Joint Configuration Adaptation and Bandwidth Allocation for Edge-based Real-time Video Analytics

Can Wang[†], Sheng Zhang*[†], Yu Chen[†], Zhuzhong Qian[†], Jie Wu[‡], and Mingjun Xiao[§]
[†]State Key Lab. for Novel Software Technology, Nanjing University, P.R. China
[‡]Center for Networked Computing, Temple University
[§]School of Computer Science and Technology / Suzhou Institute for Advanced Study,
University of Science and Technology of China, P.R. China

***Abstract*—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 the nearby server, thus mitigating long latency incurred by data transmission via wide area networks. When offloading frames from the front-end device to the edge server, the application configuration (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 adaption 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. Simulation results show that JCAB can effectively balance the analytics accuracy and energy consumption while keeping low system latency.**

I. Introduction

Major cities worldwide have millions of cameras deployed at traffic intersections, enterprise offices, and retail stores [1], for the purpose of surveillance, business intelligence, traffic control, crime prevention, etc. In many use cases, quick analysis on these live video streams is 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 in real time.

In general, video analytics applications demand 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 [4] and execute the deep learning algorithms there. 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]–[8], thereby enabling video analytics to occur closer to the data source.

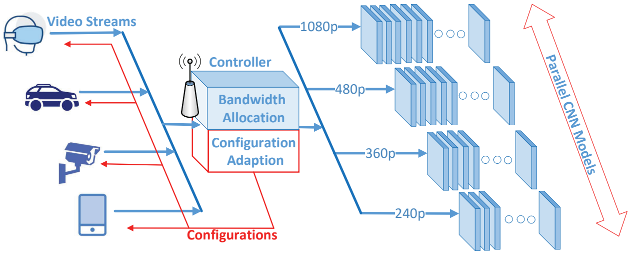


Fig. 1: An illustration of the edge-assisted video analytics system.

In video analytics, frames are extracted from the video at different sampling rates, compressed into various resolutions, and then processed by different CNN (convolutional neural network [9]) models. We refer to a particular combination of resolution and frame rate as a configuration [10]. Apparently, different configurations lead to different accuracies and energy consumptions. Since edge nodes serve as the backend for video processing [11], transmitting videos from their sources to the edge server via time-varying network links is inevitable. Thus, efficient offloading of video analytics involves *configuration adaption* and *bandwidth allocation*.

Researchers from both industry and academia have invested heavily in these two problems. Most of the previous works [3], [5], [10], [12]–[16] considered only one of them in offloading video analytics, which leads to sub-optimal performance. However, the scheduling of computing resources (via configuration adaption) and networking resources (via bandwidth allocation) are both of importance to the overall performance of edge-assisted video analytics.

In this work, we 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. Different CNN models are deployed on the edge server to match various resolutions. A small CNN with fewer convolutional layers [1] is cheaper, faster but less accurate. The objective 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 consumption, subject to a service latency budget. This problem faces many challenges for the following reasons:

The best offloading configuration varies over time. As

*The Corresponding Author is Sheng Zhang (sheng@nju.edu.cn). This work was supported in part by National Key R&D Program of China (2017YFB1001801), NSFC (61872175), Natural Science Foundation of Jiangsu Province (BK20181252), NSF (CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS 1651947, CNS 1564128), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.

we mention 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.

Network bandwidth is often unpredictable. As many have observed, the wide area network bandwidth has come to a standstill [17] while traffic demands are growing at a staggering rate [18]. Not only is WAN bandwidth scarce, it is also relatively expensive, and highly variable [5]. Similar scarcity and variations exist in wireless networks [19], [20], broadband access networks and cellular networks [21]. 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 reasons motivate us to propose *adaptive video analytics which is capable of optimizing the trade-off among the analytics accuracy, service latency, and energy consumption.* Our solution aims to find the most suitable video analytics offloading configuration and bandwidth allocation scheme for a multi-user edge-assisted video analytics system.

To the best of our knowledge, this is the *first* work to jointly optimize configuration adaption and bandwidth allocation for multi-video streams in the edge environment, explicitly taking into account the trade-off among the analytics accuracy, service latency, and energy consumption. 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 the edge server. The 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 adaption and bandwidth allocation problem, for optimizing the trade-off between accuracy and energy consumption, 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 a novel online algorithm, i.e., JCAB, which can efficiently adapt configurations and allocate bandwidth resources for video streams on the fly without foreseeing the future. JCAB utilizes 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 achieves 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 baselines.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III describes our system model. Section IV develops the JCAB algorithm. We evaluate our proposed design using extensive simulations in Section V. Finally Section VI concludes the paper.

II. Related work

There are many offloading frameworks focused on video stream processing [5], [12], [13]. VideoStorm [12] takes the resource-quality trade-off and the variety in quality and latency goals into account. GigaSight [13] 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 and mobile devices [22].

Various works have studied the computation offloading of video analytics in mobile edge computing, where videos are treated as sequences of images. In [14], the authors proposed to parallelize frame offload and local detection to optimize the processing time. The authors in [3] designed and implemented an edge network orchestrator which 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.

Several previous papers have considered optimizing video processing by adjusting the configuration. Chameleon [10] periodically searches an exponentially large configuration space to find the optimal configuration for a video query. 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. JetStream [15] 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, with a long-term latency constraint, and thus none of these previous works can be directly and effectively applied to our problem.

The most related work is probably [23], 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.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.

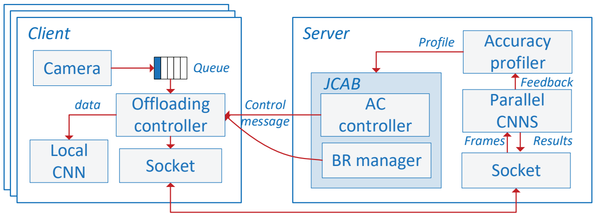


Fig. 2: The architecture of the edge-based live video analytics system.

However, there are no analytical models for resource demand and quality for a query configuration in [23], 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.

III. System Model

Suppose that a set of K users or video streams, denoted by $\mathcal{U} = \{u_1, u_2, ..., u_K\}$, 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 $\mathcal{M} = \{m_1, m_2, ..., m_N\}$ deployed on the edge server, with different input sizes of images. Let m_0 be the lightweight CNN model in each local device. Let r_i be the input resolution for the ith CNN.

We use s_i and c_i to denote the processing time and cost, respectively, per frame for m_i. It has been well studied in [24] that 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, CNN with lower input image resolution has a faster processing speed (i.e., smaller s_i) and needs less 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 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. On the client side, the proposed algorithm runs in the Adaptive Configuration (AC) controller and the Bandwidth Resource (BR) manager, which take accuracy profiles, network conditions and latency goals as the input, and send configurations along with bandwidth allocation back to the clients. The underlying characteristics (e.g., the velocity and sizes of objects) of video streams are extracted from analytics results and then utilized to update the accuracy profiles.

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

A. Analytics Accuracy Model

The accuracy models are derived based on the performance measurements obtained from our real experiments.

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 [25], 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 sufficient spatial overlap with the corresponding ground truth [10]. The spatial overlap can be measured by IOU (interaction of Union). In our experiment, an object is correctly detected when $IOU \geq 0.7$.

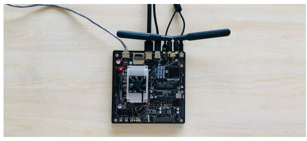
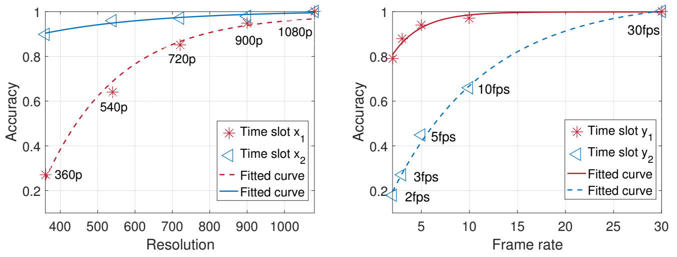


Fig. 3: We implement YOLO on NVIDIA Jetson TX2.

The results are plotted in Fig. 4(a). The red dashed line shows the accuracy in slot x_1, when the targets are small in the scene, while the blue line shows the accuracy in slot x_2, when pedestrians walk nearby. There are two observations. The first is that a higher image resolution produces a better analytics accuracy and the performance gain decreases at a high video resolution. Hence the relationship between accuracy and the resolution can be formulated as concave functions, i.e. the red line 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 the size of targets. Based on these observations, we use $\epsilon_k^t(r)$ to represent the accuracy function with respect to resolution for user u_k in slot t. We also introduce a binary variable $x_{k,t}^i$ to indicate whether 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 form 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_1, the cars in the scene are moving fast, while in slot y_2, 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. Denote by $f_{k,t}$ the frame sampling rate of u_k in time slot t. Many prior studies also show that the relationship between resolution/framerate and accuracy can often be formulated as

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.



(a) Resolution

(b) Frame Rate

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

concave functions [10].

It has been experimentally observed in [10] 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 $\epsilon_k^t(\sum_{i=0}^{N} x_{k,t}^i r_i)\phi_k^t(f_{k,t})$. The average accuracy over K users in time slot t is:

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

B. Energy Consumption Model

Battery life may become the primary concern 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 average transmission energy consumption of all users in slot t is:

$$e_t^{tran} = \frac{1}{K}\sum\nolimits_{k=1}^{K}\sum\nolimits_{i=1}^{N}\gamma_k\alpha(x_{k,t}^i r_i)^2 f_{k,t}. \tag{2}$$

We use μ_k to denote the energy cost of processing one frame on the local device of u_k [26]. Then the average data processing energy consumption for all users in time slot t is:

$$e_t^{proc} = \frac{1}{K}\sum\nolimits_{k=1}^{K} x_{k,t}^0\mu_k f_{k,t}. \tag{3}$$

If Eqs. (2) and (3) are combined, the average energy consumption of all users in time slot t is $e_t = e_t^{tran} + e_t^{proc}$.

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 bandwidth; we use $b_{k,t}$ to denote the bandwidth shared by u_k. Then, the latency per frame experienced by u_k in time slot t is:

$$l_{k,t} = \frac{\sum_{i=1}^{N}(\alpha x_{k,t}^i r_i)^2}{b_{k,t}} + \sum\nolimits_{i=0}^{N} x_{k,t}^i s_i. \tag{4}$$

Thus, the average latency for K video streams in slot t is:

$$l_t = \frac{1}{K}\sum\nolimits_{k=1}^{K} l_{k,t}. \tag{5}$$

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 energy cost. The natural objective is the maximum of time-averaged utility for all video streams, which can be formulated as:

$$\begin{aligned}
\mathcal{P}: \max_{\{x,b,f\}} \quad & \lim_{T\to+\infty}\frac{1}{T}\sum\nolimits_{t=0}^{T}(a_t - \omega e_t).\\
s.t. \quad & C_1: \sum\nolimits_{i=0}^{N} x_{k,t}^i = 1, u_k \in \mathcal{U}, t \in \mathcal{T} = \{1,...,T\}.\\
& C_2: x_{k,t}^i = \{0,1\}, u_k \in \mathcal{U}, t \in \mathcal{T}.\\
& C_3: f_{k,t} < \sum\nolimits_{i=0}^{N} x_{k,t}^i\frac{1}{s_i}, u_k \in \mathcal{U}, t \in \mathcal{T}.\\
& C_4: \sum\nolimits_{k=1}^{K}\sum\nolimits_{i=1}^{N} x_{k,t}^i c_i < C, u_k \in \mathcal{U}, t \in \mathcal{T}.\\
& C_5: \sum\nolimits_{k=1}^{K} b_{k,t} = B_t, u_k \in \mathcal{U}, t \in \mathcal{T}.\\
& C_6: \lim_{T\to+\infty}\frac{1}{T}\sum\nolimits_{t=0}^{T} l_t < L_{max}.
\end{aligned} \tag{6}$$

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 u_k. Constraint C_3 says that the selected frame rate cannot exceed the processing frequency of the CNN (remote or local), otherwise video frames would accumulate and lead to queue delay. The fourth constraint C_4 is due to the capacity of edge server, denoted as C. Let B_t represent the uplink bandwidth over the entire time 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.

IV. Online Algorithm

To decouple the long-term latency constraint, we transform the original time-averaged problem into a series of real-time minimization problems leveraging the Lyapunov framework [27], and then we develop a lightweight online algorithm which only relies on the current bandwidth information and

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.

video content to derive the adaptation strategy, without global information over the long run.

A. Problem Transformation Using Lyapunov Optimization

A major challenge of directly solving $\mathcal{P}$ is that the long-term latency constraint couples the CNN model selection, frame rate adaption and bandwidth allocation across different time slots. To address this challenge, we define a virtual queue as a historical measurement of the exceeded latency and assume that the initial queue backlog is 0 (i.e., $q(0) = 0$). The queue length evolves as follows:

$$q(t+1) = [q(t) + \sum\nolimits_{k=1}^{K} l_{k,t} - L_{max}]^{+}, \tag{7}$$

where $[x]^{+}$ denotes $max\{x, 0\}$ for any x. If we aggressively pursue high accuracy by adopting the most expensive configuration, 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. Actually, it can be proved that the stability of the virtual queue can ensure that the time-averaged latency does not exceed the threshold L_{max}. We first define a quadratic Lyapunov function: $L(q(t)) = \frac{1}{2}(q(t))^2$, 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:

$$\Delta(q(t)) = E[L(q(t+1)) - L(q(t))|q(t)]. \tag{8}$$

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:

$$\Delta(q(t)) - V \cdot E[a_t - \omega e_t|q(t)]. \tag{9}$$

The positive parameter V is used to adjust the tradeoff between latency minimization and utility maximization. Rather than directly minimizing the drift-plus-penalty term in each slot, the min-drift-plus-penalty algorithm [28] in Lyapunov optimization seeks to minimize an upper bound of it. We derived an upper bound on the drift-plus-penalty term in our specific problem and it is stated in the following lemma:

***Lemma** 1:* For all possible values of $q(t)$ by using any offloading configuration over all time slots, the following statement holds:

$$\begin{aligned} &\Delta(q(t)) - V \cdot E[a_t - \omega e_t|q(t)] \leq \\ &B + q(t)E[(l_t - L_{max})|q(t)] - V \cdot E[a_t - \omega e_t|q(t)], \end{aligned} \tag{10}$$

where $B = \frac{1}{2}(l_{max} - L_{max})^2$ is a constant value for all time slots, and $l_{max} = \max_{t \in T}\{l_t\}$ represents the largest average delay in all slots. The detailed proof is given in the Appendix.

Algorithm 1: The JCAB Algorithm

```
Input: $q(0) \leftarrow 0$, $\mu_k$, $\gamma_k$, $L_{max}$, $C$;
for $t = 0$ to $T$ do
    Profile accuracy function $\epsilon_k^t(r)$, $\forall k$;
    Profile accuracy function $\phi_k^t(f)$, $\forall k$;
    Choose $\{x_t, f_t, b_t\}$ by solving $\mathcal{P}_1$ using Algorithm 2;
    $q(t+1) = [q(t) + l_t - L_{max}]^{+}$;
```

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:

$$\begin{aligned} \mathcal{P}_1 : &\min_{\{x,b,f\}} \quad q(t) \cdot l_t - V \cdot (a_t - \omega e_t). \\ s.t. \quad &C_1, C_2, C_3, C_4, C_5. \end{aligned} \tag{11}$$

Notice that solving $\mathcal{P}_1$ requires only currently available information as 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 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. Now, to complete the algorithm, it remains to solve the optimization problem $\mathcal{P}_1$, which will be discussed in the next subsection.

B. Lyapunov-based Online Algorithm

Unfortunately, this real-time optimization problem $\mathcal{P}_1$ is NP-hard in general [29], due to its combinatorial nature. We develop the JCAB algorithm to solve the problem based on the Markov approximation method [30].

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. Similarly, we use $f_t = \{f_{k,t} | \forall u_k \in \mathcal{U}\}$ to represent frame rate selection for all video streams, 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 the minimization of average latency l_t,

$$\mathcal{P}_2 : \min_{\{b_t\}} \quad q(t) \cdot l_t. \tag{12}$$

The solution satisfies the Karush-Kuhn-Tucker (KKT) condition [31], so the optimal bandwidth allocation can be derived as follows:

$$b_{k,t}{}^{*} = \frac{\sqrt{\sum_{i=0}^{N} \alpha x_{k,t}^i r_i}}{\sum_{k=1}^{K} \sqrt{\sum_{i=0}^{N} \alpha x_{k,t}^i r_i}}. \tag{13}$$

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.

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 constant. The objective of frame rate adaption is to find the optimal tradeoff between accuracy and energy, which is equal to the maximum of utility function:

$$\mathcal{P}_3 : \max_{\{f_t\}} \quad a_t - \omega e_t. \tag{14}$$

As we mentioned before, the relationship between accuracy and frame rate can be formulated as concave functions. Suppose that the accuracy function with respect to frame rate for u_k in slot t is $c_1 - c_2 e^{-f_{k,t}/c_3}$, where c_1, c_2, and c_3 are known constant coefficients. It can be proved that Eq. (14) has global maximum, and the optimal frame rate is:

$$f_{k,t}{}^{*} = \begin{cases} -c_3 \log(\frac{\omega \mu_k c_3}{c_2 \cdot \epsilon(\sum_{i=0}^{N} \alpha x_{k,t}^{i} r_i)}) & \text{if } x_{k,t}^{0} = 1, \\ -c_3 \log(\frac{\omega \gamma_k c_3 \alpha \sum_{i=0}^{N} (x_{k,t}^{i} r_i)^2}{c_2 \cdot \epsilon(\sum_{i=0}^{N} \alpha x_{k,t}^{i} r_i)}) & \text{otherwise.} \end{cases} \tag{15}$$

Based on the analysis above, we can come to the conclusion that once optimal 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 a mix-integer nonlinear problem, hence, it is impossible to find an optimal solution in polynomial time. In this paper, we propose to leverage Markov approximation to obtain a near-optimal solution for model selection, as shown in Algorithm 2.

In our online control algorithm JCAB, we divide time into T discrete time slots. At the beginning of each time slot, the accuracy models are updated by the accuracy profiler according to the current video content. Given the accuracy functions, we can find the optimal configurations and bandwidth allocation scheme for the current slot by solving problem $\mathcal{P}_1$. Finally, the average latency is utilized to update the latency queue.

The one slot optimization algorithm for JCAB is described in Alg. 2, in which $x_{k,t} = \{x_{k,t}^1, x_{k,t}^2, ..., x_{k,t}^N\}$ denotes 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. (6); 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 date 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 $\widehat{f}_t$ and $\widehat{b}_t$ can be derived by solving $\mathcal{P}_2$ and $\mathcal{P}_3$. 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 processes 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.

```
Algorithm 2: One Slot Optimization for JCAB
   Input: $\epsilon_k^t(r)$, $\phi_k^t(f)$, $\mu_k$, $\gamma_k$, $L_{max}$, $C$, initial model
          selection vector $x_t$;
   Output: configuration ($x_t$, $f_t$), bandwidth allocation $b_t$;
1  repeat
2      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 $\widehat{m}$;
3      if $\widehat{x}_{k,t}$ is feasible then
4          $\widehat{x}_t \leftarrow \{x_{1,t}, x_{2,t}, ..., \widehat{x}_{k,t}, ..., x_{K,t}\}$;
5          Obtain $\widehat{b}_t^*$ by solving problem $\mathcal{P}_2$ using Eq. (13);
6          Obtain $\widehat{f}_t^*$ by solving problem $\mathcal{P}_3$ using Eq. (15);
7          $\eta \leftarrow \frac{1}{1+e^{(\frac{\widehat{g}-g}{\tau})}}$;
8          With probability $\eta$, user $u_k$ accepts the new
             model $\widehat{m}$, $b_t \leftarrow \widehat{b}_t^*$, $f_t \leftarrow \widehat{f}_t^*$;
9          With probability $(1 - \eta)$, $u_k$ keeps $x_{k,t}$
             unchanged;
10 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;
11 return $x_t, f_t, b_t$;
```

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 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 lead to more efficient solutions. When $\tau \rightarrow +\infty$, 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 [32], by proper parameter tuning, the Markov approximation-based Alg. 2 can converge in a super-linear rate.

C. Theoretic Analysis

Theorem 1: JCAB achieves the following performance bounds on the time-averaged utility and queue backlog:

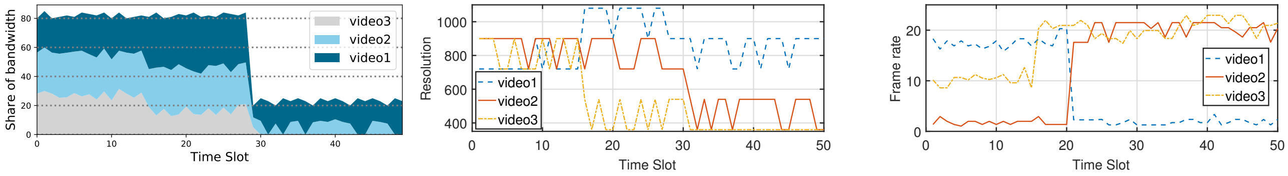
$$\lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=0}^{T} E[a_t - \omega e_t] >= \nu_{opt} - B/V, \tag{16}$$

$$\lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=0}^{T} E[l_t] \leq \frac{B}{\varepsilon} + \frac{V}{\varepsilon}(\nu_{opt} - \nu_{min}) + L_{max}, \tag{17}$$

where ν_{min} is the objective value of the worst solution for $\mathcal{P}$, ν_{opt} is the optimal utility 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. Please refer to the Appendix for the proof.

Note that Eqs. (16) and (17) characterize the utility delay tradeoff within [O(1/V),O(V)]. Specifically, we can use an

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.



(a) Bandwidth

(b) Resolution

(c) Frame rate

Fig. 5: Runtime behavior of JCAB over time.

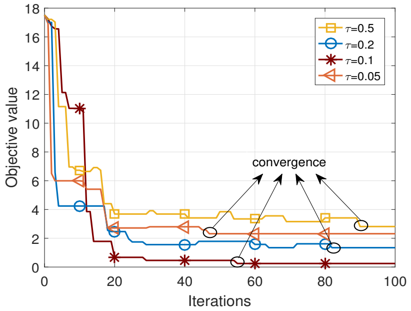


Fig. 6: Convergence of Algorithm 2.

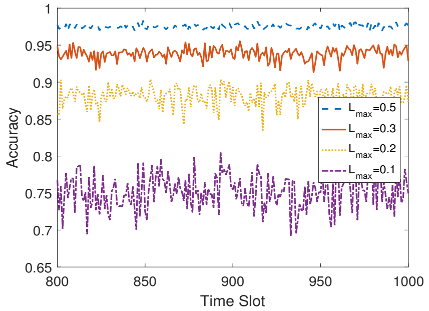


Fig. 7: The impact of latency constraint L_{max}.

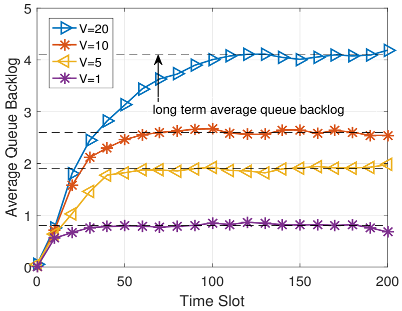


Fig. 8: The impact of weight V.

arbitrarily large value of V to drive the time-averaged utility arbitrarily close to the optimal v_{opt} at a cost. As Eq. (17) 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 performance in Section V.

V. Simulation

In this section, we evaluate the performance of JCAB through simulations, and compare its performance against several baselines. We simulate an 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. 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) out of convenience.

We roughly divide targets into three categories according to their relative sizes in the image, and we profile several accuracy functions with respect to frame resolution under these three different conditions. 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. Note that this heuristic method is not always accurate but it is flexible and quick, while the profiling cost is relatively low. In the simulation, we compare our algorithm with three other benchmarks:

- **Non-adaptive**: All video streams pick the most expensive configurations all the time 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.

A. A Running Example

Fig. 5 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. 5(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.

B. The Impact of Different Parameters

1) Convergence: Fig. 6 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

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.

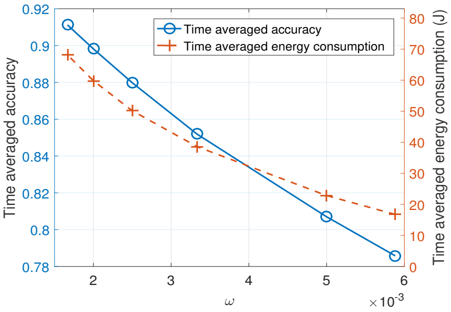


Fig. 9: Trade off between accuracy and energy.

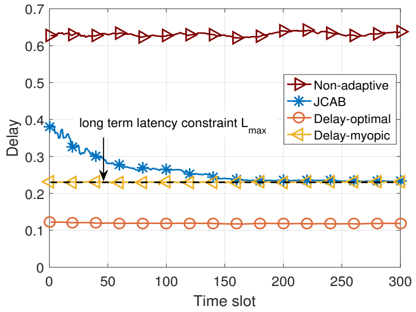


Fig. 10: Delay comparison of four algorithms.

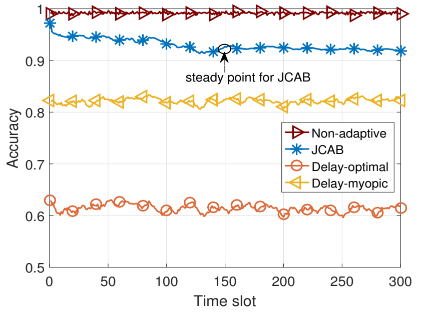


Fig. 11: Accuracy comparison of four algorithms.

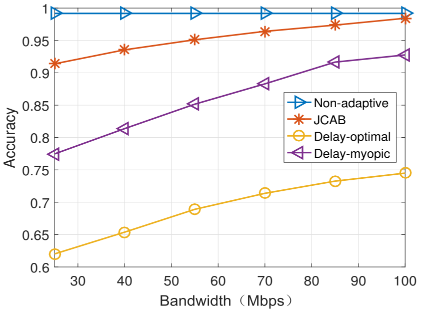


Fig. 12: Variation of accuracy with bandwidth.

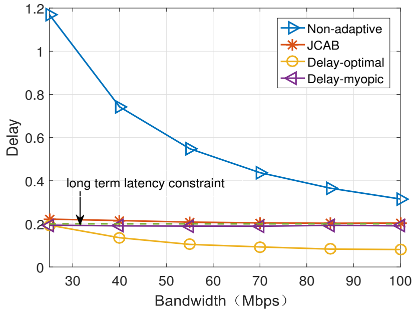


Fig. 13: Variation of latency with bandwidth.

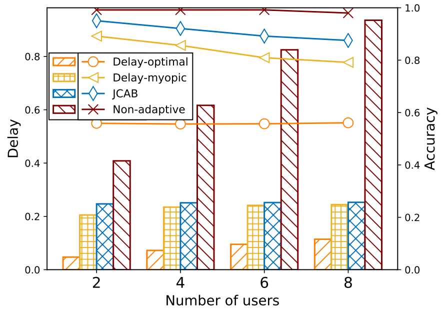


Fig. 14: The impact of user number.

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.

2) Latency-accuracy tradeoff: Fig. 7 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. 8 compares the average queue backlog with different values of the control parameter V. In Fig. 8, we know that all queue backlogs gradually converge to certain values. Thus, if there are more time slots, the long-term constraint can be satisfied. When the control parameter V is large, it needs more time slots to converge; On the contrary, when V decreases, latency becomes the primary goal, which makes the service latency more stable.

3) Accuracy-energy tradeoff: Fig. 9 presents the converged time-averaged accuracy and energy consumption of JCAB under different values of the control parameter ω. 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.

C. Algorithm Comparison

Figs. 10 and 11 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 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 a cost of long average latency per frame. Compared to Non-adaptive, JCAB slightly sacrifices the accuracy performance to meet the latency constraint. 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 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.

D. The Impact of Bandwidth

Figs. 12 and 13 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. 12, 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

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.

more sensitive to bandwidth variation, under which the system latency decreases dramatically when bandwidth increases.

E. The Impact of User Number

Fig. 14 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 resource limitation in 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.

VI. Conclusion and Future work

In this paper, we study joint configuration adaption and bandwidth allocation for the edge-assisted real-time video analytics system. We proposed an efficient online algorithm 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.

Here are a few limitations in the current model that demand future research effort. First, to decrease the amount of data per frame, we can utilize the redundancy of video frames by encoding frames based on intra-frame difference. Second, the accuracy model may not generalize. We pre-trained some accuracy models in the edge server, 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.

Appendix

Proof of Lemma 1: From Eq. (8), we have:

$$\begin{aligned}\Delta(q(t)) &= E[L(q(t+1)) - L(q(t))|q(t)] \\ &= \frac{1}{2}E[q^2(t+1) - q^2(t)|q(t)] \\ &\leq \frac{1}{2}E[(q(t) + l_t - L_{max})^2 - q(t)^2|q(t)] \\ &= \frac{1}{2}(l_t - L_{max})^2 + q(t)E[(l_t - L_max)|q(t)] \\ &\leq B + q(t)E[(l_t - L_{max})|q(t)].\end{aligned} \tag{18}$$

where $B = \frac{1}{2}(l_{max} - L_{max})^2$ is a constant value for all time slots, and $l_{max} = \max_{t \in T}\{l_t\}$ represents the largest average delay in all slots. We now incorporate the expected utility over one time slot to both sides of Eq. (18), then we have:

$$\begin{aligned}&\Delta(q(t)) - V \cdot E[a_t - \omega e_t|q(t)] \leq B \\ &+ q(t)E[(l_t - L_{max})|q(t)] - V \cdot E[a_t - \omega e_t|q(t)].\end{aligned} \tag{19}$$

Proof of Theorem 1: To prove the performance guarantee, 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:

$$E[l_t^{\Pi} - L_{max}] \leq \delta, \text{ and } E[a_t^{\Pi} - \omega e_t^{\Pi}] \leq v_{opt} + \delta. \tag{20}$$

Proof: The proof can be obtained by Theorem 4.5 in [27], which is omitted for brevity. ■

Recall that the 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 (Eq. (10)), we obtain

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

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

$$\begin{aligned}&\frac{1}{T}E[L(q(t)) - L(q(0))] - \frac{V}{T}\sum\nolimits_{t=0}^{T-1} E[a_t^{\Pi} - \omega e_t^{\Pi}] \\ &\leq B - V \cdot v_{opt}.\end{aligned} \tag{22}$$

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

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

To obtain the energy consumption bound, we assume there are $\varepsilon \leq 0$, $\Phi(\varepsilon)$ and a policy a_t^{Γ}, b_t^{Γ} that satisfy:

$$E[l_t^{\Gamma} - L_{max}] \leq -\varepsilon, \text{ and } E[a_t^{\Gamma} - \omega e_t^{\Gamma}] = \Phi(\varepsilon). \tag{24}$$

Plugging the above into Eq. (10), we have

$$\Delta(q(t)) - V \cdot E[a_t - \omega e_t] \leq B - \varepsilon q(t) - V\Phi(\varepsilon). \tag{25}$$

Summing the above over $t \in \{0, 1, ..., T-1\}$ and rearranging terms give:

$$\begin{aligned}&\frac{1}{T}\sum\nolimits_{t=0}^{T-1} E[q(t)] \leq \frac{B - V(\Phi(\varepsilon) - \frac{1}{T}\sum_{t=0}^{T-1} E[a_t^{\Gamma} - \omega e_t^{\Gamma}])}{\varepsilon} \\ &\leq \frac{B}{\varepsilon} - \frac{V}{\varepsilon}(v_{opt} - v_{min}).\end{aligned} \tag{26}$$

Considering $\sum_{t=0}^{T-1} E[q(t)] \geq \sum_{t=0}^{T-1} E[l_t - L_{max}]$, we have:

$$\frac{1}{T}\sum\nolimits_{t=0}^{T-1} E[l_t] \leq \frac{B}{\varepsilon} - \frac{V}{\varepsilon}(v_{opt} - v_{min}) + L_{max}. \tag{27}$$

Taking a lim sup of Eq. (27) as $t \to +\infty$ yields the energy consumption bound.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu, "Focus: Querying large video datasets with low latency and low cost," in *Proc. of USENIX NSDI*, 2018, pp. 269–286.

[2] A. Henrysson and M. Ollila, "Umar: Ubiquitous mobile augmented reality," in *Proc. of ACM Mobiquitous*, 2004, pp. 41–45.

[3] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. of IEEE INFOCOM*, 2018, pp. 1–9.

[4] "Avigilon," http://avigilon.com/products/.

[5] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *Proc. of ACM SIGCOMM*, 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 Transactions on Parallel and Distributed Systems*, pp. 1–12, 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. of IEEE INFOCOM*, 2019, pp. 2287–2295.

[9] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE CVPR*, 2016, pp. 4820–4828.

[10] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proc. of ACM SIGCOMM*, 2018, pp. 253–266.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *arXiv preprint arXiv:1701.01090*, 2017.

[12] 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. of USENIX NSDI*, 2017, pp. 1–14.

[13] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan, "Scalable crowd-sourcing of video from mobile devices," in *Proc. of ACM MobiSys*, 2013, pp. 139–152.

[14] Z. Lu, K. S. Chan, and T. La Porta, "A computing platform for video crowdprocessing using deep learning," in *Proc. of IEEE INFOCOM*, 2018, pp. 1430–1438.

[15] 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. of USENIX NSDI*, 2014, pp. 275–288.

[16] 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. of IEEE INFOCOM*, 2019, pp. 1270–1278.

[17] 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, 2016.

[18] "Cisco VNI: Global mobile data traffic forecast update," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, 2016.

[19] M. Xiao, J. Wu, L. Huang, R. Cheng, and Y. Wang, "Online task assignment for crowdsensing in predictable mobile social networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 8, pp. 2306–2320, Aug 2017.

[20] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-e, A. Bhartia, and D. Aguayo, "Large-scale measurements of wireless network behavior," in *Proc. of ACM SIGCOMM*, 2015, pp. 153–165.

[21] 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. of Springer PAM*, 2014, pp. 12–22.

[22] "Amazon AWS DeepLens," https://aws.amazon.com/deeplens/.

[23] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proc. of IEEE INFOCOM*, 2018, pp. 1421–1429.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of ICLR*, 2015.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. of IEEE CVPR*, 2016, pp. 779–788.

[26] Z. Lu, S. Rallapalli, K. Chan, and T. La Porta, "Modeling the resource requirements of convolutional neural networks on mobile devices," in *Proc. of ACM MM*, 2017, pp. 1663–1671.

[27] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[28] F. Liu, P. Shu, and J. C. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[29] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. of IEEE INFOCOM*, 2018, pp. 207–215.

[30] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[31] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[32] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on January 06,2021 at 05:37:56 UTC from IEEE Xplore. Restrictions apply.