A Low-Cost Video Analytics System with Velocity Based Configuration Adaptation in Edge Computing

Woo-Joong Kim and Chan-Hyun Youn

1 Introduction

As video cameras are pervasive as a CCTV or a mobile device, many researches attempt to utilize the potential of these cameras and develop various intelligent systems and services with Video Analytics (VA), which is a generic term for various methods of video analysis [3, 11]. Deep Neural Network (DNN) recently has been a key technology for video analytics due to their high accuracy and various applications such as object detection and tacking, action recognition and instance segmentation, etc. [2]. A desired video analytics quality requires high accuracy with low latency, but the high accuracy of DNN requires a prohibitive cost on latency [1, 4]. DNN based VA involves high complexity computation which cannot be performed at high speed on the limited computing resource of end devices, such as CCTV or mobile device [5]. For example, on Qualcomm Adreno 530 GPU which is the embedded GPU of end devices, the processing time per image of big-yolo, a DNN based object detection model with 22 convolution layers, is about 600∼4500 ms (1.6∼0.2) fps and those of tiny-yolo, a DNN based object detection model with 9 convolution layer, is about 200∼1100 ms (5∼0.9) fps [9].

In this regard, Mobile Edge Computing (MEC) has been emerged as a promising solution to address these issues. The MEC provides computing resources located in close proximity to end users within the Radio Access Network. The latency on DNN based VA in the limited computing resource of end devices would be significantly improved by offloading the computation to the MEC. However, since multiple video

W.-J. Kim · C.-H. Youn (✉)
School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea
e-mail: w.j.kim@kaist.ac.kr; chyoun@kaist.ac.kr

H. R. Arabnia et al. (eds.), *Advances in Artificial Intelligence and Applied Cognitive Computing*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-70296-0_48

cameras are usually served concurrently for VA in the MEC, even the computing resources (i.e. GPUs) of the MEC are limited to handle each video stream.

Many VA systems efficiently utilizing the limited computing resources of the MEC shared by multiple video cameras have been proposed. Their main target application is object detection, which is the basis of various high-level VA applications such as scene graph detection and instance segmentation. In order to guarantee the VA performance in real time, several systems address these issues by adapting configurations [3, 11], such as frame resolution, frame sampling rate and detector model. However, they have a high profiling cost problem derived from an online-profiling technique, which is essential due to the dynamics of the configuration's impact on video analytics accuracy [3].

In this paper, we propose a low-cost VA system analyzing multiple video streams efficiently under limited resource. The objective of our proposed system is to find the best configuration decision of frame sampling rate for multiple video streams in order to minimize the accuracy degradation while guaranteeing the real time VA in the shared limited resource. Firstly, the frame sampling rate knob of each video stream is adapted optimally based on the velocity feature of objects extracted from its video context in low-cost. Its objective on each video stream is to find the best frame sampling rate for reducing the resource demands as much as possible while guaranteeing a desired accuracy. Secondly, the frame sampling rate knob of each video stream is adapted additionally in a greedy way, considering a limited resource shared by multiple video streams. Its objective is to reduce the resource demands of each video stream in fairness to analyze multiple video streams in real time in the shared limited resource while minimizing the accuracy degradation of each video stream.

2 Low-Cost Video Analytics System

We introduce our VA system which supports multiple camera feeds fairly under limited resource capacity. Figure 1 shows the structure of our VA system for edge computing environment. It consists of a pre-processor, an object detection model, and a configuration controller [3]. It is implemented on a GPU-enabled edge server attached to a small cell base station and its input video streams are generated from multiple cameras. Let $[n] = 1, \ldots, n$ denote the set of integer numbers which has a cardinality of n.

There are I multiple video streams, each of which has its default frame rate and resolution. Each video stream is split into smaller segments, each of which is a contiguous set of frames spanning a T-second interval (By default, $T = 4$). Let $S_i = \{S_{i,1}, S_{i,2}, \ldots, S_{i,j}, \ldots, S_{i,J}\}$ denote the segments of i-th video stream whose size is J. The raw frames of $S_{i,j}$ are denoted as $S_{i,j} = \{f_{i,j}^1, f_{i,j}^2, \ldots, f_{i,j}^{l_s}\}$, where the number of frames in $S_{i,j}$ is l_s. Let fps_{def} be the default frame rate of all cameras

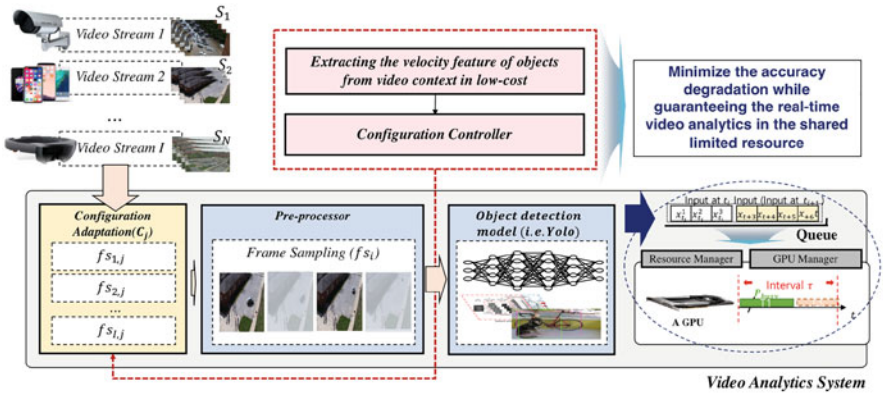

Video Stream 1
Video Stream 2
Video Stream I
S_1
S_2
S_N
Extracting the velocity feature of objects from video context in low-cost
Configuration Controller
Minimize the accuracy degradation while guaranteeing the real-time video analytics in the shared limited resource
Configuration Adaptation(C_j)
$fs_{1,j}$
$fs_{2,j}$
$fs_{I,j}$
Pre-processor
Frame Sampling (fs_i)
Object detection model (i.e.Yolo)
Queue
Resource Manager
GPU Manager
Interval τ
A GPU
Video Analytics System

Fig. 1 An illustration of our proposed video analytics system

(By default, $fps_{def} = 30$),[1] respectively. The generated raw frames are transferred to the VA system. We assume l_s is same with fps_{def}.

The pre-processor samples the received raw frames of each video stream in a certain frame rate. The frame rate is considered as a main configuration knob and controlled by the configuration controller. Let $fs_{i,j}$ denote the frame rate of i-th video stream for j-th segment. There are allowable candidates which the configuration controller can decide for frame rate knob. The candidate set of frame rate is denoted as S and are sorted in a descending order (e.g. $S = \{30, 15, 10, 6, 5, 3, 2, 1\} fps$. For simplicity, we use the divisors of fps_{def} for S. Then, the frame rate knob of i-th video stream for j-th segment is defined as $fs_{i,j}$. Let $C_j = \{fs_{1,j}, fs_{2,j}, \ldots, fs_{I,j}\}$ denote the configurations of I video streams on j-th segment, decided by the configuration controller.

The sampled frames of each video stream are fed into the queue and processed on a single GPU for VA by an object detection model (i.e. Yolo). A GPU is entirely occupied by a video stream at a time and the usage time of the GPU is used as a resource. We assume that the processing time of the object detection model on a frame is static since the GPU tends to be stable and static for processing each operation of the object detection model.

Lastly, the model recognizes objects and draws bounding boxes around them in each sampled frame of video streams.

Based on the detection results on the sampled frames of each video stream from the object detection model, the system predicts and finalizes the detection results of all frames in j-th segment of each video stream (including the frames not sampled). The accuracy of the j-th segment in each video stream is calculated as the average accuracy per frame of a segment. In the same way as existing VA systems, the

[1] We assume that the default frame rate of all cameras is same.

detection results of the last sampled frame are used for the frames not sampled in a segment.

2.1 System Objectives

We formulate the VA problem to optimize the average accuracy of analyzing live video streams by adapting configurations under limited resource capacity over the time horizon.

We divide the continuous time to discrete time slots. The length of a time slot is defined as τ, which is defined as the segment length l_s. Therefore, the j-th time slot is defined as the time at which the j-th segment, which has generated during $(j-1)$-th time slot, have to be processed. Then, in the j-th time slot, the problem is to process the j-th segments of video streams, denoted as $\{S_{i,j} | \forall i \in [I]\}$.

The objective of our VA system is to make an optimal configuration decision for its VA problem over video streams at each time slot. Then, the configuration decision vectors for j-th segments is defined as follows:

$$C_j = \{fs_{1,j}, fs_{2,j}, \ldots, fs_{I,j}\}. \tag{1}$$

The total processing time on the GPU for all frames sampled in the configuration C_j for j-th segments of video streams is modeled as follows:

$$PT(C_j) = \sum_{i \in I} fs_{i,j} * l_p, \ \ where \ fs_{i,j} \in S. \tag{2}$$

Here, $fs_i(t)$ is decided among the candidates of S.

The accuracy of analyzing a video stream is affected by several factors: the video context, the object detection model, the frame resolution, rate and bitrate of the video stream [9]. Since we assume that our VA system receives each video stream in high enough bitrate and uses a single object detection model, which has a fixed input size, the accuracy depends only on the video context, the frame rate of the video stream. Then, the accuracy model is defined as $a_{i,j}(fs_j)$ which represents the VA accuracy on j-th segment of i-th video stream with fs_j. In i-th video stream, there is a trade-off between the accuracy $a_{i,j}(fs_{i,j})$ and the processing time $fs_{i,j} * l_p$. Increasing the frame rate may increase the accuracy but will also increase processing time.

Then, we formulate the VA problem to find the configuration decision vector C_j optimizing the trade-offs of video streams as follows:

$$\underset{C_j}{\text{maximize}} \sum_{i \in I} a_{i,j}(fs_{i,j}) \text{ subject to } PT(C_j) \leq \tau. \tag{3}$$

The objective is to maximize the sum of the accuracies of I video streams for j-th segment. The constraint says that the total processing time cannot exceed the time slot length τ, in order to keep the queue stable. The VA problem is a multiple-choice knapsack problem (MCKP). A brute-force solution to the VA problem would take a running time of $O\big(I \cdot |S|\big)$. Obviously, it is impractical to search a large space of the configuration by profiling the accuracy models of video streams in each segment. Moreover, the accuracy model $a_{i,j}(fs_{i,j})$ is non-linear function and cannot be formulated cleanly in analytic form with the lack of analytic understanding on theoretical properties of the object detection models. It is necessary to design an efficient algorithm resolving the time complexity and the non-linearity.

2.2 *Velocity Based Configuration Adaptation Algorithm*

In this procedure, we consider the l_s raw frames of j-th segment generated from each live video stream. Then, we decided the sampling frame rate $fs_{i,j}$ for the l_s raw frames of each live video stream, using the velocity of objects on 1-th raw frame, $f_{i,j}^1$. Firstly, since $f_{i,j}^1$ is sampled in default, we have the bounding boxes of detected objects $B_1 = \{b_{1,1}, b_{1,2}, \ldots, b_{1,q}, \ldots, b_{1,Q}\}$ on $f_{i,j}^1$, where $b_{1,q}$ is the bounding box of q-th object detected. We extract K tracks on $f_{i,j}^1$, denoted as $\{tr_{1,k} | \forall k \in [K]\}$, using an optical flow method which extracts many tracks of feature points over contiguous frames. Let $tr_{1,k} = \{p_{1,k}, p_{2,k}, \ldots, p_{l,k}\}$, $k \in [K]$ denote the track of k-th feature point at $f_{i,j}^1$ where l is the track length ($l = 5$) and $p_{h,k} = \{val_{h,k}^x, val_{h,k}^y\}$ is the coordinates of k-th feature point at h-th time.

Secondly, we estimate the velocity of each object by calculating the average movement distance per frame of feature points included in the detected bounding box of an object on $f_{i,j}^1$. The tracks of M feature points included in the detected box $b_{1,q}$ are denoted as $TR_1^q = \{tr_{1,1}, tr_{1,2}, \ldots, tr_{1,M}\}$ where $tr_{1,m} = \{p_{1,m}, p_{2,m}, \ldots, p_{l,m}\}, m \in [M]$. The delta of the track of m-th feature point over its track length is calculated and denoted as $(\Delta tr_{1,m}^x, \Delta tr_{1,m}^y) = \frac{p_{l,m} - p_{1,m}}{trackLen} = \frac{(val_{l,m}^x, val_{l,m}^y) - (val_{1,m}^x, val_{1,m}^y)}{trackLen}$. Based on it, the estimated velocity of $b_{1,q}$ is defined as follows:

$$v_{1,q} = (v_{1,q}^x, v_{1,q}^y), \tag{4}$$

where

$$v^x_{1,q} = \frac{1}{|M|}\sum_{i=1}^{|M|} \Delta tr^x_{1,i},$$
$$v^y_{1,q} = \frac{1}{|M|}\sum_{i=1}^{|M|} \Delta tr^y_{1,i}. \tag{5}$$

As a result, we estimate how objects on $f^1_{i,j}$ move per frame in a segment with $\{v_{1,q}|\forall q \in [Q]\}$. In order to decide the best frame rate intuitively which maximizes the accuracy with minimum resource consumption, we focus on the velocity of the fastest object $q_{max} = argmax_{q\in Q}\{v_{1,q}\}$ and, based on it, adapt the sampling frame rate $fs_{i,j}$ for j-th segment. First, we predict the q_{max}-th object's bounding boxes over j-th segment by shifting the q_{max}-th object's bounding box on $f^1_{i,j}$, $b_{1,q_{max}}$, with its velocity of $v_{1,q_{max}}$. Let $\{b^*_{t,q_{max}}|\forall t \in [l_s]\}$ denote the predicted bounding boxes of q_{max}-th object over the l_s frames of j-th segment. It is performed by shifting the coordinates of $b_{1,q_{max}}$ based on $v_{1,q_{max}}$ and predicting those of $\{b^*_{j,q_{max}}|\forall t \in [l_s]\}$, as defined as follows:

$$\begin{aligned} rect^*_{t,q_{max}} &= \left(x^*_{t,q_{max}},\ y^*_{t,q_{max}},\ w^*_{t,q_{max}},\ h^*_{t,q_{max}}\right) \\ &= \left(x_{1,q_{max}} + t * v^x_{1,q_{max}},\ y_{1,q_{max}} + t * v^y_{1,q_{max}},\right. \\ &\qquad \left. w_{1,q_{max}},\ h_{1,q_{max}}\right), \forall t \in [l_s]. \end{aligned} \tag{6}$$

Here, $(x^*_{t,q_{max}}, y^*_{t,q_{max}}, y^*_{t,q_{max}}, w^*_{t,q_{max}})$ is the left and top coordinates of the predicted bounding box $b^*_{t,q_{max}}$. We assume that the width and height of $b^*_{t,q_{max}}$ are static in this short time. Finally, we decide the sampling frame rate $fs_{i,j}$ based on the reciprocal of the longest interval from 1-th frame to the t-th frame whose predicted bounding box $b^*_{t,q_{max}}$ shows IoU above the desired threshold η_v (By default, $\eta_v = 0.5$)with $b_{1,q_{max}}$. In practice, we choose the closest one among S for the sampling frame rate $fs_{i,j}$.

$$fs_{i,j} = \frac{fps_{def}}{max\{t|IoU(b^*_{t,q_{max}}, b_{1,q_{max}}) > \eta_v, \forall t \in [l_s]\}}. \tag{7}$$

To minimize the accuracy degradation, we prevent the prediction from being far from $b_{1,q_{max}}$ in a certain level with the desired threshold and adapt the sampling frame rate depending on how fast the prediction exceeds the desired threshold over l_s frames.

By default, our VA system runs only based on the frame rate $fs_{i,j}$ of each live video stream decided by the aforementioned algorithm for j-th segment. Then, the default configuration of video streams is denoted as $C^0_j = \{fs^0_{1,j}, fs^0_{2,j}, \ldots, fs^0_{I,j}\}$.

However, if $PT(C_j^0)$ exceeds τ, we additionally adapt the frame rates of video streams in C_j^0. That is, in order to remove the total exceeded time $ET = PT(C_j^0) - \tau$, the configuration C_j^0 is adapted to be cheaper.

Firstly, ET is split into I smaller segments, denoted as $ET_i, i = 1, \ldots, I$, and distributed to all live video streams. Then, each live video stream is required to reduce ET_i from $PT(C_j^0)$. ET_i is decided in proportion to $fs_{i,j}$ fairly because $fs_i(t)$ reflects the resource demand of i-th live video stream, defined as follows:

$$ET_i = \left(PT(C_j^0) - \tau\right) * \frac{fs_{i,j}}{\sum_{i \in [I]} fs_{i,j}}. \tag{8}$$

Secondly, each video stream $i = 1, \ldots, I$ independently starts to adapt its frame rate $fs_{i,j}^0$ cheaper for reducing ET_i. The adaptation is conducted one by one iteratively in a greedy way for maximizing the resource consumption reduction while minimizing the accuracy degradation until $ET_i \leq (fs_{i,j}^0 - fs_{i,j}) * l_p$. As a result, $C_j = \{fs_{1,j}, fs_{2,j}, \ldots, fs_{I,j}\}$ is determined.

3 Evaluation

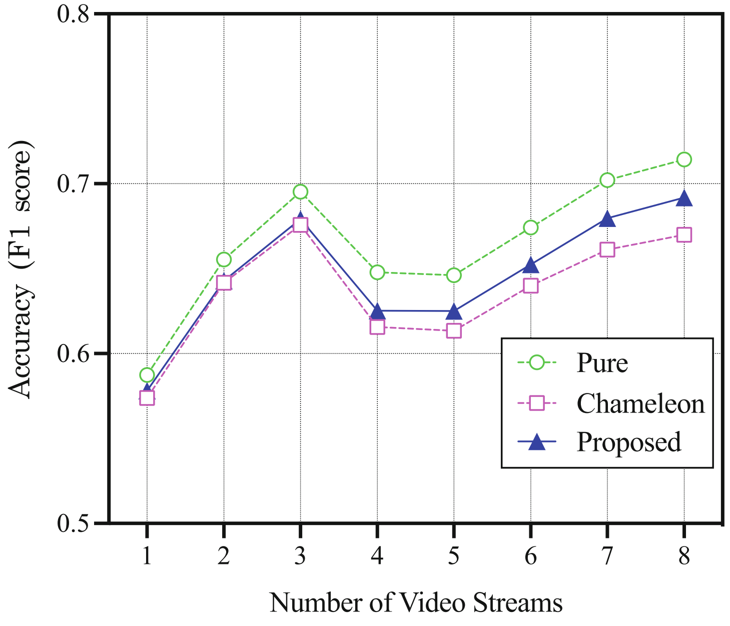
In our experiment, we deploy our implemented VA system on the physical machine equipped with CPU, Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz, GPU, GeForce GTX 1080(8 GB), memory 16 GB in order to process all analysis tasks on video streams. The VA task is a DNN based object detection. We use CUDA 9.0 and cuDNN 7.0 [7] to accelerate the DNN speed based on a GPU. We use Keras 2.2.4 [6] APIs with Tensorflow framework 1.12 [10] in order to execute an object detection model of Yolov3. We determine a particular DNN called Yolo, which input size is 608×608. It can detect 80 object classes and are pre-trained on the COCO image dataset. We use a subset of surveillance videos and annotations from VIRAT 2.0 Ground dataset [8]. Based on them, we construct 9 video streams from nine real-world static cameras deployed in different area. Each video stream generates 10800 frames totally in 30 fps and 1920×720p (for 6 video streams) or 1280×720p (for 3 video streams). We implement optical flow algorithm used in the proposed algorithm by Lucas–Kanade method.

In these experiments, the proposed algorithm was compared with existing aforementioned well-known algorithm in existing VA systems such as Chameleon, which are based on online profiling. For the existing algorithm, we use the original parameters and configurations described in its paper, such as segments of profiling window $w = 4$, segment interval $T = 4$, profiling interval $t = 1$. We assume the maximum performance we can achieve is the analytics performance with the expensive configuration (i.e. 30 fps) and it is denoted as Pure in this paper. In order to evaluate the VA performance of the proposed algorithm on accuracy and resource consumption, we measure F1 score, the harmonic mean of precision and recall, and average GPU processing time per frame. We calculate an accuracy of a single frame by comparing the detected objects with the ground truth box. We identify true positives in the F1 score using a label-based condition, which checks if the bounding box has the same label and sufficient spatial overlap with some ground truth box. This IoU threshold is set as 0.5.

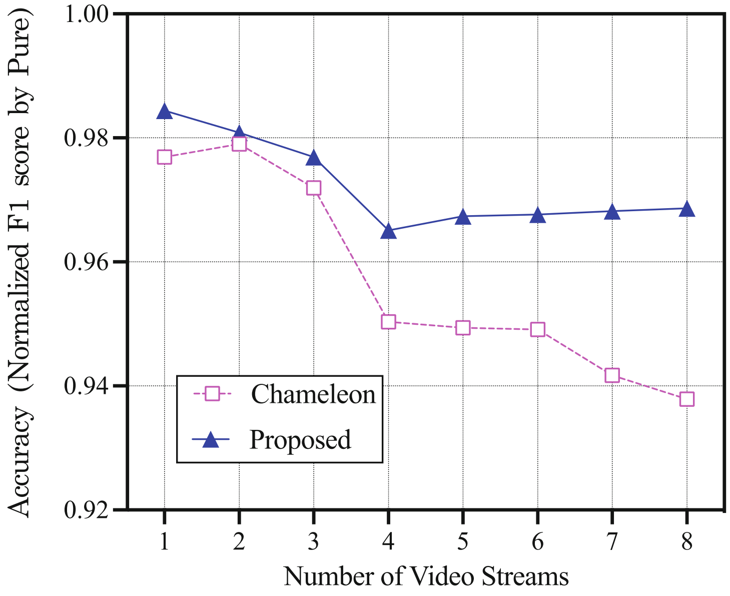
In this experimental environment, we evaluate the VA performance of each algorithm by measuring accuracy (F1 score) and resource consumption (GPU processing time) for multiple video streams in limited resource scenario. We apply several video streams concurrently (from 2 video streams to 9 video streams) among 9 video streams. Figure 2 shows the accuracy and normalized accuracy performance of proposed algorithm over multiple video streams. The x-axis represents the number of streams to be processed, ranging from two to nine. The proposed algorithm shows higher accuracy in every cases compared to existing VA systems.

In Fig. 2a, b, the proposed algorithm shows higher accuracy for multiple streams in every cases. In Fig. 2b, the accuracy of the proposed algorithm is almost constant above five streams, since the configuration is already adapted to the cheapest one decided within the proposed algorithm's constraint. By relaxing this constraint we impose, it is possible to reduce the resource consumption although the accuracy decreases. However, we judge that the accuracy below the certain level is not meaningful, so this constraint is needed to guarantee the least desired accuracy.

Figure 3 shows the latency performance of the proposed algorithm over multiple video streams. The proposed algorithm also shows better performance on resource consumption in every cases. However, as the number of streams increases, the resource consumption of Chameleon drops sharply from two streams to eight streams while those of the proposed algorithm falls and stops to a proper level from six streams. Compared to Chameleon, the proposed algorithm finds better configurations on accuracy and resource consumption by profiling. Meanwhile, Chameleon, which profiles limited configuration candidates basically, profiles fewer configuration candidates feasible in divided resource allocated to each video stream. Obviously, it is not enough to find efficient configurations and realize a desired accuracy. Especially, as the number of video streams increases and the resource allocated to each video stream decreases, configuration candidates to be profiled decreases. Consequently, although reducing significantly its profiling load and resource consumption, it shows unacceptable accuracy with this deficient profiling. The resource consumption of the proposed algorithm is also almost constant from five streams with its aforementioned constraint.



(a)



(b)

Fig. 2 Accuracy performance of the proposed algorithm over multiple video streams, (**a**) F1 score, (**b**) Normalized F1 score by Pure

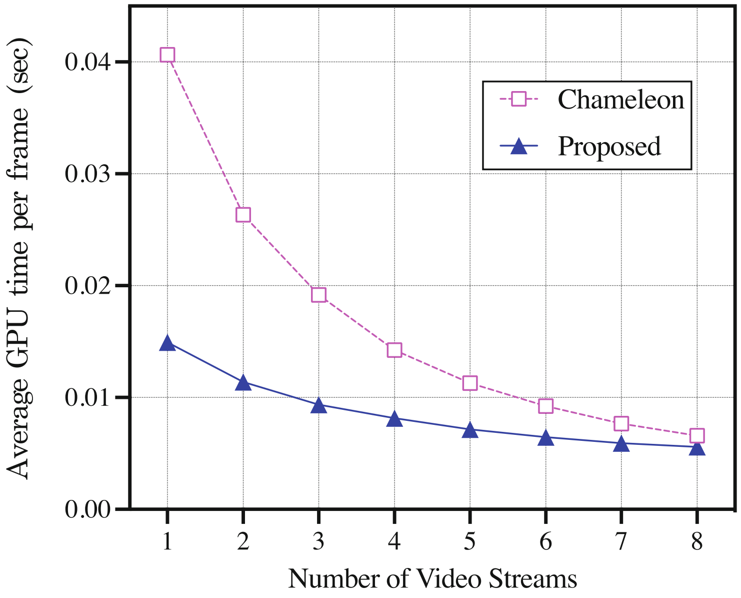


Fig. 3 Latency performance of the proposed algorithm over multiple video streams (Average GPU time per frame)

4 Conclusion

In this paper, we propose a low-cost VA system with velocity based configuration adaptation to find the best configuration decision of frame sampling rate for multiple video streams in order to minimize the accuracy degradation in the shared limited resource. Firstly, the frame sampling rate knob of each video stream is adapted optimally based on the velocity feature of objects extracted from its video context in low-cost. Secondly, the frame sampling rate knob of each video stream is adapted additionally in a greedy way, considering a limited resource shared by multiple video streams. As a result, the proposed VA system outperforms the existing VA systems in terms of accuracy and resource consumption.

Acknowledgments This work was partly supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korean Government (MSIT) (No. 2017-0-00294, Service Mobility Support Distributed Cloud Technology) and (No. 2020-0-00537, Development of 5G based low latency device—edge cloud interaction technology).

References

1. Z. Fang, D. Hong, R.K. Gupta, Serving deep neural networks at the cloud edge for vision applications on mobile platforms, in *Proceedings of the 10th ACM Multimedia Systems Conference* (ACM, New York, 2019)
2. C.-C. Hung, et al., Videoedge: processing camera streams using hierarchical clusters, in *2018 IEEE/ACM Symposium on Edge Computing (SEC)* (IEEE, Piscataway, 2018)
3. J. Jiang, et al., Chameleon: scalable adaptation of video analytics, in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (ACM, New York, 2018)
4. S. Jain, et al., Scaling video analytics systems to large camera deployments, in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications* (ACM, New York, 2019)
5. D. Kang, et al., NoScope: optimizing neural network queries over video at scale. Proc. VLDB Endowment **10**(11), 1586–1597 (2017)
6. Keras: the python deep learning library. https://keras.io/
7. NVIDIA developer. https://developer.nvidia.com/
8. S. Oh, et al., A large-scale benchmark dataset for event recognition in surveillance video, in *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2011* (IEEE, Piscataway, 2011)
9. X. Ran, et al., Deepdecision: a mobile deep learning framework for edge video analytics, in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (IEEE, Piscataway, 2018)
10. Tensorflow: an open source machine learning library for research and production. https://www.tensorflow.org/
11. H. Zhang, et al., Live video analytics at scale with approximation and delay-tolerance, in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI'17)* (2017)