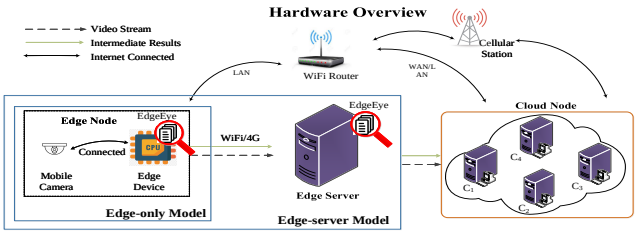


Fig. 4. Overview of EdgeEye-enabled evaluation system.

Table 2) in this test. In the edge-only model, the six devices are used as edge devices to perform face detection and recognition. However, in the edge-server model, devices D1, D2, D4, and D5 are used as edge devices. Device D6 is used as the edge server to sends feedback to each edge device. The costs of these devices are different, see Table 2. It must be noted that the memory size on device D_4 is larger than that of D_3.

The edge server receives the video data uploaded by the edge node and performs all or part of the video analytics. The cloud node can provide DNN model training, partial DNN inference, video or state data management, etc. It must be noted that the DNN-based detection or recognition models deployed on the edge device and the edge server are the same. In this evaluation system, the video stream and status information transmits among nodes through 4G/WiFi. We deploy the EdgeEye engine at the edge device and the edge server. EdgeEye can obtain feasible configurations and provide system designers with optimal configurations for deploying video analytics on the platform.

In the edge-only model, face detection and recognition is deployed on edge device. Three types of minimum detectable face size are used in MTCNN, including 30 pixel^2, 60 pixel^2, and 90 pixel^2. The input frame rate (frames/s) in the video stream is 15 fps, 10 fps, and 5 fps. On each device, the total execution time and energy consumption for video analytics are recorded in a log file to obtain the detection frame rate and power consumption. In the edge-server model, face detection and recognition is partitioned into face detection and face recognition sub-tasks which are offloaded at the edge device and the edge server, respectively. When face detection completes at the edge device, the detected face images are transmitted to the edge server through the ZeroMQ protocol [28]. The edge server returns the result to the edge device when face recognition finishes. The edge device records the total execution time and energy consumption in a log file. Finally, we calculate the detection frame rate and power consumption.

We employ 50 video clips in the data set [29] as tested videos. MTCNN face detection and FaceNet face recognition are conducted at the edge device with different parameters settings. We use wondershaper (V1.4) [30] to configure upload bandwidths at the edge (*i.e.*, $\leq$ 50 kbps, $\leq$100 kbps, $\leq$200 kbps, $\leq$500 kbps, and $>$1 Mbps) to study the impact of the network bandwidth on the analytics results.

TABLE 2
Details of Edge Devices

Device	No.	CPU@GHz	RAM (GB)	OS	Mobile	Device Costs
Raspberry Pi(3 B+)	D1	ARMv71*4@1.4	1	Linux 4.19	✓	$\leq$ \$80
NanoPi-NEO4	D2	(Cortex A72*2 + A53*4)@1.5	1	Linux 4.4	✓	$\leq$ \$80
Toybrick RK3399Pro	D3	Cortex A72@1.8*2 + A53@1.4*4	2	Linux 4.4	✓	\$80 $\sim$ \$150
Firefly-RK3399	D4	(Cortex A72*2 + A53*4)@1.8	4	Linux 4.4	✓	\$80 $\sim$ \$150
Jetson TX2	D5	(ARMv8*6 + Pascal GPU)@2	8	Linux 4.4	✓	\$150 $\sim$ \$750
ASUSPRO	D6	Intel Core i5-7400*4@3	12	Linux 4.15	×	$\geq$ \$750

5 Evaluation for EdgeEye

5.1 Input Parameters of EdgeEye in an Application

The preliminary test in Fig. 3 shows the complex relationship between the configuration and performance of video analytics. It motivates us to establish abstract models between performance metrics and their independent variables, and propose a data-driven optimal configuration solution approach, EdgeEye. In EdgeEye, it is necessary to take the performance tested results under different configurations as input to solve the optimal configuration of the video analytics. Thus, in order to validate the rationality of the preset performance models in Section 3.3.1 and obtain the input tested result of **Algorithm** 2, we conducted extensive performance tests in terms of accuracy, delay, and power consumption of face detection and recognition at edge-only and edge-server models under different configuration parameters.

5.1.1 Accuracy

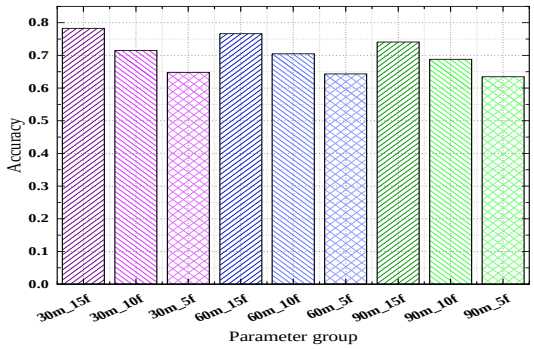
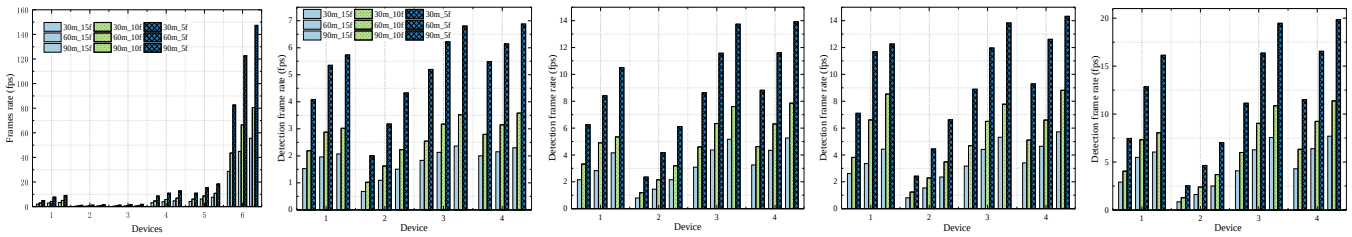


Fig. 5. The accuracy of face detection under the minimum detectable face size in the MTCNN and the input frame rate in video stream. The symbol '$30m\_15f$' means the 30-pixel^2 minimum detectable face size and 15-fps input frame rate.

Fig. 5 shows the accuracy of the video analytics with the minimum detectable face size in the MTCNN and the input frame rate of video stream. When the minimum detectable face size is 30 pixel^2 and the input frame rate of video stream is 15 fps, the accuracy is the highest. However, the accuracy decreases as the minimum size of the detected face becomes large.

Insight 1. The above results validate the relationship between accuracy and its independent variables in Eq. 3. In this paper, we focus on accuracy of MTCNN face detection,



(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 6. Fig. 6(a) presents the detection frame rate of face detection and recognition in the edge-only model. Figs. 6(b) $\sim$ 6(e) show the detection frame rate in the edge-server model under various network bandwidths (50 kbps, 100 kbps, 500 kbps, and >1 Mbps).

rather than the FaceNet face recognition. According to Eq. 4, on each device, the edge-only accuracy of face detection and recognition has the same value with edge-server models under the same configuration.

5.1.2 Delay

In this section, we present the system performance in terms of delay (*i.e, the detection frame rate*) of face detection and recognition in the two edge models.

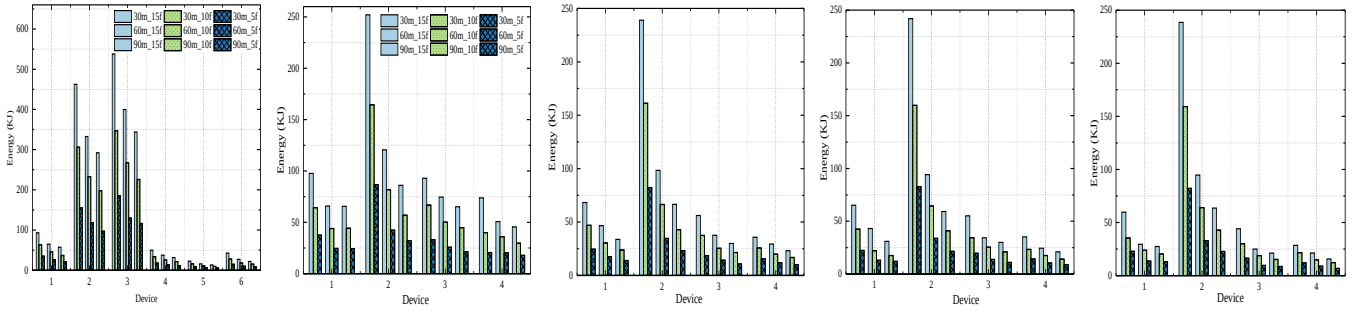
Fig. 6(a) shows the detection frame rate in the edge-only model. Figs. 6(b) $\sim$ 6(e) present the detection frame rate on edge devices D1, D2, D4, and D5 in the edge-server model. We employ four network bandwidths (*i.e.*, 50 kbps, 200 kbps, 500 kbps, >1 Mbps). In Fig. 6(a), edge device D6 presents the highest detection frame rate while edge device D2 has the lowest in the edge-only model. With a high-bandwidth network, the edge-server model has a higher detection frame rate than the edge-only model. In the edge-server model, the real-time performance varies with the bandwidth. For devices D1, D4, and D5, the performance improves when the network bandwidth increases. With 30-pixel^2-size minimum detectable face size and 15-fps input frame rate, the execution time on device D1 is 1.53 fps, 2.16 fps, 2.61 fps, and 2.91 fps under four bandwidths, respectively, in the edge-server model. The execution time of face recognition is 1.95 fps in the edge-only model.

Insight 2. We can find that the edge-server model significantly improves the real-time performance of each device in the edge-only model. In Fig. 6, device D6 has the highest detection frame rate while device D2 has the lowest one in the edge-only model. In the edge-server model, the performance of devices D1 and D2 rises under the $\geq$ 200-kbps bandwidth. The edge-server model presents a higher detection frame rate compared with the edge-only one.

5.1.3 Power Consumption

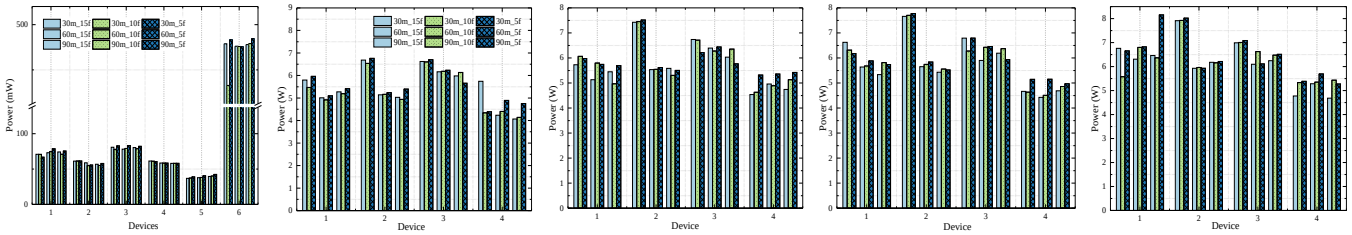
Fig. 7(a) shows the energy consumption on devices D1 $\sim$ D6 in the edge-only model. Figs. 7(b) $\sim$ 7(e) present the edge-server energy consumption on devices D1, D2, D4, and D5 under four network connections. With high bandwidth, the edge-server energy consumption reduces compared with that of the edge-only model. With 30-pixel^2-size minimum detectable face size and 15-fps input frame rate, the energy consumption of device D1 using edge-server model is 97.6 KJ, 68.3 KJ, 65.2 KJ, and 59.8 KJ under four network bandwidth. The energy consumption of device D1 using edge-only model is 93.4 KJ, which is higher than that of the edge-server model when network bandwidth is higher than 200 kbps. On devices D2, D4, and D5, the edge-server energy consumption is smaller than that of the edge-only model when the bandwidth is higher than 500 kbps.

Insight 3. We present the power consumption of face recognition on each device in the two models in Fig. 8. The power consumption increases slightly when the bandwidth reduces because of less data transmission. Results validate that the bandwidth (*i.e., bw* in Eq. 7) is an independent variable in the power consumption model. In the edge-server model, an x86 based D6 that has the highest power consumption cannot be used as the edge device (see Fig. 7(a)). However, device D2 takes the longest time to complete the task but the device D5 presents the lowest power consumption compared with device D2. This is because the most powerful edge device D5 is equipped with NVIDIA graphics processing units (GPU).



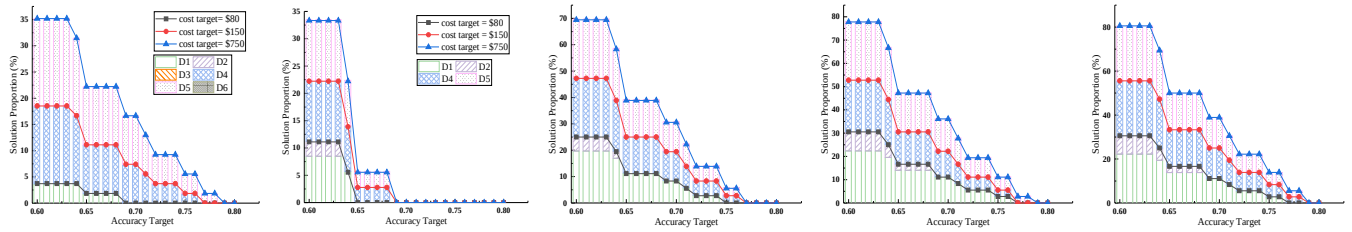
(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 7. Fig. 7(a) shows the energy consumption of face detection and recognition in the edge-only model. Fig. 7(b) $\sim$ 7(e) present the energy consumption in the edge-server model under various network bandwidths (50 kbps, 100 kbps, 500 kbps, and >1 Mbps).



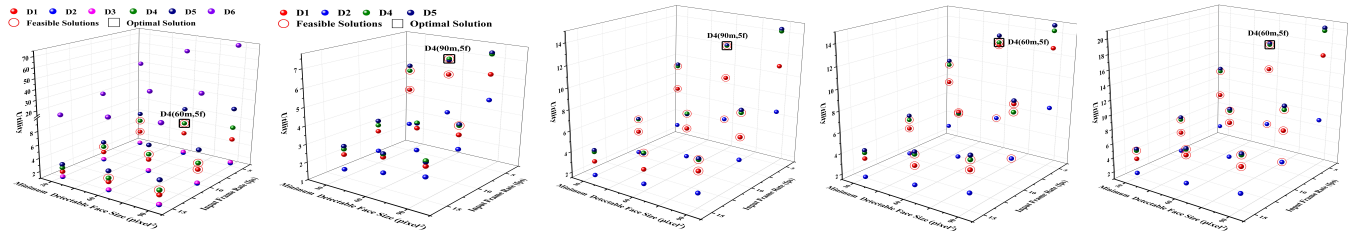
(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 8. Fig. 8(a) presents the power consumption of face detection and recognition in the edge-only model. Fig. 8(b) ∼ 8(e) show the power consumption in the edge-server model under various network bandwidths (50 kbps, 100 kbps, 500 kbps and >1 Mbps).



(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 9. Fig. 9(a) presents the proportion of configuration solutions on six edge devices under the targeted accuracy in the edge-only model. Figs. 9(b) ∼ 9(e) show the proportion of configuration solutions based on the targeted accuracy under network bandwidth in the edge-server model.



(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 10. Fig. 10(a) presents optimal configuration solutions based on the targeted accuracy of 0.64 in the edge-only model. Figs. 10(b) ∼ 10(e) show optimal configuration solutions based on the targeted accuracy of 0.64 in the edge-server model under 50-kbps, 100-kbps, 500-kbps, and >1-Mbps bandwidths.

5.2 Configuration Solutions Analytics

EdgeEye aims to obtain optimal configurations to deploy the DNN based video analytics on edge computing systems. Taking tested results (see Section 5.1) as the input of **Algorithm 2**, we configure the targeted accuracy, detection frame rate, and power consumption. Under different configurations, results of **Algorithm 2** are a combination of values for parameters which meet the targeted performance in terms of accuracy, detection frame rate, power consumption, and device costs. When the feasible configurations exist, we obtain the optimal configurations for the objective function in **Problem 2**. We may want to say that there could be zero, single, or multiple combinations of values of parameters to meet the targeted performance. If there is not possible to meet the targeted performance, no parameters can be found at the edge-server or edge-only modes. If there is multiple combinations of configuration parameters, we can find the optimized one to meet the targeted performance. Figs. 9, 11, and 13 present the proportion of solutions for face detection and recognition in the two edge models.

In this section, results unveil that the optimal configurations depend on the three-type relationship. First, the minimum detectable face size determines the face detection accuracy, delay, and power consumption. In the MTCNN model, the small detectable face size in the MTCNN model results in a high accuracy for face detection, but it also leads to low detection frame rate and few feasible configurations to meet the targeted accuracy (see Section 5.2.1). Second, the input frame rate of the video stream can impact on the delay and accuracy of video analytic tasks. Small input frame rate can increase the detection frame rate and shorten the delay, then we can obtain many configurations to meet the targeted delay; however, it can impair the accuracy (see Section 5.2.2). Third, power consumption depends on computing resources at edge devices. Using powerful edge devices can reduce delay while maintain accuracy, but it may fail to meet targeted power consumption and device costs (see Section 5.2.3).

It must be noted that the targeted detection frame rate and targeted power consumption are fixed when we analyze the results under the targeted accuracy. The system designers can select the optimal configurations from the feasible solutions according the objective function in the COP.

TABLE 3
Optimal configuration solution in terms of the minimum detectable face size, input frame rate, selected edge device, and edge computing model under different targeted accuracy.

	Device Costs	Targeted Accuracy						
		0.61	0.64	0.67	0.70	0.73	0.76	0.79
50kbps	C<$80	D1(90m,5f)	D1(60m,5f)	D1(90m,10f)	/	/	/	/
	C<$150	D4(90m,5f)	D4(90m,5f)	D4(90m,10f)	D4(90m,10f)	D4(90m,5f)	D4(90m,5f)	/
	C<$750	D5(90m,5f)	D5(60m,5f)	D5(30m,5f)	D5(60m,10f)	D5(90m,15f)	D5(60m,15f)	/
500kbps	C<$80	D1(90m,5f)	D1(60m,5f)	D1(90m,10f)	D1(60m,10f)	D1(90m,15f)	D1(60m,15f)	/
	C<$150	D4(90m,5f)	D4(60m,5f)	D1(90m,10f)	D1(60m,10f)	D4(90m,15f)	D4(60m,15f)	/
	C<$750	D5(90m,5f)	D5(60m,5f)	D5(90m,10f)	D1(60m,10f)	D5(90m,15f)	D5(60m,15f)	/

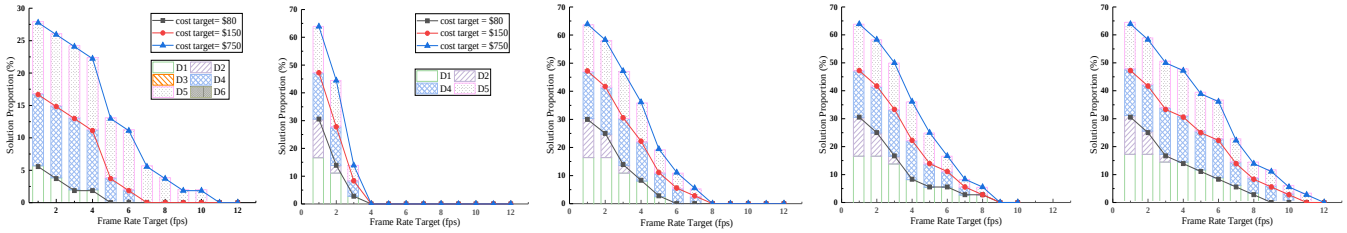
5.2.1 *Targeted Accuracy based Configuration Solutions Analytics.*

Fig. 9 shows the solutions proportion based on the targeted accuracy in the two edge models. The number of available solutions reduces as the targeted accuracy increases in the edge-only model. Terms A and C represent the user's targeted accuracy of the edge model and the targeted costs of an edge device, respectively. With $C\leq$ \$150 and $A\geq$ 0.69, only device D4 has feasible configuration solutions. With $C\leq$ \$150 and $A\geq$ 0.77, there are no feasible solutions. When $C\leq$ \$750, device D5 has a feasible configurations in Fig. 9(a). As the targeted accuracy rises, the proportion of feasible configuration solutions presents a similar downward trend in the two models. In the edge-server model, the number of feasible configuration solutions increases as device costs constraint reduces. In Fig. 9(b), with $bw\leq$ $50kbps$ and $C\leq$ \$150 and $A\geq$ 0.65, D2 is not feasible; meanwhile D1 becomes infeasible when $C\leq$ \$150 and $A\geq$ 0.69. Figs. 9(b) $\sim$ 9(e) unveil that devices have feasible configurations when $C\leq$ \$750. Devices D1 and D2 have feasible configurations with $C\leq$ \$750. When we have $C\leq$ \$750 and $A=$ 0.60, the proportion of feasible configurations to all configurations we select is 33.33 %, 69.44 %, 77.78 %, and 80.56 % under $\leq$50-kbps, $\leq$200-kbps, $\leq$500-kbps, and $\geq$ 1-Mbps bandwidth, respectively.

Observation 1. According to feasible configuration solutions of the targeted accuracy, when we have the fixed detection frame rate, power consumption constraints and the edge device costs constraint of \$150, we can obtain the optimal configuration solution for video analytics. Fig. 10 shows the optimal configuration of edge-only and edge-server models when the targeted accuracy is 0.64. In Fig. 10(a), the optimal configuration under edge-only computing model is D4(60m,5f), which means that the optimal configuration solution includes the minimum detectable face size of 60 pixel^2, the input frame rate of 5 fps, and the selected edge device of D4. in the edge-server computing model. When the network bandwidth is low (*e.g.*, 50 kbps), there are fewer feasible configurations. The optimal configuration is D4(90m,5f), see Fig. 10(b). As shown in Figs. 10(c) $\sim$ 10(e), when the network bandwidth increases, the number of feasible configurations increases, and the optimal configuration solutions searched by EdgeEye are D4(90m,5f), D4(90m,5f), and D4(90m,5f) under bandwidth of 200 kbps, 500 kbps, and >1 Mbps, respectively. Specifically, Table 3 shows that the optimal configuration solution (including the minimum detectable face size, input frame rate, selected edge device, and edge computing model) varies with the targeted accuracy under three different cost constraints and two bandwidth (50 kbps and 500 kbps). The blue cell indicates the edge-server computing model is used, the orange cell indicates the edge-only computing model, and the symbol '/' indicates that there is no feasible configuration solution in this case. We can find that when the network bandwidth is low, the optimal configuration solution is more inclined to the edge-only computing model. When the network bandwidth increases, the edge-server computing model is selected. Above analytics results reveal that for user's different requirements of targeted accuracy, the data-driven EdgeEye still can well address the combiantorial optimization problem. EdgeEye obtains the optimal parameters in terms of the DNN model, video stream, and the optimal choices of edge devices to deploy video analytics. Besides, we can find the optimal computing model between the edge-only and the edge-server models under different network connections.

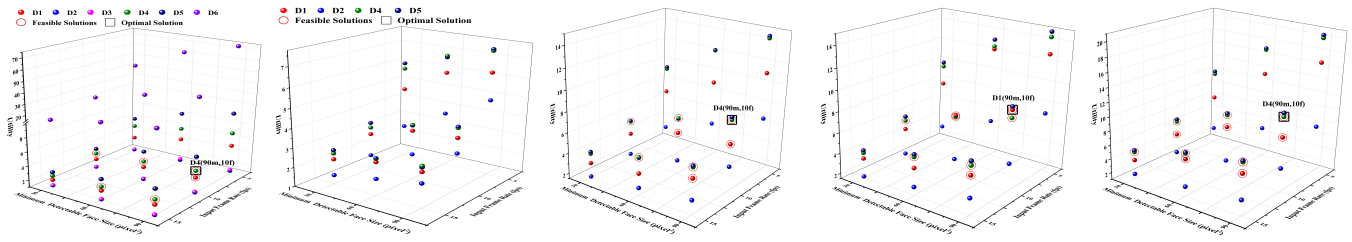
5.2.2 *Targeted Delay based Configuration Solutions Analytics.*

Figs. 11(a) $\sim$ 11(e) show that the feasible configuration solutions vary with the targeted detection frame rate in the two



(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 11. Fig. 11(a) presents the proportion of configuration solutions based on the targeted detection frame rate in the edge-only model. Figs. 11(b) $\sim$ 11(e) show the proportion of configuration solutions based on the targeted detection frame rate in the edge-server model under 50-kbps, 100-kbps, 500-kbps, and >1-Mbps bandwidth, respectively.



(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 12. Fig. 12(a) presents the optimal configuration solution under different targeted detection frame rate of four in the edge-only model. Figs. 12(b) $\sim$ 12(e) show optimal solutions based on the targeted detection frame rate of four under 50-kbps, 100-kbps, 500-kbps, >1-Mbps bandwidths in the edge-server model.

TABLE 4
The optimal configuration solution (including the minimum detectable face size, input frame rate, edge devices, and edge computing model) under different targeted detection frame rate.

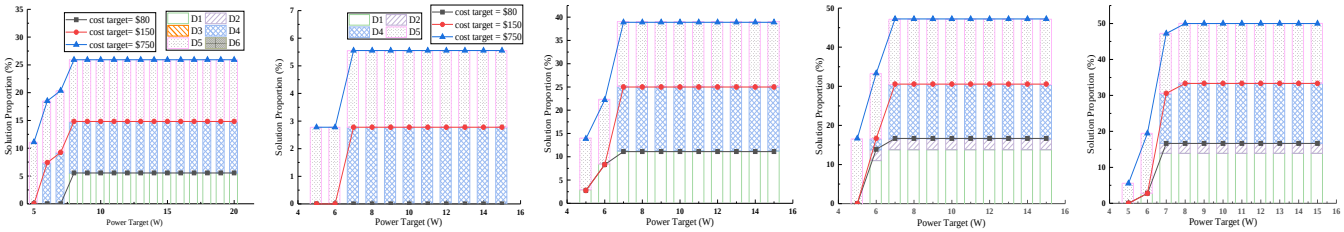
	Device Costs	Targeted Input Frame Rate(fps)						
		2	4	6	8	10	12	14
50kbps	C<$80	D1(90m,10f)	D1(90m,10f)	/	/	/	/	/
	C<$150	D4(90m,10f)	D4(90m,10f)	D4(90m,10f)	/	/	/	/
	C<$750	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	/	/
500kbps	C<$80	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	/	/	/	/
	C<$150	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	/	/	/	/
	C<$750	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	/	/	/	/

models. The proportion of feasible configuration solutions shows a downward trend as the targeted detection frame rate increases; meanwhile, the number of feasible devices decreases. In the edge-only model, the feasible configuration solutions based on the targeted detection frame rate have the same trend as that of the targeted accuracy. In Fig. 11(a), with $C\leq$ \$150 and $F\geq$ 5, the device that has feasible configuration solutions is only device D4. With $C\leq$ \$150 and $F\geq$ 7, there is no feasible configuration solutions for the six devices. The number of feasible configuration solutions and available devices increases with the reduction of costs constraint. When we have $C\leq$ \$750, device D5 has a feasible configuration solution. In the edge-server model, device D2 has no feasible configuration solution when $bw\leq$ 200 kbps and $C\leq$ \$150, and $F\geq$ 4, see Fig. 11(c). With $C\leq$ \$150 and $F\geq$ 6, device D1 becomes infeasible. When $C\leq$ \$150 and $F\geq$ 8, four edge devices have no feasible configuration solution. With the reduction of costs constraint, the number of feasible configuration solutions and feasible devices increases. With costs constraint $C\leq$ \$750, four devices have feasible configuration solutions. The $C\leq$ \$80-device costs constraint enables devices D1 and D2 to have feasible configuration solutions. Solutions for each edge device with different configurations change under network bandwidths. When $C\leq$ \$750 and $F=$ 3, the proportion of feasible configuration solutions is 33.33 %, 69.44 %, 77.78 %, and 80.56 % under $\leq$ 50-kbps, $\leq$ 200-kbps, $\leq$ 500-kbps, and $\geq$ 1-Mbps network bandwidth, respectively.

Observation 2. When the targeted accuracy and power consumption constraint are fixed, we choose edge devices to achieve the optimal performance. Fig. 12 shows the optimal configuration of edge-only computing and edge-server computing models when the targeted detection frame rate is four. As shown in Fig. 12(a), the optimal configuration under edge-only computing model is D4(60m,5f). For the edge-server model, when the network bandwidth is low, such as 50 kbps, there are no feasible configuration (see Fig. 12(b)). However, when the network bandwidth is >200 kbps in Fig. 12(c) $\sim$ 12(e), the number of feasible configurations increases, and the optimal configuration solutions searched by EdgeEye is D4(90m,10f). Specifically, Table 4 shows the optimal configurations (including the minimum detectable face size, input frame rate, selected edge device and edge computing model) that vary with the targeted detection frame rate under three different cost constraints and two bandwidth (50 kbps and 500 kbps). We also find that the optimal solution is more inclined to the edge-only computing model under the lower bandwidth; meanwhile the optimal solution is more inclined to the edge-server computing model under the higher bandwidth. EdgeEye can obtain the optimal configuration for deploying video analytics under user's different requirements of targeted detection frame rate. In addition, EdgeEye provides the optimal edge computing model under different network connections.

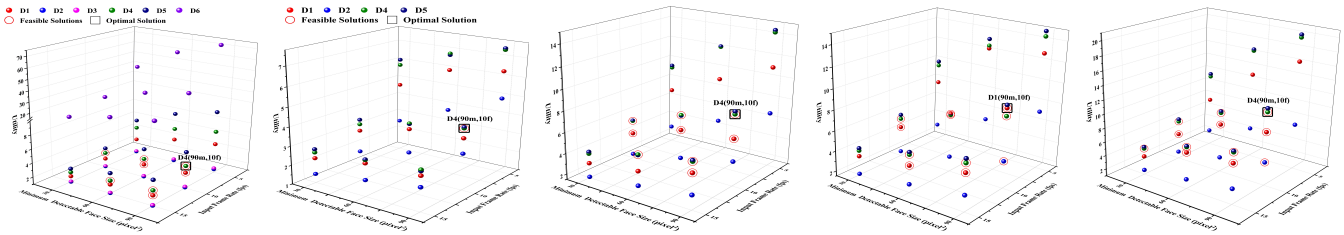
5.2.3 Targeted Power Consumption based Configuration Solutions Analytics.

In Fig. 13(a), the number of feasible configuration solutions and available devices increases as the power consumption constraint decreases in the edge-only model. The number of feasible configuration solutions in the edge-server model is much more than that of the edge-only model. In Fig. 13(b), the number of feasible configuration solutions is small (*i.e.*, no more than 6.0%) under $\leq 50-kbps$ bandwidth. With $\leq 50-kbps$ bandwidth and $C\leq$ \$80 costs constraint, there is no feasible configuration solution. The reason is that system performance cannot meet the targeted performance under configurations on each device. When bw is 200 kbps and the costs constraint $C\leq$ \$80, device D1 have a feasible configuration solution of 10% at $P\geq 7W$ (see Fig. 13(c)).



(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 13. Fig. 13(a) presents the proportion of configuration solutions based on different targeted power consumption of face detection and recognition in the edge-only model. Figs. 13(b) ∼ 13(e) present the proportion of configuration solutions based on different targeted power consumption under 50-, 100-, 500-, and >1-Mbps bandwidths in the edge-server model.



(a) edge-only (b) 50 kbps (c) 200 kbps (d) 500 kbps (e) >1 Mbps

Fig. 14. Fig. 14(a) presents the optimal configuration solution under different targeted power consumption of 0.64 in the edge-only model. Optimal solutions based on the targeted accuracy of 0.64 are shown in Figs. 14(b) ∼ 14(e) under 50-kbps, 100-kbps, 500-kbps, and >1-Mbps bandwidths in the edge-server model.

Device D2 has a 3% feasible configuration solutions when $bw \geq$ 500 kbps.

Observation 3. When we consider the constraints on accuracy and detection frame rate instead of the costs, we choose device D5 for the minimum power consumption. Particularly, Fig. 14 shows the optimal configuration of edge-only computing and edge-server computing models when the targeted power consumption is 8 Watt. As shown in Fig. 14(a), the optimal configuration in the edge-only model is D4(90m,10f). In the edge-server model, when the network bandwidth is low (50 kbps), there is one feasible configuration (*i.e*, optimal configuration D4(90m,10f)), see Fig. 14(b). However, when the network bandwidth is >200 kbps (see Fig. 14(c) ∼ 14(e)), the number of feasible configurations increases, and the optimal configuration solutions searched by EdgeEye is also D4(90m,10f). In details, Table 5 shows the optimal configuration solutions (including the minimum detectable face size, input frame rate, selected edge device and edge computing model) that vary with the targeted power consumption under three different cost constraints and two bandwidth. There also is that the optimal solution is more inclined to the edge-only computing model under lower bandwidth, and it is more inclined to the edge-server computing model under higher bandwidth. The above analytics unveils that EdgeEye effectively provides the optimal configuration and edge computing model for video analytics, even though users have different requirements of the targeted power consumption and network bandwidth changes.

5.2.4 *Summary*

According to feasible configuration solutions based on the different targeted performance, we can find that under user's targeted accuracy, targeted detection frame rate, and targeted power consumption, EdgeEye shows good effect to solve the combiantorial optimization problem for deploying edge video analytics. EdgeEye can help system designers choose the edge computing model under different network bandwidth. There is a similar trend in feasible configuration solutions on edge devices in the edge-server and edge-only models. The edge-server model is much flexible compared to the edge-only. However, the system performance at edge-server model varies with the bandwidth, because there are a small number of detected face images and intermediate

TABLE 5
The optimal configuration solution in terms of the minimum detectable face size, input frame rate, selected edge device, and edge computing model under different targeted power consumption.

	Device Costs	Targeted Power Consumption(W)						
		6	8	10	12	14	16	18
50kbps	C<$80	/	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	**D1(90m,10f)**
	C<$150	D4(90m,10f)	**D4(90m,10f)**	D4(90m,10f)	D4(90m,10f)	D4(90m,10f)	D4(90m,10f)	**D4(90m,10f)**
	C<$750	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	**D5(90m,10f)**
500kbps	C<$80	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)
	C<$150	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)	D1(90m,10f)
	C<$750	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)	D5(90m,10f)

results that are transmitted to the edge server for face recognition. Thus, the edge-only model is much beneficial to deploying tasks compared with the edge-server model under a low-bandwidth network.

6 Related Work

Edge Intelligent. In the big-data era, a big amount of data from edge devices needs to be analyzed [31] in real-time. The traditional approaches to transmit large-scale edge data to the cloud bring network overhead [32]. The edge computing paradigm [3] migrates computing tasks to edge devices close to source data. The edge computing reduces transmission delay and network overload. However, edge device usually have limited resources; thus, deep learning based video analytics (*e.g.,* FaceNet [23], MXNet [33] and Caffe [34]) that requires much computing resources has difficulty to achieve low latency on the edge device. A variety of optimization methods for DNN models [35] [36] [37] [38] are proposed to improve latency and power consumption. DNN models partitioning between cloud and edge devices [39] [40] is applied in the edge environment. Li et al. [41] designed a device-edge on-demand collaborative inference framework (Edgent). The performance of DNN models on edge devices are studied in [42], which provided an open-source framework to determine whether edge devices meets performance requirement of intelligent video analytics.

The DDNN model [40] is a distributed deep neural network over cloud, edge, and end-users. For the inference, the DDNN model selects an exit point according to the accuracy. For light-weight samples, the inference exit point can early occur, which provides high performance. While heavy-weight samples are transferred to the cloud for high accuracy. However, the DDNN rarely considers the network status for the exit point. When the bandwidth is low, data transmission impairs the real-time performance.

Mobile Video Processing. Instead of stationary cameras in building and aside road [43], the mobile cameras can capture video stream at different location [44]. Glasses camera-enabled applications have been developed in [44], [45], [46]. Wearable glasses cameras allow users to keep working while performing a remote video service. A glasses camera can send/receive video data by a connected smart-phone, but the video data cannot be analyzed in the smart-phone [47]. Video quality is essential for applications to acquire accurate in-site information. Google Glass project lacks connectivity stability and a concise battery life [48].

Deep learning models (*e.g.,* MXNet [33] and Caffe [34]) analyze video stream in the cloud with a large amount of resources. Due to limited resources on edge devices, video data acquired from a camera is uploaded to cloud for video analytics. Nowadays, researchers [49] employed edge computing to conduct video analytics on edge devices to improves the quality of video services under harsh network condition. These lightweight object detection algorithms should be designed on edge devices.

Mobile Offloading Strategy. Mobile offloading strategies improve energy consumption of mobile edge devices. MAUI [50] designed a framework to decide when to offload tasks. Zhang et al. [51] proposed an offloading strategy based on energy consumption. Liu [52] provided a delay-optimal task scheme for a single MEC. Chen et al. [53] designed a decentralized computation offloading game for multi-user mobile cloud computing. The efficiency of computing offloading strategies depends on the networks during data transmission. Thus, offloading strategies [52] [53] [54] [55] are proposed based on the wireless status. However, the prior pieces of work rarely consider the complex parameters in the DNN model and video stream. Researchers [56] intended to reduce the delay by changing the frame rate in the video stream or deploying a compression DNN model at the edge device [27]. These existing offloading strategies should consider the impact of computing resources on edge devices.

We find that resources on edge devices, parameters in the video stream and DNN models, and constraints can affect the deployment of the DNN based video analytics. Our proposed approach focuses on the feasible optimal deployment for video analytics based on parameters in the DNN model and video stream and devices settings. We consider the user requirements in terms of the system performance and costs of edge devices. EdgeEye provides optimal configurations of the DNN model, video stream, and edge devices for deploying video analytics, which can maximize the utilization of computing resources and achieve the best overall performance in the edge video system.

7 Conclusion

In this paper, we propose a data-driven approach, EdgeEye, to address the optimal system configurations problem for deploying the DNN based video analytics in mobile edge computing systems. EdgeEye first formulates relationship models of performance (*i.e.,* accuracy, delay, and power consumption) with its independent variables (different configuration parameters). It models the problem as a combinatorial optimization problem, and designs an algorithm to find the solutions for the optimal configuration. We apply EdgeEye in the face detection and recognition application based on edge-only and edge-server computing models. Experimental results reveal that the data-driven approach can provide the optimal system configurations and choice of the optimal edge computing models to meet user's requirements of performance under different network conditions in the deployment of edge video analytics.

References

[1] P. L. Venetianer, A. J. Lipton, A. J. Chosak, M. F. Frazier, N. Haering, G. W. Myers, W. Yin, and Z. Zhang, "Video surveillance system employing video primitives," Jan. 11 2011, uS Patent 7,868,912.

[2] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 2015, pp. 426–438.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[4] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia iot systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, 2017.

[5] Q. Zhang, H. Sun, X. Wu, and H. Zhong, "Edge video analytics for public safety: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1675–1696, 2019.

[6] E. M. E. Computing, "Mobile-edge computing: Introductory technical white paper; etsi: Sophia antipolis, france, 2014," *Sensors*, vol. 17, no. 731, p. 30, 2017.

[7] Y. Wang, M. Liu, P. Zheng, H. Yang, and J. Zou, "A smart surface inspection system using faster r-cnn in cloud-edge computing environment," *Advanced Engineering Informatics*, vol. 43, p. 101037, 2020.

[8] L. Yue, W. Wanguo, X. Ronghao, L. Zengwei, and T. Yuan, "An intelligent identification and acquisition system for uavs based on edge computing using in the transmission line inspection," in *Proceedings of the 2019 4th International Conference on Robotics, Control and Automation*, 2019, pp. 205–209.

[9] B. Luo, S. Tan, Z. Yu, and W. Shi, "Edgebox: Live edge video analytics for near real-time event detection," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 347–348.

[10] G. Grassi, K. Jamieson, P. Bahl, and G. Pau, "Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1–14.

[11] Y. Wang, S. Liu, X. Wu, and W. Shi, "Cavbench: A benchmark suite for connected and autonomous vehicles," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 30–42.

[12] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2980–2988.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[14] D. N. Parmar and B. B. Mehta, "Face recognition methods & applications," *arXiv preprint arXiv:1403.0485*, 2014.

[15] M. Haghighat, S. Zonouz, and M. Abdel-Mottaleb, "Cloudid: Trustworthy cloud-based and cross-enterprise biometric identification," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7905–7916, 2015.

[16] W. Wang, A. X. Liu, and M. Shahzad, "Gait recognition using wifi signals," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 363–373.

[17] C. S. S. Prasanna, N. Sudha, and V. Kamakoti, "A principal component neural network-based face recognition system and asic implementation," in *18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*. IEEE, 2005, pp. 795–798.

[18] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong, "Fp-dnn: An automated framework for mapping deep neural networks onto fpgas with rtl-hls hybrid templates," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2017, pp. 152–159.

[19] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-m. Hwu, and D. Chen, "Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[20] X. Hu, L. Liang, S. Li, L. Deng, P. Zuo, Y. Ji, X. Xie, Y. Ding, C. Liu, T. Sherwood *et al.*, "Deepsniffer: A dnn model extraction framework based on learning architectural hints," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 385–399.

[21] H. Sun, Y. Yu, K. Sha, and B. Lou, "mvideo: Edge computing based mobile video processing systems," *IEEE Access*, vol. 8, pp. 11 615–11 623, 2019.

[22] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[23] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[24] A. Rosebrock, "Intersection over union (iou) for object detection," *Online] http://www. pyimagesearch. com/2016/11/07/intersection-overunion-iou-for-objectdetection*, 2016.

[25] G. Ananthanarayanan, P. Bahl, P. Bodik, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[26] P. Korshunov and W. T. Ooi, "Video quality for face detection, recognition, and tracking," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 7, no. 3, pp. 1–21, 2011.

[27] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 1421–1429.

[28] P. Hintjens, *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[29] Xiph.org, "Xiph.org video test media [derf's collection]," https://media.xiph.org/video/derf/, accessed Sepeteber, 2019.

[30] S. S. Bert Hubert, Jacco Geul, "The wonder shaper 1.4.1," https://github.com/magnific0/wondershaper.

[31] M. Ge, H. Bangui, and B. Buhnova, "Big data for internet of things: A survey," *Future Generation Computer Systems*, vol. 87, pp. 601 – 614, 2018.

[32] A. Khattab, A. Abdelgawad, and K. Yelmarthi, "Design and implementation of a cloud-based iot scheme for precision agriculture," in *2016 28th International Conference on Microelectronics (ICM)*. IEEE, 2016, pp. 201–204.

[33] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.

[34] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[35] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[36] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015.

[37] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4820–4828.

[38] N. D. Lane, S. Bhattacharya, A. Mathur, C. Forlivesi, and F. Kawsar, "Dxtk: Enabling resource-efficient deep learning on mobile and embedded devices with the deepx toolkit." in *MobiCASE*, 2016, pp. 98–107.

[39] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1. ACM, 2017, pp. 615–629.

[40] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[41] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proceedings of the 2018 Workshop on Mobile Edge Communications*. ACM, 2018, pp. 31–36.

[42] X. Zhang, Y. Wang, S. Lu, L. Liu, L. Xu, and W. Shi, "Openei: An open framework for edge intelligence," *arXiv preprint arXiv:1906.01864*, 2019.

[43] H. Sun, X. Liang, and W. Shi, "Vu: Video usefulness and its application in large-scale video surveillance systems: An early experience," in *Proceedings of the Workshop on Smart Internet of Things*, ser. SmartIoT '17. New York, NY, USA: ACM, 2017, pp. 6:1–6:6. [Online]. Available: http://doi.acm.org/10.1145/3132479.3132485

[44] R. Schaer, T. Melly, H. Müller, and A. Widmer, "Using smart glasses in medical emergency situations, a qualitative pilot study." in *Wireless Health*, 2016, pp. 54–58.

[45] A. Widmer, R. Schaer, D. Markonis, and H. Müller, "Facilitating medical information search using google glass connected to a content-based medical image retrieval system," in *engineering in medicine and biology society (embc), 2014 36th annual international conference of the IEEE*. IEEE, 2014, pp. 4507–4510.

[46] U.-V. Albrecht, U. Von Jan, J. Kuebler, C. Zoeller, M. Lacher, O. J. Muensterer, M. Ettinger, M. Klintschar, and L. Hagemeier, "Google glass for documentation of medical findings: evaluation in forensic medicine," *Journal of medical Internet research*, vol. 16, no. 2, 2014.

[47] T. Elgamal, B. Chen, and K. Nahrstedt, "Teleconsultant: Communication and analysis of wearable videos in emergency medical environments," in *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 2017, pp. 1241–1242.

[48] A. Widmer and H. Müller, "Using google glass to enhance prehospital care," *Swiss Medical Informatics*, vol. 30, pp. 1–4, 2014.

[49] P. Liu, B. Qi, and S. Banerjee, "Edgeeye: An edge service framework for real-time intelligent video analytics," in *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, ser. EdgeSys'18. New York, NY, USA: ACM, 2018, pp. 1–6. [Online]. Available: http://doi.acm.org/10.1145/3213344.3213345

[50] E. Cuervo, A. Balasubramanian, D. K. Cho, A. Wolman, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys 2010), San Francisco, California, USA, June 15-18, 2010*, 2010.

[51] S. Zhang, N. Zhang, S. Zhou, J. Gong, Z. Niu, Xuemin, and Shen, "Energy-aware traffic offloading for green heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1116–1129, 2016.

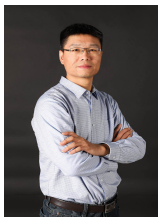
[52] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 1451–1455.

[53] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2014.

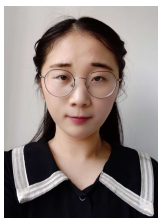
[54] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

[55] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2014.

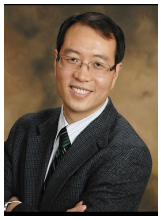
[56] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 155–168.



Hui Sun received the PhD degree from Huazhong University Science and Technology in 2014. He is an Associate Professor of Computer Science with Anhui University. His research interests include computer systems, edge computing, performance evaluation, Non-Volatile Memory-based storage systems, file systems, and I/O architectures.



Ying Yu born in 1996. She is currently pursuing the M.S. degree in Anhui University. Her main research interests include edge computing, computer systems, intelligent video analytics, and storage systems.



Kewei Sha is an Associate Professor of Computer Science and Associate Director of Cyber Security Institute at University of Houston-Clear Lake (UHCL). Before he moved to UHCL, he was the Department Chair and Associate Professor in the Department of Software Engineering at Oklahoma City University (OCU). He received Ph.D. in Computer Science from Wayne State University in 2008. His research interests include Internet of Things, Cyber-Physical Systems, Edge Computing, Security and Privacy, and Data Management and Analytics. His research has been supported by NSF, NASA, UHCL and OCU. Dr. Sha has served as the secretary of Technical Committee on the Internet of the IEEE Computer Society (IEEE-CS TCI), an Associate Editor of IEEE IoT Journal, Elsevier Smart Health and Springer Computing, a Guest Editor at several prestigious international journals, and an organizing committee member of many conferences, including the General Chair of IEEE/ACM CHASE 2021, the TPC Chair of IEEE ICCCN 2015, the workshop general chair of IEEE ICCCN, and the workshop founder and Chair of MobiPST and MedSPT. He is a Senior member of both ACM and IEEE.

Hong Zhong received the B.S. degree in applied mathematics from Anhui University, Hefei, China, in 1986, and the Ph.D. degree in computer science and technology from the University of Science and Technology of China, Hefei, in 2005. She is currently a Professor and a Ph.D. Advisor with Anhui University. Her current research interests include applied cryptography, Internet of Things (IoT) security, vehicular Ad-Hoc network, and software-defined networking (SDN).