Adaptive Configuration Selection and Bandwidth Allocation for Edge-Based Video Analytics

Sheng Zhang, *Member, IEEE*, Can Wang, Yibo Jin, *Graduate Student Member, IEEE*, Jie Wu, *Fellow, IEEE*, Zhuzhong Qian, *Member, IEEE*, Mingjun Xiao, *Member, IEEE*, and Sanglu Lu, *Member, IEEE*

***Abstract*— Major cities worldwide have millions of cameras deployed for surveillance, business intelligence, traffic control, crime prevention, etc. Real-time analytics on video data demands intensive computation resources and high energy consumption. Traditional cloud-based video analytics relies on large centralized clusters to ingest video streams. With edge computing, we can offload compute-intensive analysis tasks to nearby servers, thus mitigating long latency incurred by data transmission via wide area networks. When offloading video frames from the front-end device to an edge server, the application configuration (i.e., frame sampling rate and frame resolution) will impact several metrics, such as energy consumption, analytics accuracy and user-perceived latency. In this paper, we study the configuration selection and bandwidth allocation for multiple video streams, which are connected to the same edge node sharing an upload link. We propose an efficient online algorithm, called JCAB, which jointly optimizes configuration adaption and bandwidth allocation to address a number of key challenges in edge-based video analytics systems, including edge capacity limitation, unknown network variation, intrusive dynamics of video contents. Our algorithm is developed based on Lyapunov optimization and Markov approximation, works online without requiring future information, and achieves a provable performance bound. We also extend the proposed algorithms to the multi-edge scenario in which each user or video stream has an additional choice about which edge server to connect. Extensive evaluation results show that the proposed solutions can effectively balance the analytics accuracy and energy consumption while keeping low system latency in a variety of settings.**

***Index Terms*— Bandwidth allocation, configuration selection, edge computing, Lyapunov optimization, video analytics.**

I. Introduction

A LARGE number of cameras have been deployed for various purposes, such as business intelligence, traffic control, and crime prevention [1]. These cameras generate a large amount of video data every second. Quick analysis on these live video streams is often required. In addition, many other emerging applications such as cognitive assistance, mobile gaming, virtual reality and augmented reality [2], [3] also rely on effective analysis of videos with low latency.

Video analytics applications usually require intensive computation resources and high energy consumption. Thus the front-end devices are often resource-limited to support these applications with acceptable latency. One way to overcome this limitation is to transfer videos to cloud data centers and execute the deep learning algorithms there [4]. However, cloud-based solutions may incur excessive transmission delay in wide area networks [5]. Edge computing is an emerging computing paradigm which advocates processing data at the logical edge of a network [6]–[12], thereby enabling video analytics to occur closer to the data source and end users.

In typical video analytics, frames are extracted from a video at different sampling rates, compressed into various resolutions, and then processed by different CNN (convolutional neural network [13]) models. Following previous works [14]–[16], we refer to a combination of a resolution and a frame rate as a configuration [14]. Apparently, different configurations often lead to different accuracies and energy consumptions. Since edge nodes serve as the backend for video processing [17], transmitting videos from their sources to an edge server via time-varying network links is inevitable. Thus, efficient offloading of video analytics involves *configuration selection* and *bandwidth allocation*.

Both industry and academia have invested heavily in these two aspects. Most of the previous works [3], [5], [14], [18]–[22] considered only one of them in offloading video analytics, which leads to sub-optimal performance. However, the joint scheduling of computing resources (via configuration selection) and networking resources (via bandwidth allocation) are both of importance to the overall performance of edge-assisted video analytics.

In this paper, we first consider a practical scenario in which multiple video streams connect to the same edge server sharing a narrow uplink channel, as shown in Fig. 1. We call this scenario the single-edge scenario. Different CNN models are deployed on the single edge server to match various video resolutions. A smaller CNN model with fewer convolutional layers is cheaper, faster but less accurate [1]. We then consider the multi-edge scenario in which each video stream has an additional choice about which edge server to connect. The objective in both scenarios is to decide the frame rate, resolution, and the share of bandwidth for each video stream to maximize the overall accuracy and minimize the energy

Manuscript received July 16, 2020; revised January 12, 2021 and June 24, 2021; accepted August 19, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor B. Shrader. This work was supported in part by the National Key Research and Development Program under Grant 2017YFB1001801, in part by the NSFC under Grant 61872175 and Grant 61832008, and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization. *(Corresponding author: Sheng Zhang.)*

Sheng Zhang, Can Wang, Yibo Jin, Zhuzhong Qian, and Sanglu Lu are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: sheng@nju.edu.cn; mf1733063@smail.nju.edu.cn; yibo.jin@smail.nju.edu.cn; qzz@nju.edu.cn; sanglu@nju.edu.cn).

Jie Wu is with the Center for Networked Computing, Temple University, Philadelphia, PA 19122 USA (e-mail: jiewu@temple.edu).

Mingjun Xiao is with the School of Computer Science and Technology/Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei 230026, China (e-mail: xiaomj@ustc.edu.cn).

Digital Object Identifier 10.1109/TNET.2021.3106937

consumption, subject to a service latency budget. This problem faces many challenges for the following reasons:

A. The Best Offloading Configuration Varies Over Time

As we mentioned above, different configurations lead to different accuracies and energy consumptions. We can keep choosing the most expensive configuration to ensure a high accuracy, but this demands more resources and energy. In many cases, the policy that reduces the frame rate and lowers the resolution can save energy significantly without impacting the accuracy. For example, we can choose a lower frame sampling rate when the target moves slowly. Meanwhile, the policy that lowers frame resolution will not hurt accuracy when the target is large in the scenes [5]. The best configuration can optimize the trade-off between accuracy and energy consumption, which varies over time depending on the video content.

B. Network Bandwidth Is Often Unpredictable

As many previous works have observed, the wide area network bandwidth has come to a standstill [23] while traffic demands are growing at a staggering rate [24]. Not only is WAN bandwidth scarce, it is also relatively expensive, and highly variable [5]. Similar scarcity and variations exist in wireless networks [25], [26], broadband access networks and cellular networks [27]. When the available bandwidth becomes insufficient, an offloading configuration which adopts high frame resolution may incur long transmission latency, and this problem becomes more noticeable when multiple video streams have to share the same uplink channel, hence bandwidth resource management becomes crucial and the main challenge is to deal with the trade-off between analytics accuracy and bandwidth consumption.

These observations motivate us to propose *adaptive video analytics which is capable of optimizing the trade-off among the analytics accuracy, service latency, and energy consumption*. We aim to find the most suitable video analytics offloading configuration and bandwidth allocation scheme, subject to the computing capacity, bandwidth capacity, and the delay constraint, for a multi-user edge-assisted video analytics system. Since the Lyapunov optimization framework [28] is the de facto standard method to achieve stability in control theory and is capable of minimizing the time average of a network attribute subject to additional time average constraints, in this paper, we utilize the Lyapunov framework to transform the original problem into a series of one slot optimization problems, each of which is solved by leveraging the Markov approximation and the KKT condition. We prove that our solution achieves a close-to-optimal performance, while bounding the potential violation of the latency constraint.

To our best knowledge, this is the *first* study to jointly optimize configuration adaption and bandwidth allocation for multi-video in the edge environment, explicitly taking into account the trade-off among the analytics accuracy, service latency, and energy cost. The main contributions of this paper are summarized as follows.

- We study a more practical model. We explicitly consider limited and varying bandwidths between video sources and edge servers. Each edge server has limited computing resource. The accuracy function with respect to resolution or frame rate varies depending on the video contents. Both transmission energy consumption and processing energy consumption are taken into account.
- We formalize the joint configuration selection and bandwidth allocation problem, for optimizing the trade-off between accuracy and energy cost, under a long-term latency constraint. The insight behind our problem is adapting video streams to bandwidth variation and intrinsic dynamics of their contents.
- We develop novel online algorithms, i.e., JCAB and mJCAB, which can efficiently adapt configurations and allocate bandwidth resources for video streams on the fly without foreseeing the future. Both algorithms utilize the Lyapunov framework to transform the original problem into a series of one slot optimization problems, each of which is solved by leveraging the Markov approximation and the KKT condition. We prove that JCAB and mJCAB achieve a close-to-optimal performance, while bounding the potential violation of the latency constraint.
- We evaluate the performance of the design through extensive and practical simulations with accuracy profiles obtained from our experiments. Results confirm the superiority of our approach compared to several algorithms.

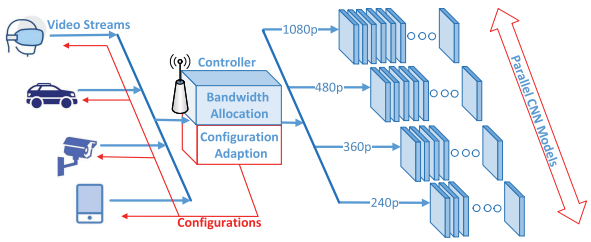


Fig. 1. An illustration of the edge-assisted video analytics system. Different CNN models are deployed on the single edge server to match various video resolutions.

The remainder of this paper is organized as follows. Section II describes our system model. Section III develops the JCAB algorithm. Section IV shows how to deal with the multi-edge scenario. We evaluate our proposed design in Section V. Section VI reviews the related work. Section VII discusses the limitations and future work. Finally, Section VIII concludes the paper.

II. System Model

We introduce the system model and present the problem formulation for the single-edge scenario in this section.

Suppose that a set of K users or video streams, denoted by $\mathcal{U} = \{u_1, u_2, \ldots, u_K\}$, connect to the same edge server nearby. They keep offloading video frames to the server, sharing a narrow uplink channel. As illustrated in Fig. 1, there are N parallel CNN models $\mathcal{M} = \{m_1, m_2, \ldots, m_N\}$ deployed on the edge server, with different input sizes of images. Let m_0 be the most lightweight CNN model in each local device. Let r_i be the input resolution for the ith CNN m_i.

We use s_i and c_i to denote the processing time and cost, respectively, per frame for the ith CNN m_i. It has been well studied in [29] that a CNN can be compressed to a smaller size at the expense of accuracy. Such techniques include removing some expensive convolutional layers and reducing input image resolution. Thus in our design, a CNN with a lower input image resolution has a faster processing speed (i.e., smaller s_i) and needs less computational resources (i.e., smaller c_i).

We divide time into discrete time slots, each of which has a duration that matches the timescale at which offloading

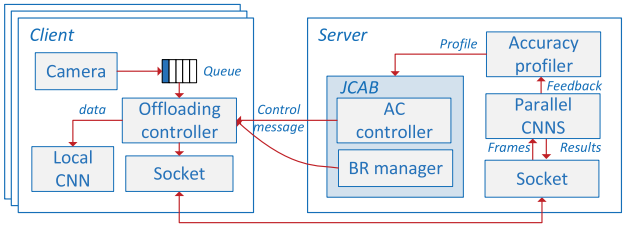


Fig. 2. The architecture of the edge-based live video analytics system.

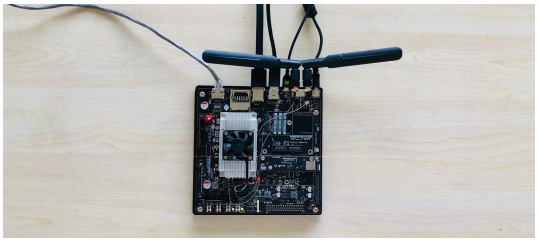
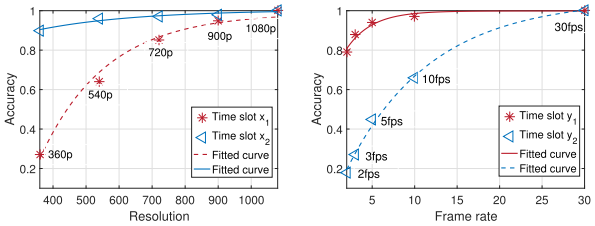


Fig. 3. We implement YOLOv3 on NVIDIA Jetson TX2.



(a) Resolution (b) Frame Rate

Fig. 4. Impact of offloading configurations on the detection accuracy.

configurations can be updated. The system architecture is depicted in Fig. 2. We mainly focus on video queries such as detecting certain objects (like cars or pedestrians). On the client side, videos are continuously recorded from cameras and object recognition can be performed locally using lightweight CNNs, i.e., m_0's. On the server side, the proposed algorithm runs in the Adaptive Configuration (AC) controller and the Bandwidth Resource (BR) manager. The former is responsible for computing a configuration for the next time slot, given accuracy profiles,[1] network conditions and latency goals as the input; while the latter computes and sends bandwidth allocation back to the clients.

In the remainder of this section, we first provide analytical models on the accuracy (Section II-A), the energy consumption (Section II-B), and the latency (Section II-C). Then, we present the problem formulation (Section II-D).

A. Analytics Accuracy Model

Since we want to optimize the time-averaged accuracy through CNN model selection and bandwidth allocation, first of all, we have to know how the model selection and bandwidth allocation affect accuracy. In this subsection, we provide the analytics accuracy models, which captures the relationship between model selection (i.e., resolution) or bandwidth allocation (i.e., frame rate) and accuracy. These models are derived based on the performance measurements obtained from our real experiments implemented on NVIDIA Jetson TX2.

Since a query configuration is multi-dimensional and different decision variables may affect the analytics accuracy in different ways, profiling the accuracy of a configuration is no easy task. We first have to figure out the relationship between the analytics accuracy and the input image resolution. To do so, we implement YOLO [30], an object detector CNN on NVIDIA Jetson TX2 (shown in Fig. 3) to perform pedestrian detection on a clip from a surveillance video. In this experiment, video frames are resized to different resolutions, and the accuracy of a compressed frame is computed by comparing the detected objects with the objects detected in the frame with the highest resolution, using the F1 score, which is the harmonic mean of precision and recall. A detected object is identified as true positive when its bounding box has the same label and it has sufficient spatial overlap with the corresponding ground truth [14]. The spatial overlap can be measured by IOU (Interaction over Union). In our experiment, an object is correctly detected when IOU is no less than 0.7.

The results are plotted in Fig. 4(a). The red dashed line shows the detection accuracy when the targets are small in the scene in time slot x_1, while the blue line shows the detection accuracy when pedestrians walk nearby in time slot x_2. There are two observations.

The first observation is that a higher resolution produces a better analytics accuracy and the performance gain decreases at a high resolution. Hence the relationship between accuracy and resolution can be formulated as concave functions, e.g., the red line in Fig. 4(a) can be fitted as $0.988 - 4.469e^{-\frac{r}{200}}$ with less than 0.02 root mean square error.

The second observation is that the accuracy profile of a video stream varies over time: high resolution is crucial when targets are small, but the policy to lower resolution will not hurt latency much when targets are big enough. The accuracy models should be updated periodically according to target size. Based on these observations, we use $\epsilon_k^t(r)$ to represent the accuracy function with respect to resolution for user u_k in time slot t. We also introduce a binary variable $x_{k,t}^i$ to indicate whether the i-th CNN model m_i is selected by user u_k in slot t, so $\sum_{i=0}^{N} x_{k,t}^i r_i$ is the frame resolution of u_k in slot t.

The relationship between accuracy and the sampling frame rate f is illustrated in Fig. 4(b). We perform cars counting on a clip from a traffic video with different sampling frame rates. Since the video segment consists of many frames, we compute accuracy as the fraction of frames with F1 score $\geq$ 0.67. To compute the accuracy of a frame that was not sampled, we use the location of objects from the previous sampled frame. In time slot y_2, the cars in the scene are moving fast, while in slot y_1, all cars slow down due to a traffic congestion. Similarly, we model the accuracy function with respect to frame rate as a concave function $\phi_k^t(f)$, which should be updated at the start of each time slot according to the velocities of targets. These observations are consistent with many previous studies [1], [14], in which the relationship between resolution or frame sampling rate and accuracy can often be formulated as concave functions.

It has been experimentally observed in [14] that frame resolution and frame sampling rate independently impact accuracy, allowing us to model the accuracy of the configuration of u_k in time slot t as

$$\epsilon_k^t\Big(\sum_{i=0}^{N} x_{k,t}^i r_i\Big)\phi_k^t(f_{k,t}), \tag{1}$$

[1]We use 'accuracy profile' to represent the content-varying relationship between the detection accuracy by a CNN model and the frame rate and/or resolution of a video stream.

in which $f_{k,t}$ is the frame sampling rate of u_k in time slot t. Then, the average detection accuracy over K users in time slot t is

$$a_t = \frac{1}{K}\sum_{k=1}^{K}\epsilon_k^t(\sum_{i=0}^{N}x_{k,t}^i r_i)\phi_k^t(f_{k,t}). \tag{2}$$

B. Energy Consumption Model

Battery life may become the primary concern of end users since it is usually inconvenient to recharge mobile devices. Therefore, we take energy efficiency into consideration. The energy consumption of a mobile device or smart camera mainly consists of two parts: transmission energy due to data transmission and processing energy caused by local video frame processing.

The transmission energy consumption is proportional to the size of data which is uploaded to an edge server. The data size of a video frame with resolution r is calculated as αr^2 bits [3], where α is a constant. Let γ_k represent the transmission energy consumption per bit for u_k. Then the transmission energy consumption of user u_k in slot t is

$$e_{k,t}^{tran} = \sum_{i=1}^{N}\gamma_k\alpha(x_{k,t}^i r_i)^2 f_{k,t}. \tag{3}$$

We use μ_k to denote the energy cost of processing one frame on the local device of u_k [31]. Then the data processing energy cost for user u_k in time slot t is

$$e_{k,t}^{proc} = x_{k,t}^0\mu_k f_{k,t}. \tag{4}$$

Combining Eqs. (3) and (4) together, we know that the average energy consumption of all users in time slot t is

$$e_t = \frac{1}{K}\sum_{k=1}^{K}(e_{k,t}^{tran} + e_{k,t}^{proc}). \tag{5}$$

C. Service Latency Model

The latency per frame consists of two parts: data transmission latency and CNN processing latency. The data transmission latency is jointly determined by the data size of a frame and the share of the upload bandwidth; we use $b_{k,t}$ to denote the upload bandwidth shared by u_k at time slot t. Then, the latency per frame experienced by u_k in time slot t is

$$l_{k,t} = \frac{\sum_{i=1}^{N}\alpha(x_{k,t}^i r_i)^2}{b_{k,t}} + \sum_{i=0}^{N}x_{k,t}^i s_i, \tag{6}$$

where the first term is the transmission latency and the second one is the processing latency. Thus, the average latency for K video streams in slot t is

$$l_t = \frac{1}{K}\sum_{k=1}^{K}l_{k,t}. \tag{7}$$

Main notations are summarized in Tab. I.

D. Problem Formulation

Analytics on live video streams is energy-consuming and latency-sensitive, generally requiring high quality. Hence on designing the adaptive algorithm, we aim at achieving desirable analytics accuracy under the long-term latency constraint, while keeping the energy cost as low as possible. For simplicity of illustration, we define the utility function of a configuration as the achieved accuracy minus the energy cost. The natural objective is the maximum of time-averaged utility for all video streams, which can be formulated as

$$\begin{aligned}
\mathcal{P}: \max_{\{x,b,f\}} \lim_{T\to+\infty}\frac{1}{T}\sum\nolimits_{t=0}^{T}(a_t - \omega e_t)\\
s.t.\ C_1: \sum_{i=0}^{N}x_{k,t}^i = 1, \quad \forall u_k\in\mathcal{U}, t\in\mathcal{T}\\
C_2: x_{k,t}^i\in\{0,1\}, \quad \forall u_k\in\mathcal{U},\\
\qquad t\in\mathcal{T}, m_i\in\mathcal{M}\cup\{m_0\}\\
C_3: f_{k,t}\le\sum_{i=0}^{N}x_{k,t}^i\frac{1}{s_i}, \quad \forall u_k\in\mathcal{U}, t\in\mathcal{T}\\
C_4: \sum_{k=1}^{K}\sum_{i=1}^{N}x_{k,t}^i c_i\le C, \quad \forall t\in\mathcal{T}\\
C_5: \sum_{k=1}^{K}b_{k,t}\le B_t, \quad \forall t\in\mathcal{T}\\
C_6: \lim_{T\to+\infty}\frac{1}{T}\sum_{t=0}^{T}l_t\le L_{max}.
\end{aligned} \tag{8}$$

The weighted parameter ω controls the trade-off between accuracy and energy consumption. As a result, the optimal solution of Problem $\mathcal{P}$ trades the average accuracy for lowering the energy consumption on mobile devices. Constraints C_1 and C_2 ensure that, in each time slot, one and only one CNN model can be selected by user u_k. Constraint C_3 says that the selected frame rate cannot exceed the processing frequency of the CNN model (remote or local), otherwise video frames would accumulate and lead to a long queue delay. The fourth constraint C_4 is due to the capacity of the edge server, denoted as C. In this paper, we can estimate the available bandwidth of the upload link to the edge server at the beginning of each time slot, and we assume that *the bandwidth does not change in each single time slot*. In fact, this assumption can hold in many scenarios. Even if the bandwidth changes frequently in some extreme case, we can reduce the length of each slot to mitigate the effect of the bandwidth change on the performance. Let B_t represent the uplink bandwidth over the entire slot t, constraint C_5 imposes per-slot constraint on the available bandwidth. The last constraint C_6 requires that the long-term average latency not exceed the threshold L_{max}.

The first major challenge that impedes the derivation of the optimal solution to the above problem is the lack of future information. To optimally solve problem $\mathcal{P}$, near future information about the network condition and the dynamics of video contents is required, which is difficult to accurately predict in advance. Moreover, $\mathcal{P}$ is a mixed integer nonlinear programming and is very difficult to solve even if the future information is known a priori. These challenges call for an online approach that can efficiently adapt configurations and allocate bandwidth resources for video streams on the fly without foreseeing the future.

III. Online Algorithm for the Single-Edge Scenario

To decouple the long-term latency constraint, we transform the original time-averaged problem into a series of minimization problems leveraging the Lyapunov framework [28], and then we develop a lightweight online algorithm which only

TABLE I
MAIN NOTATIONS FOR QUICK REFERENCE

Symbol	Meaning	Symbol	Meaning
u_k	k-th user or video stream	$\epsilon_k^t(r)$	the accuracy function wrt. resolution r for u_k in slot t
m_i	i-th CNN model	$x_{k,t}^i$	indicating whether m_i is selected by user u_k in slot t
m_0	the local CNN model	$\phi_k^t(f)$	the accuracy function wrt. frame rate f for u_k in slot t
r_i	the input resolution for m_i	μ_k	the energy cost of processing one frame by m_0 locally
s_i	the processing time for m_i	γ_k	the transmission energy cost per bit for u_k
c_i	the computational cost for m_i	$e_{k,t}^{tran}$	the transmission energy cost of user u_k in slot t
C	the computational capacity of each server	$e_{k,t}^{proc}$	the data processing energy cost of user u_k in slot t
L_{max}	the long-term average latency threshold	e_t	the average energy cost over K users in time slot t
$f_{k,t}$	the frame sampling rate of u_k in slot t	$b_{k,t}$	the upload bandwidth shared by u_k at time slot t
a_t	the average accuracy of K users in slot t	B_t	the uplink bandwidth over the entire time slot t
l_t	the average latency of K users in slot t	$l_{k,t}$	the latency per frame experienced by u_k in time slot t

relies on the current bandwidth information and video content to derive the adaptation strategy, without global information over the long run.

A. JCAB

A major challenge of directly solving Problem $\mathcal{P}$ (Eq. (8)) is that the long-term latency constraint couples CNN model selection, frame rate adaption and bandwidth allocation across different slots. To address this challenge, we define a virtual queue $q(t)$ as a historical measurement of the latency overflow and assume that the initial queue backlog is 0 (i.e., $q(0) = 0$). Since the queue is used to capture the latency overflow in each slot, it is not hard to see that its length evolves as follows:

$$q(t+1) = [q(t) + \frac{1}{K}\sum_{k=1}^{K} l_{k,t} - L_{max}]^+, \tag{9}$$

where $[x]^+$ denotes the maximum among x and 0. From Eq. (9), we see that the queue backlog at time slot $(t+1)$ is the sum of the backlog at slot t and the latency overflow $(\frac{1}{K}\sum_{k=1}^{K} l_{k,t} - L_{max})$ at the current time slot. Without loss of generality, we assume that the arrivals (i.e., $l_t = \frac{1}{K}\sum_{k=1}^{K} l_{k,t}$) for the queue backlog are independent and identically distributed (i.i.d) over slots.

If we aggressively pursue high accuracy by adopting the most expensive configuration, the queue backlog $q(t)$ will increase unboundedly, leading to unacceptable delays and poor user experience, so it is crucial to keep the latency queue stable. We first define a quadratic Lyapunov function as follows:

$$L(q(t)) = \frac{1}{2}(q(t))^2, \tag{10}$$

which represents a scalar measure of latency queue congestion. For instance, a small value of $L(q(t))$ implies that the queue backlog is small.

To keep queue stability by persistently pushing the Lyapunov function towards a less congested state, we introduce the one-slot Lyapunov drift as

$$\Delta(q(t)) = E[L(q(t+1)) - L(q(t))|q(t)], \tag{11}$$

in which $E[x]$ denotes the expectation of variable x.

The drift $\Delta(q(t))$ denotes the expected change in the Lyapunov function over one time slot, given the current state in time slot t. A smaller $\Delta(q(t))$ implies that the virtual queue has strong stability. Our goal is to find the optimal adaption strategy for all video streams to coordinate the network condition, taking the variation of video contents into consideration as well. By incorporating latency queue stability into the trade-off between accuracy and energy cost, we define a Lyapunov drift-plus-penalty term as

$$\Delta(q(t)) - V \cdot E[a_t - \omega e_t|q(t)]. \tag{12}$$

The positive parameter V is used to adjust the trade-off between latency minimization and utility maximization. Rather than directly minimizing the drift-plus-penalty term in each slot, the min-drift-plus-penalty algorithm [32] in Lyapunov optimization seeks to minimize an upper bound of it. We derive an upper bound on the drift-plus-penalty term in our specific problem and it is stated in the following lemma:

Lemma 1: Let the largest average delay of all video streams in all time slots be $l_{max} = \max_{t\in\mathcal{T}}\{l_t\}$, then $B = \frac{1}{2}(l_{max} - L_{max})^2$ is a constant, which implies the second moments of arrivals and service are bounded (i.e., $\forall t$, $E[\frac{1}{2}(l_t - L_{max})^2|q(t)] \leq B$). For all possible values of $q(t)$ by using any offloading configuration over all time slots, the following statement holds:

$$\begin{aligned}&\Delta(q(t)) - V \cdot E[a_t - \omega e_t|q(t)]\\ &\leq B + q(t)E[(l_t - L_{max})|q(t)] - V \cdot E[a_t - \omega e_t|q(t)].\end{aligned} \tag{13}$$

Proof: From Eq. (11), we have

$$\begin{aligned}\Delta(q(t)) &= E[L(q(t+1)) - L(q(t))|q(t)]\\ &= \frac{1}{2}E[q^2(t+1) - q^2(t)|q(t)]\\ &\leq \frac{1}{2}E[(q(t) + l_t - L_{max})^2 - q(t)^2|q(t)]\\ &= \frac{1}{2}(l_t - L_{max})^2 + q(t)E[(l_t - L_{max})|q(t)]\\ &= B + q(t)E[(l_t - L_{max})|q(t)].\end{aligned} \tag{14}$$

We now incorporate the expected utility over one time slot to both sides of Eq. (14), then we have

$$\begin{aligned}&\Delta(q(t)) - V \cdot E[a_t - \omega e_t|q(t)] \leq B\\ &\quad + q(t)E[(l_t - L_{max})|q(t)] - V \cdot E[a_t - \omega e_t|q(t)].\end{aligned} \tag{15}$$

The lemma follows immediately. □

Then we attempt to minimize the supremum bound for the drift-plus-penalty function, and the new real-time optimization problem can be presented as follows:

$$\begin{aligned}\mathcal{P}_1: \min_{\{x,b,f\}} \quad & q(t) \cdot l_t - V \cdot (a_t - \omega e_t)\\ s.t.\ & C_1, C_2, C_3, C_4, C_5.\end{aligned} \tag{16}$$

Notice that solving $\mathcal{P}_1$ requires only currently available information as the input. By considering the additional term $q(t) \cdot l_t$, the system takes into account the average latency incurred by data transmission and processing in the current

Algorithm 1: JCAB for the Single-Edge Scenario

Input: $q(0) \leftarrow 0$, μ_k, γ_k, L_{max}, C;
for $t = 0$ *to* $T-1$ **do**
  for $k = 1$ *to* K **do**
    Based on the first few frames at slot t, choose the most appropriate $\epsilon_k^t(r)$ and $\phi_k^t(f)$ from a set of pre-defined functions;
  Obtain $\{x_t, f_t, b_t\}$ by solving $\mathcal{P}_1$;
  $q(t+1) \leftarrow [q(t) + l_t - L_{max}]^+$;

slot. As a consequence, when $q(t)$ is large, minimizing the latency is more critical. Thus, our algorithm works by following the philosophy of "when bandwidth becomes insufficient, degrade the configuration to avoid violating the latency constraint". The latency queue is maintained without future information, guiding the configuration adaption and bandwidth allocation to follow the long-term latency constraint, thereby enabling online decision making.

We develop the JCAB algorithm (shown in Alg. 1) to solve Problem $\mathcal{P}$. In our online control algorithm JCAB, we divide time into T discrete time slots. At the beginning of each time slot, based on the first few frames, JCAB chooses the most appropriate $\epsilon_k^t(r)$ and $\phi_k^t(f)$ from a set of pre-defined functions. These pre-defined functions can be obtained as follows: we roughly divide targets into several categories according to their relative sizes in the video, and we profile several accuracy functions with respect to frame resolution under different conditions; similarly, different accuracy functions with respect to frame rate are drawn when targets move at different speed levels. Note that this heuristic method is not always accurate but it is flexible and quick, while the profiling cost is relatively low. Given the accuracy functions, we can find a good configuration and bandwidth allocation scheme for the current slot by solving problem $\mathcal{P}_1$. Finally, the average latency is utilized to update the virtual queue.

B. Theoretic Analysis of JCAB

In this subsection, we analyses the performance of JCAB. We first introduce the following lemma:

Lemma 2: For any $\delta > 0$, there exists a stationary and randomized policy Π for $\mathcal{P}$, which decides b_t^Π, x_t^Π and f_t^Π independent of the current queue backlogs $q(t)$, such that the following inequalities are satisfied:

$$E[l_t^\Pi - L_{max}] \leq \delta, \tag{17}$$

and

$$E[a_t^\Pi - \omega e_t^\Pi] \leq \nu_t^* + \delta, \tag{18}$$

in which ν_t^* is used as the optimum for instantaneous sub-problem per slot.

Proof: We prove this based on Theorem 4.18 in [28].

The evolution of the model selection, frame rate selection, and bandwidth allocation from time 0 to $(T-1)$ can be seen as a walk in the solution space of Problem $\mathcal{P}$. According to Theorem 4.18 in [28], we know the stationary and randomized policy Π is in the closure of the optimal solution. If it is closed, then Eqs (17) and (18) hold with $\delta = 0$. If the optimal solution is not closed, then Π is a limit point of the optimal solution, yielding Eqs (17) and (18) hold with $\delta > 0$. □

Now we can present the performance bound of JCAB.

Theorem 1: JCAB achieves the following performance bounds on the time-averaged utility and service latency:

$$\lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T} E[a_t - \omega e_t] \geq \nu_{opt} - \frac{B}{V}, \tag{19}$$

$$\lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T} E[l_t] \leq \frac{B}{\varepsilon} + \frac{V}{\varepsilon}(\nu_{opt} - \nu_{min}) + L_{max}, \tag{20}$$

where ν_{min} is the objective value of the worst solution for $\mathcal{P}$, ν_{opt} is the optimal utility of Problem $\mathcal{P}$ that can be achieved by ignoring the delay constraint, and ε>0 is a constant which represents the long-term latency surplus achieved by some stationary control strategy.

Proof: Recall that JCAB seeks to choose strategies that minimize $\mathcal{P}_1$ among the feasible decisions including the policy in Lemma 2. By plugging Lemma 2 into the drift-plus-cost inequality in Eq. (13), we obtain

$$\begin{aligned}&\Delta(q(t)) - V \cdot E[a_t - \omega e_t | q(t)] \\ &\leq B + q(t)E[(l_t^\Pi - L_{max})|q(t)] - E[a_t^\Pi - \omega e_t^\Pi | q(t)]V \\ &\leq B + \delta q(t) - V(\nu_t^* + \delta).\end{aligned} \tag{21}$$

By letting δ approach zero, summing the inequality over $t \in \{0, 1, \cdots, T-1\}$ and then dividing the result by T, we have

$$\begin{aligned}&\frac{1}{T}E[L(q(t)) - L(q(0))] - \frac{V}{T}\sum_{t=0}^{T-1} E[a_t^\Pi - \omega e_t^\Pi] \\ &\leq B - V \cdot \frac{1}{T}\sum_t \nu_t^* \\ &\leq B - V \cdot \nu_{opt},\end{aligned} \tag{22}$$

where the second inequality holds since for a minimum optimization, the sum of dynamic optimums is less than the sum of the global one for all slots.

Rearranging the terms and considering the fact that $L(q(t)) \geq 0$ and $L(q(0)) = 0$ yield the time-averaged utility bound:

$$\lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T} E[a_t - \omega e_t] \geq \nu_{opt} - \frac{B}{V}. \tag{23}$$

To obtain the service delay bound, by using Lemma 2, we have there are $\varepsilon > 0$, $\Phi(\varepsilon)$ and a policy Γ that satisfy

$$E[l_t^\Gamma - L_{max}] \leq -\varepsilon, \tag{24}$$

and

$$E[(a_t^\Gamma - \omega e_t^\Gamma) - \nu_t^*] = \Phi(\varepsilon). \tag{25}$$

Plugging the above into Eq. (13), we have

$$\Delta(q(t)) - V \cdot E[a_t - \omega e_t] \leq B - \varepsilon q(t) - V\Phi(\varepsilon). \tag{26}$$

By summing the above over $t \in \{0, 1, \cdots, T-1\}$ and rearranging terms, we have

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T-1} E[q(t)] \\ &\leq \frac{B - V(\Phi(\varepsilon) - \frac{1}{T}\sum_{t=0}^{T-1} E[a_t^\Gamma - \omega e_t^\Gamma])}{\varepsilon} \\ &\leq \frac{B}{\varepsilon} - \frac{V}{\varepsilon}(\nu_{opt} - \nu_{min}).\end{aligned} \tag{27}$$

Note that

$$\sum_{t=0}^{T-1} E[q(t)] \geq \sum_{t=0}^{T-1} E[l_t - L_{max}], \tag{28}$$

we have

$$\frac{1}{T}\sum_{t=0}^{T-1} E[l_t] \leq \frac{B}{\varepsilon} - \frac{V}{\varepsilon}(\nu_{opt} - \nu_{min}) + L_{max}. \quad (29)$$

Taking a lim sup of Eq. (29) as $t \to +\infty$ yields the service delay bound. The theorem holds immediately. □

Note that Eqs. (19) and (20) characterize the utility delay tradeoff within $[O(1/V), O(V)]$. Specifically, we can use an arbitrarily large value of V to drive the time-averaged utility arbitrarily close to the optimal ν_{opt} at a cost. As Eq. (20) implies, the time-averaged queue backlog grows linearly with V. Such a utility-delay tradeoff allows JCAB to make flexible configuration adaption. We will explain how the value of V impacts the overall performance in Section V.

C. A Practical Algorithm for Solving the One-Slot Problem for JCAB

In the last subsection, we give the performance guarantee of JCAB. However, to complete the JCAB algorithm, it remains to solve the optimization problem $\mathcal{P}_1$. Unfortunately, even this real-time optimization problem $\mathcal{P}_1$ is NP-hard in general [33], due to its combinatorial nature. In this subsection, we present a practical algorithm for solving the one-slot problem for JCAB. Note that, the performance analysis presented in the last subsection does not depend on the algorithm discussed in this subsection.

Let x_t denote $\{x_{k,t}^i | \forall m_i \in \mathcal{M}, \forall u_k \in \mathcal{U}\}$, which is the collection of model selection variables in time slot t. Similarly, we use $f_t = \{f_{k,t} | \forall u_k \in \mathcal{U}\}$ to represent frame rate selection for all video streams in time slot t, and $b_t = \{b_{k,t} | \forall u_k \in \mathcal{U}\}$ is the bandwidth allocation scheme in time slot t. Supposed that model selection x_t is fixed, there are two problems left to be solved.

The first problem is optimizing bandwidth allocation to reduce latency. Since the utility function is totally determined by the configuration, while bandwidth allocation only influences the service latency, the main objective of bandwidth allocation is to minimize the average latency l_t in each time slot,

$$\mathcal{P}_2: \min_{\{b_t\}} q(t) \cdot l_t. \quad (30)$$

The solution satisfies the Karush-Kuhn-Tucker (KKT) condition [34], so the optimal bandwidth allocation can be derived as follows:

$${b_{k,t}}^* = \frac{\sqrt{\sum_{i=0}^{N} \alpha(x_{k,t}^i r_i)^2}}{\sum_{k=1}^{K} \sqrt{\sum_{i=0}^{N} \alpha(x_{k,t}^i r_i)^2}}. \quad (31)$$

The second problem is adapting frame rates to maximize configuration utility. Given the bandwidth allocation b_t and model selection x_t, the frame resolutions are fixed and the average latency can be seen as a constant. The objective of frame rate adaption is to find the optimal tradeoff between detection accuracy and energy consumption, which is equal to the maximum of the following utility function:

$$\mathcal{P}_3: \max_{\{f_t\}} a_t - \omega e_t. \quad (32)$$

As we mentioned before, the relationship between detection accuracy and frame rate can be formulated as a concave function. Suppose that the accuracy function with respect to frame rate for u_k in slot t is $h_1 - h_2\, e^{-f_{k,t}/h_3}$, where h_1, h_2, and h_3 are known constant coefficients. Therefore, Problem $\mathcal{P}_3$ is a simple convex optimization problem [34] with only one kind of variables, i.e., $f_{k,t}$. It is not hard to find the optimal frame rates that make Eq. (32) reach its global maximum:

$${f_{k,t}}^* = \begin{cases} -h_3 \log(\dfrac{\omega \mu_k h_3}{g_2 \cdot \epsilon(\sum_{i=0}^{N} \alpha(x_{k,t}^i r_i)^2)}) & \text{if } x_{k,t}^0 = 1, \\ -h_3 \log(\dfrac{\omega \gamma_k h_3 \alpha \sum_{i=0}^{N} (x_{k,t}^i r_i)^2}{h_2 \cdot \epsilon(\sum_{i=0}^{N} \alpha(x_{k,t}^i r_i)^2)}) & \text{otherwise.} \end{cases} \quad (33)$$

Based on the analysis above, we can come to the conclusion that once the optimal CNN model selection x_t is found, $\mathcal{P}_2$ and $\mathcal{P}_3$ are both easy to solve. However, since model selection variables are binary, the whole problem is still a mix-integer nonlinear problem, hence, it is impossible to find an optimal solution in polynomial time. In this work, we propose to leverage Markov approximation [35] to obtain a near-optimal solution for model selection, as shown in Algorithm 2.

The one slot optimization algorithm for JCAB is described in Alg. 2. Without causing any confusion, we use $x_{k,t} = \{x_{k,t}^1, x_{k,t}^2, \ldots, x_{k,t}^N\}$ to denote the model selection vector for u_k. JCAB has multiple optimization objectives. On the one hand, it aims to find the optimal configuration to maximize the utility in Eq. (8); on the other hand, it should find the optimal bandwidth allocation since all cameras connected to the edge server share the same network channel and video streams vary in data size and bandwidth requirement. Firstly, we randomly select a user u_k to choose a new CNN model $\widehat{m}$, while the model selection for other users keeps unchanged, then the new model selection vector $\widehat{x}_t$ is obtained, under which the optimal $\widehat{b}_t$ and $\widehat{f}_t$ can be derived by solving $\mathcal{P}_2$ and $\mathcal{P}_3$, respectively. Afterwards, the new objective value $\widehat{g}$ is calculated, and g is known as the objective function value for the old solution $\{x_t, b_t, f_t\}$. In the current iteration, the model selected by u_k is updated to $\widehat{m}$ with probability η and keeps unchanged with probability $1 - \eta$ depending on the objective value difference $(\widehat{g} - g)$. Therefore, changing CNN model selection is more likely to occur if the new configuration $\{\widehat{x}_t, \widehat{b}_t, \widehat{f}_t\}$ results in a lower objective value. The above iterative loop will continue until T_{max} iterations have been reached or there is no significant improvement (i.e., $|\widehat{g} - g| < 0.01$) for more than 10 iterations.

The parameter $\tau \geq 0$ (Line 7), referred to as the smooth parameter, is used to control exploration versus exploitation (i.e. the degree of randomness). When τ is small, the algorithm tends to keep a new decision with a larger probability if it is better than the current decision. However, in this case, it takes more iterations to identify the global optimum since the algorithm may be stuck in a local optimum for a long time before exploring other alternatives that may lead to more efficient solutions. When τ becomes large, the algorithm tries to explore all possible solutions from time to time without convergence. The selection of τ will be discussed in Section V. As shown in [36], by proper parameter tuning, the Markov approximation-based Alg. 2 can converge in a super-linear rate.

D. Summary

To summarize this section, we list the major assumptions made in our work for using the Lyapunov framework:

- The bandwidth is assumed to be stable within each time slot.

Algorithm 2: One Slot Optimization for JCAB

Input: $\epsilon_k^t(r)$, $\phi_k^t(f)$, μ_k, γ_k, L_{max}, C, initial model selection vector x_t;
Output: model selection x_t, frame rate selection f_t, bandwidth allocation b_t;

repeat
  Randomly pick a user u_k and change its model selection vector $x_{k,t}$ into $\widehat{x}_{k,t}$ by selecting a new model $\widehat{m}$;
  if $\widehat{x}_{k,t}$ *is feasible* **then**
    $\widehat{x}_t \leftarrow \{x_{1,t}, x_{2,t}, \ldots, \widehat{x}_{k,t}, \ldots, x_{K,t}\}$;
    Obtain $\widehat{b}_t^*$ by solving $\mathcal{P}_2$ using Eq. (31);
    Obtain $\widehat{f}_t^*$ by solving $\mathcal{P}_3$ using Eq. (33);
    $\eta \leftarrow \frac{1}{1+e^{(\frac{\widehat{g}-g}{\tau})}}$;
    With probability η, user u_k accepts the new model $\widehat{m}$, $b_t \leftarrow \widehat{b}_t^*$, $f_t \leftarrow \widehat{f}_t^*$;
    With probability $(1-\eta)$, u_k keeps $x_{k,t}$ unchanged;
until T_{max} *iterations have been reached or there is no significant improvement (i.e.,* $|\widehat{g}-g| < 0.01$*) in the objective value for more than 10 iterations*;
return x_t, f_t, b_t;

- The Lyapunov framework actually supports multiple queues. However, there is only one virtual queue backlog in our work, whose arrivals in slot t are $l_t = \frac{1}{K}\sum_{k=1}^{K} l_{k,t}$ and the service is a fixed threshold L_{max}. Here, we assume that the arrivals are i.i.d over slots and their mean rate, $E[l_{k,t}]$, is constant.
- In order to measure the stability of virtual queue backlog, the latency overflow is used per slot (i.e., the latency overflow for slot t is $l_t - L_{max}$). Then, the Lyapunov framework requires such latency overflow to be bounded from two aspects: 1) the second moments of arrivals and service are bounded
$$\forall t, \quad E[\frac{1}{2}(l_t - L_{max})^2 | q(t)] \leq B,$$
where $B > 0$ is a finite constant as shown in Lemma 1; and 2) the expected latency overflow is negative
$$\forall t, \quad E[l_t^{\Gamma} - L_{max}] \leq -\varepsilon,$$
where $\varepsilon > 0$ exists for a policy Γ as shown in Inequality (24) in our paper and it is equivalent to Lemma 2. Here, the policy Γ is independent of current queue backlog, which implies a certain policy always exists for improving the queue backlog to be cleared. In sum, these two assumptions imply that the overflow for the queue backlog could be controlled and cleared to avoid the violation of time averaged constraints.
- The distance between the feasible sudation of instantaneous subproblem and its optimum is acceptable, $\forall t,\ E[a_t^{\Gamma} - \omega e_t^{\Gamma} - \nu_t^*] = \Phi(\varepsilon)$, which also implies the policy is independent of current queue backlog.

IV. The Multi-Edge Scenario

In the last section, we present the online algorithm JCAB for the single-edge scenario. In this section, we discuss how to extend JCAB to handle the multi-edge scenario.

A. Problem Formulation

We first present the problem formulation. Assuming that there are M edge servers, each of which contains K CNN models $m_1, m_2, \cdots, m_K$. We assume that the edge servers are close with each other, thus, the propagation delay can be ignored. The capacity of the j-th edge server is C_j. The uplink bandwidth of the j-th edge server in time slot t is B_t^j. We use $y_{k,t}^j$ to indicate whether user u_k chooses the j-th server to offload in time slot t; we use $x_{k,t}^{j,i}$ to indicate whether user u_k chooses the i-th CNN model on the j-th server to offload in time slot t. The other notations are similar to those in the single-edge case. The problem can be formulated as follows:

$$\begin{aligned}
\mathcal{P}_M: \quad & \max_{\{y,x,b,f\}} \quad \lim_{T\to+\infty} \frac{1}{T}\sum_{t=0}^{T}(a_t - \omega e_t) \\
& s.t.\ C_1: \sum_{j=1}^{M}\sum_{i=0}^{N} y_{k,t}^j x_{k,t}^{j,i} = 1, \quad \forall k,t \\
& C_2: x_{k,t}^{j,i} \in \{0,1\}, \quad \forall k,t,j,i \\
& C_3: f_{k,t} \leq \sum_{j=1}^{M}\sum_{i=0}^{N} y_{k,t}^j x_{k,t}^{j,i}\frac{1}{s_i}, \quad \forall k,t \\
& C_4: \sum_{k=1}^{K}\sum_{i=1}^{N} y_{k,t}^j x_{k,t}^{j,i} c_i \leq C_j, \quad \forall j,t \\
& C_5: \sum_{k=1}^{K} y_{k,t}^j b_{k,t} \leq B_t^j, \forall j,t \\
& C_6: \lim_{T\to+\infty} \quad \frac{1}{T}\sum_{t=0}^{T} l_t \leq L_{max} \\
& C_7: \sum_{j=1}^{M} y_{k,t}^j = 1, \quad \forall k,t \\
& C_8: y_{k,t}^j \in \{0,1\}, \quad \forall k,t,j \\
& C_9: x_{k,t}^{j,i} \leq y_{k,t}^j, \forall k,t,j.
\end{aligned}$$

The above problem has three new constraints. Constraints C_7 and C_8 ensure that, in each time slot, one and only one edge server can be chosen by a user. Constraint C_9 says that a user cannot use the CNN model on an edge server which is not chosen by it. This problem is harder than Problem $\mathcal{P}$, since it allows one more decision for each user about which edge server to choose.

B. Algorithm Design

The main idea is to decouple the edge server assignment from the other decisions. Initially, we assign users or video streams to edge servers based on the capacities and uplink bandwidths of edge servers; we then update or improve the edge server assignment at the beginning of each time slot based on the change of the uplink bandwidths and accuracy functions. The details are shown in Alg. 3.

Initially, we use first-fit to assign video streams to edge servers: the *size* of each video stream is seen as 1, while the *volume* of the j-th edge is seen as $\lceil \frac{(C_j+\sigma B_0^j)}{\sum_{h=1}^{M}(C_h+\sigma B_0^h)} K \rceil$. By doing so, we can ensure these M edge server can accommodate K video streams. Here, σ is a parameter controlling the trade-off between computing resources and bandwidth resources. Intuitively, an edge server with a large computing

Algorithm 3: mJCAB for the Multi-Edge Scenario

for $j = 1$ *to* M **do**
  $vol_0^j \leftarrow \lceil \frac{(C_j+\sigma B_0^j)}{\sum_{h=1}^{M}(C_h+\sigma B_0^h)} K \rceil$;
Assign K users to M edges using first-fit [37];
for $t = 0$ *to* $T-1$ **do**
  for $k = 1$ *to* K **do**
    Based on the first few frames at slot t, choose the most appropriate $\epsilon_k^t(r)$ and $\phi_k^t(f)$ from a set of pre-defined functions;
  for $j = 1$ *to* M **do**
    $K_j \leftarrow$the set of users assigned to j-th edge; **if** $\sum_{k\in K_j}(auc(\epsilon_k^t(r)) - auc(\epsilon_k^{t-1}(r))) < 0$, $\sum_{k\in K_j}(auc(\phi_k^t(f)) - auc(\phi_k^{t-1}(f))) < 0$, *or its uplink bandwidth decreases* **then**
      $u_k \leftarrow \arg_k \min_{k\in K_j}(\epsilon_k^t(r) + \phi_k^t(r))$;
      Move u_k to another edge whose uplink bandwidth increases or the total auc of all users on that edge increases;
  Choose $\{x_t, f_t, b_t\}$ by solving $\mathcal{P}_1$ for each edge server using Alg. 2;
  for $j = 1$ *to* M **do**
    $q_j(t+1) \leftarrow [q_j(t) + \frac{\sum_{k=1}^{K} y_{k,t}^j l_{k,t}}{\sum_{k=1}^{K} y_{k,t}^j} - L_{max}]^+$;

capacity and a large uplink bandwidth should accommodate more video streams.

At the beginning of each time slot, mJCAB tries to adapt the edge server assignment to the changing accuracy functions and bandwidths. The bandwidth change is easy to capture, however, how can we represent the change of an accuracy function? We introduce the concept of AUC (area under curve). The auc of a curve is the area bounded by the $y = 0$, the curve itself, $x = x_1$, and x_2. For example, in Fig. 4(a), the auc of the red dashed line is the area bounded by $y = 0$, the red dashed curve, $x = 360p$, and $x = 1080p$. When the auc of a curve decreases, it means we need a higher resolution or frame rate to achieve the accuracy as before. Therefore, in the mJCAB algorithm, when the uplink bandwidth of an edge server decreases, the total auc of all users on this edge decreases, we try to move one user that has the smallest auc to another edge whose uplink bandwidth increases or the total auc of all users on that edge increases.

We then choose $\{x_t, f_t, b_t\}$ by solving $\mathcal{P}_1$ for each edge server using Algorithm 2, after which we update the virtual queue length for each edge server. Note that, each edge server has a different virtual queue.

V. Performance Evaluation

In this section, we evaluate the performance of JCAB and mJCAB through simulations, and compare its performance against several baselines. We simulate each edge server with five CNN models, corresponding to each model, the input images are 360p, 540p, 720p, 900p and 1080p, respectively. The processing time per frame of these remote CNN models increases from 20ms to 250ms, according to their model sizes.

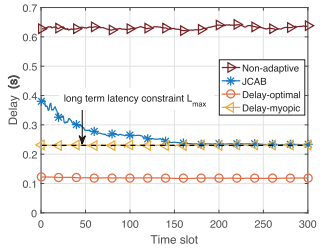


Fig. 5. Delay comparison of four algorithms.

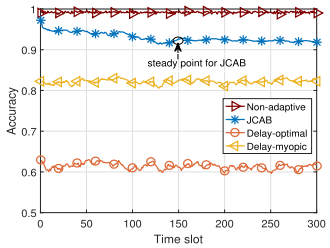


Fig. 6. Accuracy comparison of four algorithms.

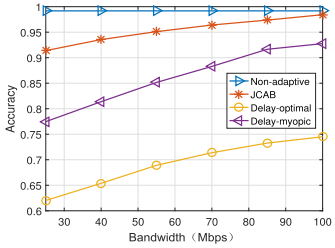


Fig. 7. Variation of accuracy with bandwidth.

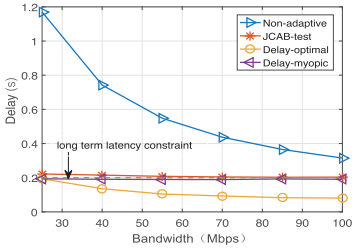


Fig. 8. Variation of latency with bandwidth.

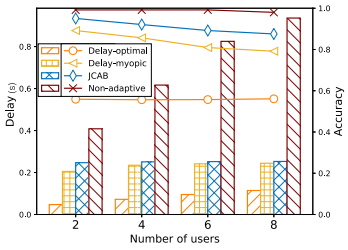


Fig. 9. The impact of user number (bars for delay while lines for accuracy).

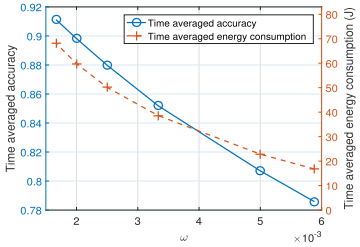


Fig. 10. Trade off between accuracy and energy.

Video frames processed by the local CNN model are scaled to 360p, but the processing speed can be much lower with the mean value of 200ms per frame. The network bandwidth varies

from 20Mbps to 100Mbps, according to the experimental measurements in [5]. Energy consumption of local processing (μ_k) is 5 J/frame and γ_k for all front-end devices are uniformly set to be 0.5×10^{-5} (J). Similar to Chameleon [14], frame rates and resolution of a video stream can be dynamically adjusted by FFmpeg [38].

We roughly divide targets into three categories according to their relative sizes in the video, and we profile several accuracy functions with respect to resolution. Similarly, different accuracy functions with respect to frame rate are drawn when targets move at different speed levels. At the beginning of each time slot, the most appropriate model will be selected according to the characteristics of targets. In the simulation, we compare our algorithms with three other algorithms:

- **Non-adaptive**: All video streams pick the most expensive configurations throughout the simulations to maximize the accuracy, while the energy and latency constraint are ignored.
- **Delay-optimal**: It aims to minimize the average service delay in each slot, regardless of the analytics accuracy and energy consumption.
- **Delay-myopic**: It imposes a hard latency constraint in each slot. It can satisfy the long-term delay constraint without requiring future information. However, it is less adaptive and purely myopic.

The reasons for choosing the above algorithms are as follows. Firstly, as we mentioned before, the joint configuration adaption and bandwidth allocation problem in this paper is significantly different from any existing studies, thus, none of existing algorithms can be used as comparison algorithm without any modifications.

Secondly, since we are mostly interested in the following two metrics: accuracy and delay, we want to compare JCAB and mJCAB with other baselines on these two metrics. **Non-adaptive** ignores the energy and delay constraints and it always pick the most expensive configurations, i.e., the highest resolution and the highest frame rate, therefore, **Non-adaptive** can achieve the highest accuracy in any situations and it can provide an upper bound on the accuracy. **Delay-optimal** minimizes the delay while ignoring the accuracy and energy constraints, thus, it provides a lower bound on the delay. **Delay-myopic** imposes a hard delay constraints in each slot, which means it may lose some possible opportunities to increase the accuracy with temporarily violating the delay constraint. **Delay-myopic** can be seen as a degeneration of JCAB when the Lyapunov optimization techniques are not used.

A. Comparison in the Single-Edge Scenario

We compare JCAB with Non-adaptive, Delay-optimal, and Delay-myopic in this subsection.

Figs. 5 and 6 show the average system delay and accuracy over time of four different algorithms. Note that JCAB has a convergence process, during which the algorithm gradually finds the optimal trade-off between latency and accuracy. Generally, JCAB achieves a desirable average accuracy while closely following the long-term energy constraint. The Non-adaptive scheme achieves the highest system accuracy as expected. However, it is achieved at the cost of an extremely long average latency per frame. Compared to Non-adaptive, JCAB slightly sacrifices the accuracy performance to meet the latency constraint. In contrast to Non-adaptive, the Delay-optimal method achieves the lowest latency in every slot, but the short latency comes with a big sacrifice in the average accuracy. For the Delay-myopic scheme, the long-term latency constraint $L_{max} = 0.23$ is also satisfied. However, because a hard latency constraint is imposed in every time slot, the algorithm is less flexible, resulting an inferior accuracy performance compared to JCAB.

1) Impact of Bandwidth: Figs. 7 and 8 show the impact of bandwidth on the converged time-averaged system delay and accuracy. Bandwidth traces are generated with the mean value increasing from 25Mbps to 100Mbps, according to the experimental measurements in [5]. As shown in Fig. 7, generally, all algorithms except Non-adaptive achieve a higher accuracy when bandwidth increases, since a higher bandwidth can support more expensive configurations. The average service latency of both JCAB and Delay-myopic are bounded. There is an insignificant latency performance gap between them when bandwidth is insufficient, but the gap decreases when bandwidth increases. However, the other two algorithms are more sensitive to bandwidth variation, under which the system latency decreases dramatically when the uplink bandwidth increases.

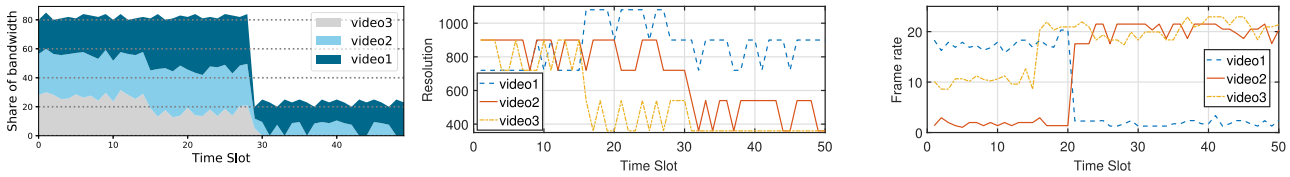
2) Impact of User Number: Fig. 9 shows the average perceived latency and analytics accuracy versus the number of users. For the Non-adaptive scheme, the latency increases significantly due to serious bandwidth contention. Accordingly, the converged time-averaged accuracy has a slight decrease when the user number exceeds 6, due to the limited computing resource of the edge server. For the Delay-optimal offloading, the achieved accuracy remains at a relative low level with a steady growth in system latency. The long-term average latency of JCAB and Delay-myopic closely follows the latency constraint under various user numbers. When serving more users, the latency constraint is achieved at a slight sacrifice in the analytics accuracy. For the Delay-myopic method, the time-averaged latency can be even slightly below the latency constraint when the user number is small.

B. A Running Example

Fig. 11 shows how the configurations adapt to bandwidth variation and video content dynamics. There are three cameras connected to the same edge server. The video content varies over time; in the 20th slot, targets in video stream 1 move slow, while targets in video stream 2 move fast, and thus the optimal frame rate for these two video streams changes accordingly. The intuition behind this adaption is that "more frames can be skipped if the difference between adjacent frames is small". In the 15th slot, targets in video stream 3 move near, we can degrade resolution for energy saving, while still maintaining the desired accuracy. As illustrated in Fig. 11(a), there are occasions when available bandwidth decreases dramatically, and all video streams subsequently lower the resolution to reduce the bandwidth requirement. Specifically, Camera 3 switches to local video processing, since it is less sensitive to resolution degradation relative to the other two video streams.

C. The Impact of Hyperparameters

We evaluate the impact of several hyperparameters in this subsection.



(a) Bandwidth (b) Resolution (c) Frame rate

Fig. 11. Runtime behavior of JCAB over time.

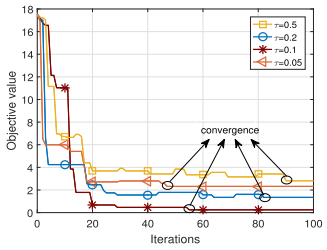


Fig. 12. Convergence of Algorithm 2.

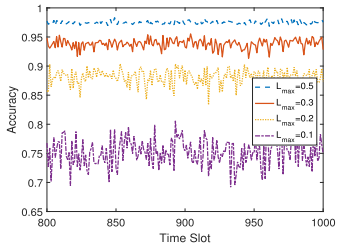


Fig. 13. The impact of latency constraint L_{max}.

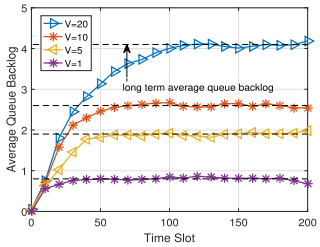


Fig. 14. The impact of weight V.

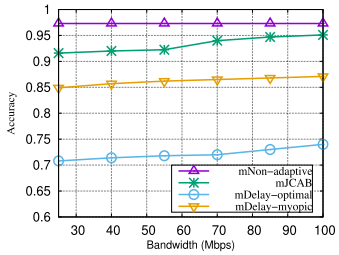


Fig. 15. Variation of accuracy in multi-edge.

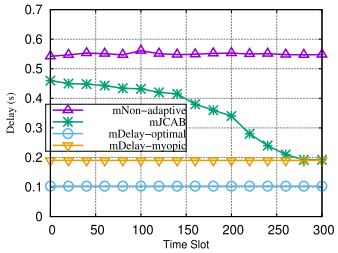


Fig. 16. Delay comparison in multi-edge.

1) Accuracy-Energy Tradeoff: Fig. 10 presents the converged time-averaged accuracy and energy consumption of JCAB under different values of the control parameter ω.

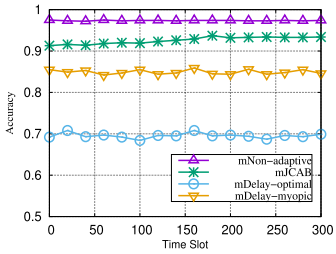


Fig. 17. Accuracy comparison in multi-edge.

We observe that when increasing ω from 0.001 to 0.003, the algorithm gains up to 44% energy consumption reduction with only 4% loss of the analytics accuracy. It implies that when a proper ω is set, our proposed algorithms will efficiently save energy consumption while maintaining a desirable accuracy.

2) Convergence: Fig. 12 shows the convergence process of Alg. 2. In general, a smaller τ leads to a faster convergence speed. When $\tau = 0.05$, the algorithm converges within 50 iterations. However, blindly decreasing τ impedes the identification of global optimum and results in the convergence to inferior solutions. In our experiment, the most appropriate value for τ is 0.1, which can achieve a good trade-off between the solution quality and the convergence rate.

3) Latency-Accuracy Tradeoff: Fig. 13 shows the accuracy of JACB under different values of L_{max} between the 800th and 1,000th time slots in an experiment. We observe that a higher accuracy can be achieved with a looser latency requirement. The accuracy fluctuates because of the variability of network bandwidth and video content. It is also obvious that the distribution of accuracy is more centralized to the median as L_{max} increases. When L_{max} is small, the latency constraint can be easily violated and sometimes accuracy should be greatly sacrificed to meet the latency constraint. On the contrary, a large L_{max} reduces the fluctuation range of the accuracy. As we mentioned before, the parameter V also controls the accuracy-latency tradeoff. Fig. 14 compares the average queue backlog with different values of the control parameter V. In Fig. 14, we know that all queue backlogs gradually converge to certain values, respectively. Thus, if there are more time slots, the long-term constraint can be satisfied. When the parameter V is large, it needs more time slots to converge. On the contrary, when the parameter V decreases, minimizing the system latency becomes the primary goal, which makes the service latency more stable.

D. Comparison in the Multi-Edge Scenario

We compare mJCAB with mNon-adaptive, mDelay-optimal, and mDelay-myopic in this subsection. mNon-adaptive denotes Non-adaptive with the same edge assignment with mJCAB; mDelay-optimal denotes Delay-optimal with greedy edge server assignment at the beginning of each time slot;

and mDelay-myopic denotes Delay-myopic with randomly generated edge assignment at the beginning of each time slot. The results are shown in Figs. 15, 16, and 17.

Fig. 15 shows the impact of bandwidth on the converged time-averaged detection accuracy. The bandwidth traces are generated using the same way in the single-edge scenario. When the available bandwidths increase, all algorithm achieve better accuracies. Figs. 16 and 17 show the comparison results in terms of delay. In general, the mNon-adaptive scheme achieves the highest system accuracy but has the worst system delay; the mDelay-optimal scheme achieves the best delay but has the lowest accuracy. mJCAB achieves a desirable average accuracy while closely following the long-term energy constraint. In other words, mJCAB slightly sacrifices the accuracy performance to meet the latency constraint.

VI. Related Work

We discuss the most closely related work in 3 categories.

A. Cloud-Assisted Video Stream Processing

Several studies [5], [18], [19] focused on cloud-assisted video stream processing. For example, VideoStorm [18] takes the resource-quality trade-off and the variety in quality and latency goals into account; GigaSight [19] continuously collects crowd-sourced videos from mobile devices. However, they all rely on remote clouds to ingest video streams, and assume that sufficient bandwidth is provisioned between cameras and the cloud. Different from them, we promote pushing computation to the network edge in proximity to data sources. In addition, we allow performing video analytics locally leveraging the computing power of smart cameras [39].

B. Video Analytics With Single Edge

Chameleon [14] periodically searches an exponentially large configuration space to find the optimal configuration for a video query, however, it only focuses on the trade-off between analytics accuracy and computation resource, while ignoring the fact that bandwidth is a scarce resource in video analytics. EAAR [16] utilizes dynamic RoI encoding and the decoupling of rendering and offloading to optimize the end-to-end latency, while we aim to find the optimal tradeoff between accuracy and energy consumption. DDS [40] uses server-side lightweight feedbacks to save bandwidth usage, however, it ignores client-side energy consumption and CNN model selection. Reducto [41] relies on camera-side filtering to mitigate the server-side load, however, Reducto did not consider camera-side energy consumption and the server latency constraint. O^3 [42] employs online learning to find the best trigger threshold for switching objection detection on edge and tracking on camera. The most related work is probably [43], in which the authors considered the complex interaction among model accuracy, video quality, battery constraints, network data usage, and network conditions to determine an optimal offloading strategy. However, there are no analytical models for resource demand and quality for a query configuration in [43], and they focused on client-side scheduling whereas we focus on server-side decisions for multiple video streams, with constraints on the network bandwidth and the capacity of edge servers.

C. Video Analytics With Multiple Edges

CrowdVision [20] parallelizes frame offload and local detection to optimize the processing time. FACT [3] enables fast and accurate object analytics. In these frameworks, images are extracted from the video with a fixed sampling rate, the analytics of different frames is treated as tasks with the same complexity and accuracy. The assumption, however, is improper since the frame rate and frame resolution will impact both the accuracy and query processing time for video analytics applications. JetStream [21] is the first to use configuration degradation to address bandwidth limits, but it requires developers to write manual policies which are generally sub-optimal. Our work aims to find the optimal tradeoff between accuracy and energy consumption in edge-assisted video analytics systems, with a long-term latency constraint, and thus none of these previous works can be directly and effectively applied to our problem.

VII. Discussion

In this section, we discuss several potential limitations and future research directions.

A. More General Accuracy Model

In our work, we pre-trained some accuracy models in edge servers, and the accuracy profiler selects models for each video stream according to the characteristics of targets. However, for two different video streams, the accuracy function with respect to resolution may not be exactly the same even if they have the same target size currently. Although our model is not perfect, we believe that it is a reasonable and valuable step towards studying content-aware adaptive video analytics in the edge environment.

B. Intra-Frame Encoding

To decrease the amount of video data per frame, we can further utilize the redundancy of video frames by encoding frames based on intra-frame difference. This is especially useful when the bandwidth is the bottleneck while the computing resources are adequate.

C. Deploying JCAB in Practice

JCAB and mJCAB can run on the user-side. This approach offers several advantages over deployment on the server-side. First, video streams do not need to send observations to edge servers, which avoids unnecessary information exchange and latency. Second, there is no need to modify edge servers; in other words, this adaptive configuration selection can be transparent to edge servers. Therefore, client-side JCAB can be seen as an overlay on the existing video stream-based applications; whenever there is failure in JCAB, we could disable the configuration selection service and fall back to the default one. This fault recovery mechanism could be invaluable.

VIII. Conclusion

In this paper, we study joint configuration adaption and bandwidth allocation for the edge-assisted real-time video analytics system. We proposed efficient online algorithms which can select appropriate configurations for multiple video streams according to the network condition and video contents, taking energy consumption, system latency, analytics accuracy into consideration. The proposed algorithm is easy to implement while providing provable performance guarantee.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and our editor Brooke Shrader.

REFERENCES

[1] K. Hsieh *et al.*, "Focus: Querying large video datasets with low latency and low cost," in *Proc. USENIX NSDI*, 2018, pp. 269–286.
[2] A. Henrysson and M. Ollila, "UMAR: Ubiquitous mobile augmented reality," in *Proc. 3rd Int. Conf. Mobile Ubiquitous Multimedia (MUM)*, 2004, pp. 41–45.
[3] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 1–9.
[4] *Avigilon*. Accessed: May 2020. [Online]. Available: http://www.avigilon.com/products/
[5] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "AWStream: Adaptive wide-area streaming analytics," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 236–252.
[6] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.
[7] Y. Liang, J. Ge, S. Zhang, J. Wu, Z. Tang, and B. Luo, "A utility-based optimization framework for edge service entity caching," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 11, pp. 2384–2395, Nov. 2019.
[8] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2019, pp. 2287–2295.
[9] S. Zhang, Y. Liang, J. Ge, M. Xiao, and J. Wu, "Provably efficient resource allocation for edge service entities using Hermes," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1684–1697, Aug. 2020.
[10] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jun. 2020.
[11] T. Ouyang, X. Chen, L. Zeng, and Z. Zhou, "Cost-aware edge resource probing for infrastructure-free edge computing: From optimal stopping to layered learning," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Dec. 2019, pp. 380–391.
[12] C. Wang *et al.*, "Joint server assignment and resource management for edge-based MAR system," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2378–2391, Oct. 2020.
[13] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4820–4828.
[14] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: Scalable adaptation of video analytics," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 253–266.
[15] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Jul. 2020, pp. 1–10.
[16] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *Proc. 25th Annu. Int. Conf. Mobile Comput. Netw.*, Aug. 2019, pp. 1–16.
[17] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," 2017, *arXiv:1701.01090*. [Online]. Available: http://arxiv.org/abs/1701.01090
[18] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *Proc. USENIX NSDI*, 2017, pp. 1–14.
[19] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, "Scalable crowd-sourcing of video from mobile devices," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2013, pp. 139–152.
[20] Z. Lu, K. S. Chan, and T. L. Porta, "A computing platform for video crowdprocessing using deep learning," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 1430–1438.
[21] A. Rabkin, M. Arye, S. Sen, V. S. Pai, and M. J. Freedman, "Aggregation and degradation in jetstream: Streaming analytics in the wide area," in *Proc. USENIX NSDI*, 2014, pp. 275–288.
[22] W. Zhang, S. Li, L. Liu, Z. Jia, Y. Zhang, and D. Raychaudhuri, "Hetero-edge: Orchestration of real-time vision applications on heterogeneous edge clouds," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2019, pp. 1270–1278.
[23] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
[24] (2016). *Cisco VNI: Global Mobile Data Traffic Forecast Update*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html
[25] M. Xiao, J. Wu, L. Huang, R. Cheng, and Y. Wang, "Online task assignment for crowdsensing in predictable mobile social networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 8, pp. 2306–2320, Aug. 2017.
[26] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-E, A. Bhartia, and D. Aguayo, "Large-scale measurements of wireless network behavior," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 153–165.
[27] A. Nikravesh, D. R. Choffnes, E. Katz-Bassett, Z. M. Mao, and M. Welsh, "Mobile network performance from user devices: A longitudinal, multidimensional analysis," in *Proc. Springer PAM*, 2014, pp. 12–22.
[28] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.
[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
[31] Z. Lu, S. Rallapalli, K. Chan, and T. L. Porta, "Modeling the resource requirements of convolutional neural networks on mobile devices," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 1663–1671.
[32] F. Liu, P. Shu, and J. C. S. Lui, "AppATP: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3051–3063, Nov. 2015.
[33] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2018, pp. 207–215.
[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[35] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[36] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6301–6327, Oct. 2013.
[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA, USA: MIT Press, 2009.
[38] *FFmpeg*. Accessed: May 2020. [Online]. Available: https://ffmpeg.org/
[39] *Amazon AWS DeepLens*. Accessed: May 2020. [Online]. Available: https://aws.amazon.com/deeplens/
[40] K. Du *et al.*, "Server-driven video streaming for deep learning inference," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 557–570.
[41] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proc. ACM SIGCOMM*, Jul. 2020, pp. 359–376.
[42] M. Hanyao, Y. Jin, Z. Qian, S. Zhang, and S. Lu, "Edge-assisted online on-device object detection for real-time video analytics," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2021, pp. 1–10, doi: 10.1109/INFOCOM42981.2021.9488741.
[43] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 1421–1429.

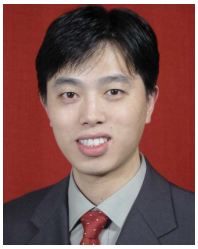
Sheng Zhang (Member, IEEE) received the B.S. and Ph.D. degrees from Nanjing University in 2008 and 2014, respectively. He is currently an Associate Professor with the Department of Computer Science and Technology, Nanjing University. He is also a member of the State Key Laboratory for Novel Software Technology. To date, he has published more than 80 articles, including those appeared in IEEE TRANSACTIONS ON MOBILE COMPUTING (TMC), IEEE/ACM TRANSACTIONS ON NETWORKING (TON), IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), IEEE TRANSACTIONS ON COMPUTERS (TC), MobiHoc, ICDCS, INFOCOM, SECON, IWQoS, and ICPP. His research interests include cloud computing and edge computing. He is a Senior Member of the CCF. He was a recipient of the 2015 ACM China Doctoral Dissertation Nomination Award. He received the Best Paper Award of IEEE ICCCN 2020 and the Best Paper Runner-Up Award of IEEE MASS 2012.

Can Wang received the B.S. degree from the School of Computer Science, Wuhan University. She is currently pursuing the M.S. degree with the Department of Computer Science and Technology, Nanjing University, under the supervision of Prof. Sheng Zhang. Her research interests include edge computing and wireless networks. Her publications include those appeared in IEEE INFOCOM, IEEE/ACM TRANSACTIONS ON NETWORKING (TON), and IEEE ISPA.

Yibo Jin (Graduate Student Member, IEEE) received the B.S. degree from the Department of Computer Science and Technology, Nanjing University, in 2017, where he is currently pursuing the Ph.D. degree, under the supervision of Prof. Sanglu Lu. He was a Visiting Student with The Hong Kong Polytechnic University, Hong Kong, in 2017. To date, his research has been published in journals, such as TPDS and in conferences, such as INFOCOM. His research interests include big data analytics and edge computing. He has received the Best Paper Candidate of IEEE WoWMoM 2021.

Jie Wu (Fellow, IEEE) is currently the Director of the Center for Networked Computing and Laura H. Carnell Professor with Temple University. He also serves as the Director for international affairs at the College of Science and Technology. He served as the Chair for the Department of Computer and Information Sciences from the Summer of 2009 to the Summer of 2016 and an Associate Vice Provost for international affairs from the Fall of 2015 to the Summer of 2017. Prior to joining Temple University, he was the Program Director at the National Science Foundation and a Distinguished Professor at Florida Atlantic University. He regularly publishes in scholarly journals, conference proceedings, and books. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He serves on several editorial boards, including IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON SERVICE COMPUTING, *Journal of Parallel and Distributed Computing*, and *Journal of Computer Science and Technology*. He was a recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award. He was the General Co-Chair of IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, and the Program Co-Chair of IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, an ACM Distinguished Speaker, and the Chair of the IEEE Technical Committee on Distributed Processing (TCDP). He is a CCF Distinguished Speaker.

Zhuzhong Qian (Member, IEEE) received the Ph.D. degree in computer science in 2007. He is currently a Professor with the Department of Computer Science and Technology, Nanjing University, China. His current research interests include cloud computing, distributed systems, and pervasive computing. He is the chief member of several national research projects on cloud computing and edge computing.



Mingjun Xiao (Member, IEEE) received the Ph.D. degree from USTC in 2004. He is currently a Professor with the School of Computer Science and Technology, University of Science and Technology of China (USTC). His research interests include mobile crowdsensing, blockchain, edge computing, mobile social networks, vehicular *ad hoc* networks, auction theory, data security, and privacy. He has published more over 90 papers in referred journals and conferences, including IEEE TRANSACTIONS ON MOBILE COMPUTING (TMC), IEEE/ACM TRANSACTIONS ON NETWORKING (TON), IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), IEEE TRANSACTIONS ON COMPUTERS (TC), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), IEEE TRANSACTIONS ON SERVICES COMPUTING (TSC), INFOCOM, ICDE, ICNP, and ICDCS. He served as the TPC member of many top conferences, including INFOCOM18-21, IJCAI 21, DASFAA20, and ICDCS19. He is on the reviewer board of several top journals, such as IEEE TRANSACTIONS ON MOBILE COMPUTING (TMC), IEEE/ACM TRANSACTIONS ON NETWORKING (TON), IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), IEEE TRANSACTIONS ON SERVICES COMPUTING (TSC), IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (TVT), and IEEE TRANSACTIONS ON CLOUD COMPUTING (TCC).

Sanglu Lu (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Nanjing University in 1992, 1995, and 1997, respectively. She is currently a Professor with the Department of Computer Science and Technology and the State Key Laboratory for Novel Software Technology. Her research interests include distributed computing, wireless networks, and pervasive computing. She has published over 80 papers in referred journals and conferences in the above areas.