Accelerating Multi-Object Tracking in Edge Computing Environment with Time-Spatial Optimization

Mengyang Liu*†, Anran Tang*, Huitian Wang*, Lin Shen*,
Yunhan Chang*, Guangxing Cai*, Daheng Yin*, Fang Dong* and Wei Zhao‡§,
*School of Computer Science and Engineering, Southeast University, Nanjing, China
†School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China
‡Institute for Artificial Intelligence, Comprehensive National Science Center, Hefei, China
§School of Computer Science and Technology, Anhui University of Technology, Ma'anshan, China
{myliu, anrant, huitwang, linshen, zachchang, guangxingcai, yindaheng98, fdong}@seu.edu.cn,
zhaoweistuart@gmail.com

***Abstract*—Real-time MOT task is a key issue in computer vision, which has broad application prospects in security, autopilot, and augmented reality. Benefit from the continuous progress of DNN, the accuracy of the MOT algorithm has been significantly improved. Nevertheless, limited to computing power, achieving real-time DNN-based MOT is difficult in IoT systems. In reality, there are many wasteful and unnecessary computations in traditional frame-by-frame full-size video analysis. Therefore, in this paper, we propose a strategy that optimizing the execution of a traditional MOT pipeline in the dimension of time and space. In the temporal dimension, DNN only works in periodic keyframes while using a predictive model for quickly generating results in the rest of these frames. In the spatial dimension, we design an image density region discriminator based on the DBSCAN algorithm, used to narrow down the input size of DNN. An edge device is introduced to perform end-edge collaborative computing to further accelerating the execution. Additionally, an end-edge parallel computing mechanism is designed that performing dynamic decisions based on the computing power and network environment between end and edge. Moreover, we rebuild the DNN model by TensorRT to optimize the model structure of DNN. By integrating the above acceleration approaches, the system can achieve 17.6∼38.1× speedup ratio, while with 3%∼10.4% absolute tracking accuracy sacrifice and can be deployed in an unstable network environment.**

***Index Terms*—Multi-Object Tracking, Deep Learning, Inference Optimization, Edge Intelligence**

I. Introduction

Multi-Object Tracking (MOT) is a significant module in computer vision. As Fig. 1 illustrates, the goal of MOT is to detect and track every predefined target in video frames. For each target that appears in the video, we can get its identity(ID), bounding box, and trajectory information in each frame it arises.

With the development of Internet of Things recent years, more and more MOT based applications has emerged, e.g., video surveillance [1], autonomous driving [2], augmented reality [3], etc. But these applications often have high demand in accuracy and real-time. Benefit from the maturity of deep learning, the accuracy requirement of applications has been satisfied in part, but there is still a big challenge in real-time due to the conflict between the limited computing power of IoT devices and the resource-hungry of deep learning. In reality, insufficient real-time performance will also lead to a decline in accuracy because of the frame loss. Therefore, the MOT algorithm that performing well in real-time performance or accelerating strategies for video analysis has attracted more and more attention.

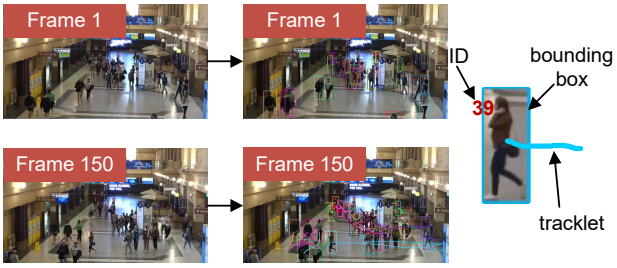


Fig. 1. An illustration of MOT task and its output.

For the design of the MOT algorithm, the joint detection and tracking (JDT) paradigm has been proposed, which used a single network to extract two different features required for the detection and re-ID, to meet the real-time requirement for applications. Additionally, there are some accelerating methods have been proposed for deep vision applications that maybe help speed up the execution of MOT tasks. On the one hand, we can compress the input to reduce the computation, such as frames refining [4], dynamic Region-of-Interest(ROI) encoding [5], narrowing down searching space [6], etc. On the other hand, model optimization was general and effective for DNN, e.g. branch pruning [7], tensor quantization [8], and operator fusion [9]. Nevertheless, due to the complexity of MOT and the computing limits of devices, most end devices are still far short to meet the requirements of real-time MOT applications.

To this point, it is worth noting that most JDT works [10]

[11] have already been able to realize real-time execution on some professional GPUs such as NVIDIA Tesla V100. However, for the actual deployment devices of applications, such as NVIDIA Jetson Nano, its performance is far from request. To solve this problem, introducing edge computing to supplementary computing resources is a common and effective way. By offloading the intensive inference to a powerful edge server, the inference latency will be reduced significantly. Existing end-edge offloading strategies can be divided into two categories. One is integral offloading [12], which offloads the whole inference task from the end to the edge. Integral offloading is easy to use but cannot utilize the overall performance of the system. The other one is partial offloading [13], which can partition the data or model to realize the parallel computing. In designing an end-edge collaborative mechanism, transmission latency is a vital factor that should be concerned for real-time video analysis. It is necessary to design a strategy for dynamically adjusting the burden of offloading tasks according to the current network situation.

To address these problems, we design an MOT acceleration system that can let the application reach real-time performance by utilizing a proposed time-spatial optimization MOT pipeline and an end-edge cooperative DNN inference mechanism. To reduce the MOT computing amount, we propose a time-spatial execution optimizing strategy, which has a frame classifier and a region divider to reduce the input of DNN. The system uses FairMOT [11], a SOTA MOT JDT algorithm, as the basic algorithm and implements its TensorRT version to realize model optimization. Before the inference, the input is partitioned into two patches where one inferencing locally, the other will be processed by the edge. The end-edge computing power and current network situation are factors of the partitioning and scheduling decision. We implement this system using a realistic prototype with Jetson Nano 2G and laptop PC. Experimental evaluations show the system can achieve 17.6$\sim$38.1$\times$ latency speedup over the local execution with 3%$\sim$10.4% absolute tracking accuracy sacrifice and can make the best effort to finish the task in the harsh network environment.

The main contributions of this paper are as follows.

- To reduce the redundant processing in time and space, we propose a time-spatial MOT execution optimizing strategy that improves the traditional MOT pipeline and can accelerate the execution of the algorithm by reducing the excessive DNN computation.
- To perform an effective and robust computation offloading between end and edge, we design a collaborative MOT end-edge mechanism that can dynamically partition data and send them to the end device and edge server for realizing parallel inference and adaptation to the network.
- We implement a prototype system based on a TensorRT version of FairMOT to optimize the structure of the model. The comprehensive experiments demonstrate that our proposed schemas can achieve a 38.1$\times$ speedup ratio with 10.4% accuracy sacrifice that giving a new way to make a tradeoff between accuracy and speed.

Fig. 2. The distribution of the MOT task difficulty. White patches are noise which are unvaluable to process. Red patches are difficult to process because there are many occlusions in them. Green patches are only one or two people in one patch, so can be easily processed.

The rest of the paper is organized as follows. Section II presents related work. Section III briefly gives the background on the MOT and discusses the motivations of our ideas. Section IV provides the overview and details of our system. Section V builds the end-edge collaborative model and gives a solution. The prototype is evaluated in Section VI and Section VII concludes.

II. Related Work

Multi-Object Tracking. At present, there are two major MOT paradigms: detection based tracking (DBT) [14] [15] and joint detection and tracking (JDT) [10] [11]. The DBT paradigm treats detection as the basic task of tracking, so it always split the task into two stages and directly uses an existing object detector to generate the bounding boxes of objects and then track them inter frames. In order to tackle occlusion problems, the re-ID module is added into the DBT paradigm, which extracts every detected re-ID feature of objects. Although the re-ID module can improve tracking accuracy, it is inefficient due to the high computation for extracting features one by one. The JDT paradigm uses a single network to detect objects and extract re-ID features, which share the underlying features of the network between two tasks for reducing computation.

Accelerating Video Analysis. FFS-VA [4] constructs a three-layer filter to select keyframes and reduce the computation amount. Chameleon [16] can help make the optimal configuration of the model by analyzing video properties. Liu, et al. [5] use dynamic RoI(Region of Interest) encoding technique to focus on significant regions in video. Narrowing down the data size of the processing video is an effective way to reduce processing latency while designing according to the specific applications.

DNN Model Optimizations. Model pruning [7] can reduce the computation of the model by pruning some parameters which have little effect on results. Tensor quantization [8] compresses each model parameter to save more memory and computation. These model optimization methods can be applied in most DNN models but often need some special steps or modifications on the DL framework, which is time-consuming.

End-Edge Cooperative Inference. DeepDecision [12] is an integral offloading model, deciding to perform the task on end locally or offload to edge considering factors included accuracy, latency, model size, etc. FastVA [17] processes images on the end device and edge server simultaneously, which takes the single image as the scheduling unit. CoEdge [18] partition image and model to execute in distributed devices. By utilizing a special cooperative inference of CNN, CoEdge can generate results as same to the local processing although introducing additional transmission overhead.

III. BACKGROUND

A. Traditional Multi-Object Tracking Pipeline

In this paper, we target the real-time MOT application and use the JDT paradigm. The JDT paradigm builds a single network to finish the object detection and re-ID feature extraction tasks together, which utilize the sharing of the backbone feature to save computations. Image first needs to be preprocessed to the input size of DNN. Then the DNN will process every pixel of the input image to generate results of object detection and results of re-ID. For motion feature extraction, a simple solution is to directly utilize a motion model, e.g., the Kalman filter. Motion feature and re-ID feature are fused into a unified feature from each object, followed by building a similarity matrix between the detected and historical features. Finally, tracking is realized by the association based on this matrix based on a matching algorithm, e.g., the Hungarian algorithm.

B. Motivations

The advantage of deep learning is its accuracy, but the speed is its drawback, so there are many existing methods to accelerate the DL-based video analysis, but they still have many limitations.

Redundant processing in time. The current common practice in video analysis is feeding frames into the network frame by frame. But for real-time MOT tasks, it is too expensive so that the application always only processes the newest one and drops the rest of the frames, which incurs the loss of accuracy. The correlation of object motion between frames often ignored by the drop-frame strategy that we think can utilize to mitigate the loss of accuracy.

Unbalanced distribution of data in space. As Fig. 2 depicted, there is much noise existing in an image from the background or non-target objects, but the common practice is processing every pixel from the image, which is wasteful. Additionally, the distribution of task difficulty is unfair in a single image, so maybe we can filter out the hard part to DNN inference and the simple part to a lightweight model.

Problems in integral offloading. Offloading the inference task onto the edge server with strong GPUs can greatly improve the real-time performance of DL application, but it faces two main problems. On the one hand, after offloading the task onto the edge server, the end device must wait for the results from the edge server, which caused low utilization of the system. On the other hand, the offloading will introduce transmission delay that may cause the overall execution latency under the bad network.

IV. SYSTEM DESIGN

To address these problems, we design an integrated acceleration system. We implement the execution optimization in time-spatial and introduce edge computing while designing a dynamical collaborative mechanism. We also rebuild the DNN model with TensorRT to optimize the model structure. The system runs followed by these steps:

1) **Video frames classification.** The frame in a video sequence is classified as a global detection frame, partial detection frame with a certain period, and ordinary frame. The ordinary frame uses the motion prediction model for processing quickly. For the global and partial detection frame, the frame image is fed into the network entirely or partially based on the result of region dividing. In this way, we take full advantage of the correlation information between frames to reduce the redundancy in processing frame by frame and can achieve better accuracy than the drop-frame strategy.
2) **Region dividing.** In the global detection frame, the image will be processed by DNN entirely. Then the dense region discriminator will determine a dense region based on the results of object detection. In the partial detection frame, the image is divided into dense and sparse regions. The location of the dense region is decided in the global detection frame, where the image patch will be cropped and fed into the network. In the sparse region, the historical targets will be processed by the motion prediction model.
3) **Image partitioning and scheduling.** The network processing in the global and partial detection frame will all be orchestrated by a designed adaptive workload partitioning and scheduling mechanism. The mechanism partitions the image into two image patches in horizon-

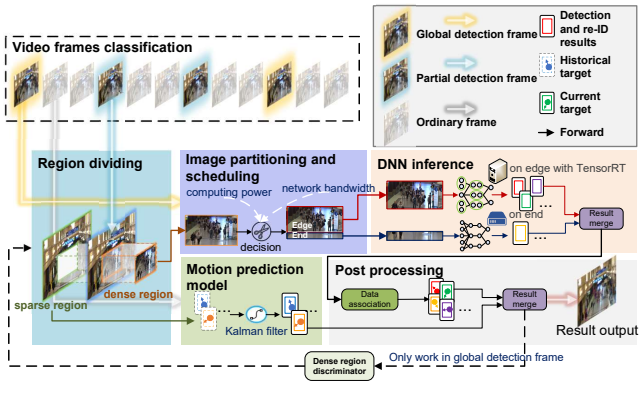
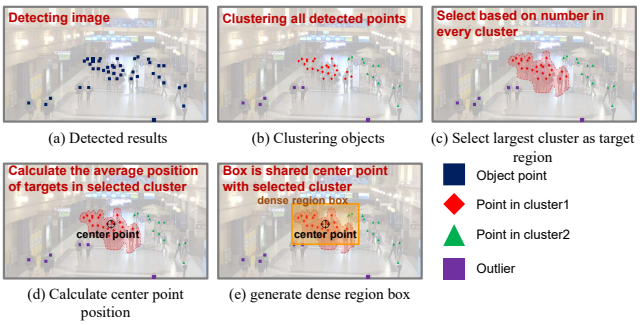


Fig. 3. The overview of the MOT acceleration system. The main of this figure illustrates the processing stages in a partial detection frame for showing more detailed information. In the global detection frame, the network processing will also be dynamic offloaded. In the ordinary frame, the results will be predicted directly by the motion prediction model.



(a) Detected results (b) Clustering objects (c) Select largest cluster as target region

(d) Calculate center point position (e) generate dense region box

Fig. 4. Steps of dense region discriminator.

tal, which is determined by the computing power and network situation. Dynamic processing is friendly to the network conditions and realizing the parallel processing on the end device and edge server.

4) **DNN inference.** We utilize the TensorRT framework as our DL deployment runtime, which can optimize our model easily and effectively for NVIDIA GPU. The results are merged from end and edge.
5) **Post processing.** After the DNN inference, the results will be associated with the historical targets in the DNN region. The results of output are merged from the results from data association and the motion prediction model.

The above workflow is illustrated in Fig. 3. In the rest of this section, we present some details of the key modules in the system.

A. Motion Prediction Model

The motion model is based on the Kalman filter. The state equation is constructed under the assumption of the linear motion of targets. The state of each object is modeled as Eq.(1). u and v represent the horizontal and vertical pixel location of the center of the target. s and r represents the scale and aspect ratio of the target's box. $\hat{u}$, $\hat{v}$ and $\hat{s}$ represents the prediction of them.

$$x = [u, v, s, r, \hat{u}, \hat{v}, \hat{s}]^\top \tag{1}$$

Depending on the linear motion assumption, we can get a prediction value of x with ease. After the prediction, the Kalman filter fuses the results from detection and prediction based on the stochastic process to update the prediction values.

B. Dense Region discriminator

In order to determine the location of the dense region, we design a dense region discriminator to generate the optimal location of the dense region based on the density information from the results of object detection in the global detection frame. There are some steps in discrimination shown in Fig. 4. First, see Fig. 4(a) and (b), the detected results will be clustered into multiple clusters by the DBSCAN algorithm. Second, we will select the cluster with the largest number of objects to define the densest cluster as illustrated in Fig. 4(c). In the end, calculate the position of the center point and generate the dense region box, whose size is predefined, with this center point indicated in Fig. 4(d) and (e).

TABLE I
NOTATION.

Notation	Description
r_{origin}	The origin resolution of image
r_{end}	The resolution of the image patch for processing on end
r_{edge}	The resolution of the image patch for processing on edge
K	The number of network sampling
i	The current frame number
p	Image compression ratio
B	The current network bandwidth
B_j	The network bandwidth in j^{th} frame
L_{send}	The latency of offloading image patch
P_{end}	The latency of inference on end
P_{edge}	The latency of inference on edge
pr	The horizontal partitioning ratio

C. Data Partitioning and End-Edge Collaborative Inference

In this work, we exploit data parallelism to partition CNN inference work on the end device and edge server. Under data parallelism, a preprocessed image is divided into two image patches sent to the end device and edge server. End device and edge server generate the results of object detection and re-ID features from the DNN with their image patch. After finishing the inference in edge, the results from it will transmit back to the end and merged with the results from it, which followed the data association processing.

For minimizing the waste of resources and latency of execution, the size of offloading image is decided in real-time according to the computing power and network bandwidth. For this point, an offline profiling and preparing stage in the online execution is essential.

V. Adaptive Workload Partitioning and Scheduling

In this section, we first model how to partition the image and schedule the task onto the end or edge for achieving the minimum execution latency. And then we give a solution to this problem.

A. Problem Modeling

Assuming the resolution of the incoming image is r_{origin}, the scheduling needs to make a reasonable partition of the frame based on the computing power of the end device and edge server and also the current network conditions. After partitioning the original image with a partitioning ratio pr to horizontal, the system will generate two image patches with r_{end} resolution for the end device and r_{edge} resolution for the edge server. The computing power of the end device and edge server are evaluated from the profiling results with different resolutions, which are denoted by P_{end} and P_{edge}. The transmission delay L_{send} is computed by B and the size of the transmitted image.

The notations used in the problem formulation and the algorithm design are shown in TABLE I. The dynamic partitioning

problem can be formulated as an optimization problem in the following way.

$$\underset{pr}{\text{minimize}} \quad \max[L_{send}(r_{edge}) + P_{edge}(r_{edge}), L_{send}(r_{edge}) + P_{end}(r_{end})] \tag{2}$$

$$s.t. \quad r_{end} = r_{origin} \cdot pr \tag{3}$$

$$r_{edge} = r_{origin} \cdot (1 - pr) \tag{4}$$

$$B = \frac{1}{K} \sum_{j=i-K}^{i-1} B_j \tag{5}$$

$$L_{send} = \frac{1}{p} \cdot r_{edge} \cdot \frac{1}{B} \tag{6}$$

$$i > K \tag{7}$$

The objective (2) is to minimize the overall inference latency that included the inference latency from the end device and the edge server and plus the transmission latency. Constraint (3) and (4) say that the partitioning will not discard any pixels and partition the image by the pr. Constraint (5) says that the bandwidth is combining with the previous K frame processing environment. Constraint (6) says that the transmission latency is influenced by the compression ratio, bandwidth, and resolution. Constraint (7) says that the scheduling can only work when the current frame number over K.

B. Model Solving

For accelerating the solving of this problem, we limit the size of the image patch with a partitioning ratio collection S_r, which satisfies the input condition of DNN and reduces the search space of the solution.

Alg. 1 is the designed solution for the dynamic partitioning and scheduling problem, which strictly limiting the options collection of the decision variable to simplify the solution.

Algorithm 1 Network-Related Partitioning Algorithm

Input: B_j, bandwidth of the last j frames
Output: pr^*, optimal partitioning ratio

$S_r \leftarrow$ The list of optional pr
$B \leftarrow \frac{1}{K} \sum_{j=i-K}^{i-1} B_j$
$L_{min} \leftarrow \infty$
for pr in S_r **do**
  $r_{end} \leftarrow r_{origin} \cdot pr$
  $r_{edge} \leftarrow r_{origin} \cdot (1 - pr)$
  $L_{send} \leftarrow \frac{1}{p} \cdot r_{edge} \cdot \frac{1}{B}$
  $P_{edge} \leftarrow P_{edge}(r_{edge})$
  $P_{end} \leftarrow P_{end}(r_{end})$
  $L \leftarrow \max(L_{send} + P_{edge}, L_{send} + P_{end})$
  if $L < L_{min}$ **then**
    $L_{min} \leftarrow L, pr^* \leftarrow pr$
  end if
end for
return pr^*

VI. Performance Evaluation

In this section, we evaluate the performance of the designed system that including a time-spatial optimization strategy, a TensorRT model optimization implementation, and a dynamic end-edge collaborative inference mechanism.

A. Experiment Setup

Experimental platform. We build a real system prototype with a laptop whose GPU is NVIDIA GTX1050 as the edge server and an NVIDIA Jetson Nano 2G as the end device. We use WiFi to connect the end device and the edge server with a router.

Datasets and measurement. We use the MOT20-01 video sequence as our experimental dataset and choose MOTA and FPS measurements to evaluate the performance of the system that are common in evaluations of the MOT algorithm.

B. Performance Comparison

We carried out a combination experiment on our system with different configurations compared to the original FairMOT executed on end locally to see the acceleration effect. Fig. 5 shows the results of the comparison. The time-spatial strategy can achieve a good acceleration effect, but the longer the time interval is used, the higher accuracy lost. Secondly, TensorRT is effective in acceleration without the sacrifice of accuracy. Finally, the end-edge collaborative inference mechanism can significantly accelerate the execution compared to the end locally execution while there is only a tiny loss of accuracy. We find that the system can achieve 17.6∼38.1× speedup ratio, while with 3%∼10.4% absolute tracking accuracy sacrifice integrated three acceleration methods.

C. Adaptability to Network Fluctuations

We limit bandwidth from 1MBs to 50KBs during 150∼320s of the execution. As depicted in Fig. 6, the dynamic offloading can adaptively adjust the offloading image size to save the

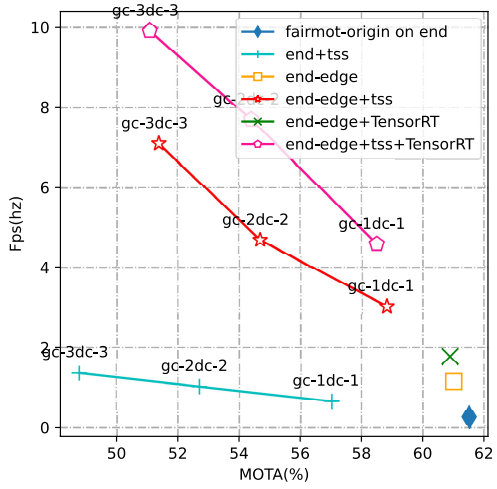


Fig. 5. Comparison of the combination of different methods. *gc* presents the number of global detection cycle. *dc* presents the number of detection cycle. *end* presents the local processing on end. *end-edge* presents the end-edge collaborative execution. *tss* presents the use of time-spatial strategy. *TensorRT* presents the enable of TensorRT model optimization.

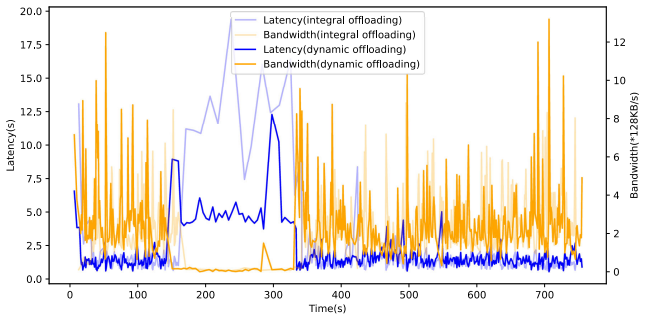


Fig. 6. Comparison of the dynamic offloading mechanism and the integral offloading. Normal bandwidth is 1MB/s and the bad bandwidth is limited in 50KB/s by wondershaper tool to simulate bad network environment.

transmission cost under the bad network, which can save 2.5∼4.5× of the overall processing delay.

VII. Conclusion

In this paper, we propose a time-spatial optimization approach that reducing the redundancy and unfairness processing existing in the traditional MOT pipeline. To effectively deploy in the end-edge environment, we exploit TensorRT to optimize the DNN model for further accelerating the inference. In order to increase the utilization and robustness in the end-edge environment, an end-edge collaborative inference mechanism, which can make the dynamic decision by computing power of end device and edge server and the network situation, is designed. Experimental evaluations show 17.6∼38.1× latency speedup over the local approach while losing 3∼10.4% absolute tracking accuracy while the system can adapt to the network conditions.

Acknowledgment

This work is supported by National Key R&D Program of China 2018AAA0100500, National Natural Science Foundation of China under Grants, No. 61872079, 61632008, 61702096, 61702097, 61902065, 61972085, 61906040, the Natural Science Foundation of Jiangsu Province under grant BK20190345, BK20190335, Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No.BM2003201, Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grants No.93K-9, the University Synergy Innovation Program of Anhui Province No. GXXT-2020-012, and partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization and Collaborative Innovation Center of Wireless Communications Technology. We also thank the Big Data Computing Center of Southeast University for providing the experiment environment and computing facility.

References

[1] S. Y. Nikouei, Y. Chen, and T. R. Faughnan, "Smart surveillance as an edge service for real-time human detection and tracking," in *2018 IEEE/ACM Symposium on Edge Computing, SEC 2018, Seattle, WA, USA, October 25-27, 2018*. IEEE, 2018, pp. 336–337.

[2] A. Buyval, A. Gabdullin, R. Mustafin, and I. Shimchik, "Realtime vehicle and pedestrian tracking for didi udacity self-driving car challenge," in *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 2018, pp. 2064–2069.

[3] H. Uchiyama and E. Marchand, "Object detection and pose tracking for augmented reality: Recent approaches," *18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, 2012.

[4] C. Zhang, Q. Cao, H. Jiang, W. Zhang, J. Li, and J. Yao, "FFS-VA: A fast filtering system for large-scale video analytics," in *Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018, Eugene, OR, USA, August 13-16, 2018*. ACM, 2018, pp. 85:1–85:10.

[5] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *The 25th Annual International Conference on Mobile Computing and Networking, MobiCom 2019, Los Cabos, Mexico, October 21-25, 2019*, S. A. Brewster, G. Fitzpatrick, A. L. Cox, and V. Kostakos, Eds. ACM, 2019, pp. 25:1–25:16.

[6] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Choi, and T. R. Faughnan, "Real-time human detection as an edge service enabled by a lightweight CNN," in *2018 IEEE International Conference on Edge Computing, EDGE 2018, San Francisco, CA, USA, July 2-7, 2018*. IEEE Computer Society, 2018, pp. 125–129.

[7] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 1398–1406.

[8] Z. He and D. Fan, "Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 11 438–11 446.

[9] L. Du, Y. Du, Y. Li, J. Su, Y. Kuan, C. Liu, and M. F. Chang, "A reconfigurable streaming deep convolutional neural network accelerator for internet of things," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 65-I, no. 1, pp. 198–208, 2018.

[10] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XI*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12356. Springer, 2020, pp. 107–122.

[11] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," arXiv preprint arXiv:2004.01888, 2020.

[12] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*. IEEE, 2018, pp. 1421–1429.

[13] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu, "Deepwear: Adaptive local offloading for on-wearable deep learning," *IEEE Trans. Mob. Comput.*, vol. 19, no. 2, pp. 314–330, 2020.

[14] H. Fu, L. Wu, M. Jian, Y. Yang, and X. Wang, "MF-SORT: simple online and realtime tracking with motion features," in *Image and Graphics - 10th International Conference, ICIG 2019, Beijing, China, August 23-25, 2019, Proceedings, Part I*, ser. Lecture Notes in Computer Science, Y. Zhao, N. Barnes, B. Chen, R. Westermann, X. Kong, and C. Lin, Eds., vol. 11901. Springer, 2019, pp. 157–168.

[15] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017*. IEEE, 2017, pp. 3645–3649.

[16] J. Jiang, G. Ananthanarayanan, P. Bodík, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*, S. Gorinsky and J. Tapolcai, Eds. ACM, 2018, pp. 253–266.

[17] T. Tan and G. Cao, "Fastva: Deep learning video analytics through edge processing and NPU in mobile," in *39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020*. IEEE, 2020, pp. 1947–1956.

[18] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "Coedge: Cooperative DNN inference with adaptive workload partitioning over heterogeneous edge devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, 2021.